11-1-1994

Distributed Scheduling in Finite Capacity Networks

Perry Fizzano
Dartmouth College

Clifford Stein
Dartmouth College

Distributed Scheduling in Finite Capacity Networks

Perry Fizzano & Clifford Stein
Department of Computer Science
Dartmouth College

In this paper, we show that a simple distributed algorithm for network scheduling in arbitrary m processor networks with unit capacity links is an $O(\log m)$-approximation algorithm if the optimal schedule length is sufficiently large. We will assume that there are m machines or processors labeled $p_1, p_2, \ldots, p_m$, such that processor p_i has j_i jobs and $\sum_i j_i = n$. Let d be the maximum degree in the network and let $\mathcal{L}$ be the length of the optimal schedule. We also assume that each processor knows the current number of jobs on its neighboring processors.

In one step of the algorithm, which we will refer to as EAGER-SCHEDULER, a processor with j_i jobs will pass one job to each of its neighbors with less than $j_i - 2d$ jobs. A precise statement of EAGER-SCHEDULER appears in Figure 1.

This algorithm was analyzed for the load balancing problem by Aiello *et al.* [1] and Ghosh and Muthukrishnan [2]. The difference between the load balancing problem and network scheduling is that in the load balancing problem there is no overlap of communication and computation. The objective is to balance (either perfectly or approximately, depending on the variation of the problem) the number of jobs on each machine. The bounds of [1, 2] are not in terms of an actual lower bound but instead an existential lower bound. They can prove that their algorithm balances the number of jobs on each processor so that no two neighbors are more than d apart. Let Γ be the average number of jobs in a network with edge expansion α. Their algorithm will perform this approximate balancing in $O(\Delta/\alpha)$ steps where $\Delta = \max_i(j_i - \Gamma)$ and is referred to as the *imbalance* in the network. Further, they show that there exists a network with imbalance Δ and edge expansion α that requires $O(\Delta/\alpha)$ steps to balance. Thus, their algorithm is not an approximation algorithm by the usual definition since the lower bound is not necessarily for the instance at hand but instead an existential lower bound. While their algorithms may produce a short schedule by load balancing, our algorithm gives a *guarantee* that the schedule produced is no more than an $O(\log m)$ factor longer than the shortest possible schedule for the instance given.

EAGER-SCHEDULER

if $j_i \neq 0$
then process a job, set $j_i = j_i - 1$
  for each neighbor p_k such that $j_i - j_k > 2d$ **do**
    pass a job to neighbor p_k
  update j_i accordingly
receive jobs from other neighbors, update j_i accordingly.

Figure 1: One step of EAGER-SCHEDULER for processor p_i.

1 Algorithm and Analysis

At each step of the algorithm, we can classify each processor, p_i, by the number of jobs it has compared to $\mathcal{L}$. If p_i has more than $\mathcal{L}$ jobs, we call it a *surplus* processor with surplus $s_i = j_i - \mathcal{L}$. Similarly, if p_i has fewer than $\mathcal{L}$ jobs, we call it a *deficit* processor with deficit $d_i = \mathcal{L} - j_i$. We define $\mathcal{S} = \sum s_i$ and $\mathcal{D} = \sum d_i$. Note that $\mathcal{L}$ does not change over the course of the algorithm, but j_i does.

We begin the analysis by proving some bounds on the distribution of jobs and on the amount of the surplus and deficit processors.

Lemma 1 *Let $\mathcal{L}$ be the optimal schedule length. Any connected region of p processors that has q neighbors outside of the region must have less than $(p+q)\mathcal{L}$ jobs at time 0.*

Proof: The maximum number of jobs that could be processed in $\mathcal{L}$ time steps by the region of p processors is $p\mathcal{L}$. The maximum number of jobs that can be passed out of this region and still processed by time $\mathcal{L}$ is $q(\mathcal{L}-1)$. Thus the maximum number of jobs that could have started in the connected region is $p\mathcal{L} + q\mathcal{L} - q$. □

Fact 1 *$d_i \leq \mathcal{L}$ for all i.*

Lemma 2 *$\mathcal{D} \geq \mathcal{S}$.*

Proof: Denote the number of processors with surplus by x, the number of processors with deficit by y and the number of processors with exactly $\mathcal{L}$ jobs by z. Then the total number of jobs in the system is $(x\mathcal{L} + \mathcal{S}) + (y\mathcal{L} - \mathcal{D}) + z\mathcal{L}$. We know that $x + y + z = m$ since every processor is counted exactly once. Thus, the total number of jobs in the system is $m\mathcal{L} + \mathcal{S} - \mathcal{D}$. If $\mathcal{S} > \mathcal{D}$ then the total number of jobs would be greater than $m\mathcal{L}$ which we know can't be true if the optimal schedule length is $\mathcal{L}$. □

Lemma 3 *There are no more than $m/2$ processors with $s_i \geq \mathcal{L}$.*

Proof: Assume there are more than $m/2$ processors with $s_i \geq \mathcal{L}$. This implies that $\mathcal{S} > \frac{m}{2}\mathcal{L}$. Since $d_i \leq \mathcal{L}$ for all p_i (by Fact 1) we can say that $\mathcal{D} \leq \frac{m}{2}\mathcal{L} < \mathcal{S}$. This contradicts Lemma 2. □

Informally, passing jobs to neighbors with fewer jobs is a good idea, we now formally prove that in the following lemma. This will be used to handle one case in the more complicated analysis below.

Lemma 4 *If S is the length of the schedule in which no jobs are passed and S' is the length of the schedule produced by* Eager-scheduler *then $S' \leq S$.*

Proof: Let $m(t)$ denote the maximum number of jobs on any processor on time step t. We know that when no jobs are passed $m(t+1) = m(t) - 1$, thus $S = m(0)$. We will show that $m(t+1) \leq m(t) - 1$ if Eager-scheduler is used, which will imply that $S' \leq S$.

Notice that no processor with more than $m(t) - 2d - 1$ jobs will receive any jobs by definition of the algorithm. If $m(t) \leq 2d$ then this means that no passing will occur, thus $m(t+1) = m(t) - 1$. If $m(t) > 2d$ then since all processors that have jobs process a job, and no processor can receive more than d jobs, we can see that no processor that receives a job can end up with more than $m(t) - d - 1$ jobs. Thus, the maximum number of jobs any processor will have after this step is $m(t) - 1$. □

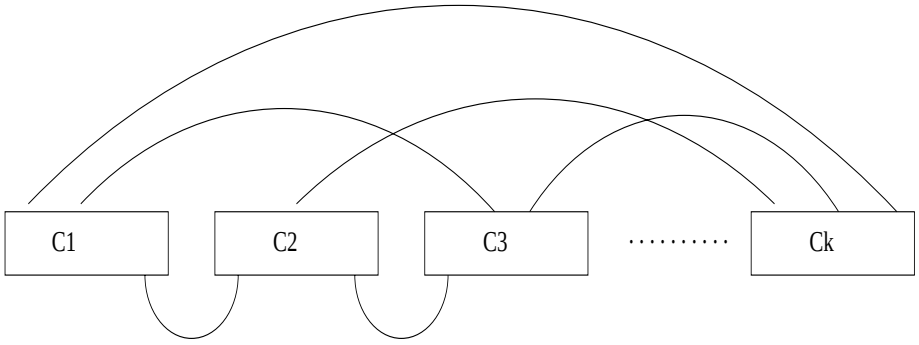

Figure 2: Example of edges in the network with respect to classes. Notice that the edges on the top will definitely have a job passed over them, while the edges on the bottom may not.

To analyze algorithm EAGER-SCHEDULER, we'll further classify each processor based on the number of jobs it has. This classification parallels one by Maggs and Leighton ([3] as communicated by Bruce Maggs) which they used for the approximate load balancing problem. In addition, the proof of Theorem 1 follows a similar structure to that of Maggs and Leighton [3] even though the details are quite different due to the difference between the two problems.

Let class C_i contain all processors, p_k, with between $\mathcal{L} + 2d(i-1) \leq s_k \leq \mathcal{L} + 2di - 1$ surplus. We'll say that all processors with $\mathcal{L} + 2di$ or more surplus are in class $C_{>i}$ and all processors with less than $\mathcal{L} + 2d(i-1)$ surplus are in $C_{<i}$. A class, C_i, is *good* if half of the neighbors of processors in $C_{>i}$ are in $C_{<i}$. A class is *bad* if it is not good. Notice that any edge that goes between processors in $C_{>i}$ and $C_{<i}$ will have a job passed across it in a time step. We use $|C_i|$ to denote the number of processors in class C_i.

We need to introduce the concept of expansion to facilitate the analysis. Let G be an m vertex graph and let S be any subset of vertices such that $|S| \leq m/2$. Finally, let the number of neighbors of vertices in S that are outside of S be denoted by $N(S)$. We say the expansion $\mu = \min_S |N(S)|/|S|$. The values of μ can range from $1/m$ to 1. Let $\beta = \frac{1}{1-(\mu/2)}$.

Lemma 5 ([3]) *At least half of the first* $2\log_\beta m$ *classes are good.*

Proof: Consider the classes by increasing index starting with C_1. Notice, all classes with index higher than one have less than $m/2$ nodes by Lemma 3. Any class, C_j, that is bad has at least $\frac{\mu}{2}|C_{>j}|$ processors since at least half the neighbors of $C_{>j}$ are in C_j. So each time we encounter a bad class the number of possible processors in subsequent classes is decreased by a factor of $1/\beta$. Since there are no more than $m/2$ processors in $C_{>1}$ we can only have $\log_\beta m$ bad classes. Hence, at least half of the first $2\log_\beta m$ classes must be good. □

Figure 2 gives an pictorial example of the classification of the processors and what edges jobs will get passed over. Now, we prove the main result of the paper.

Theorem 1 EAGER-SCHEDULER *is an* $O(\log m)$*-approximation algorithm if* $\mathcal{L} > m^2$.

Proof: Assume that EAGER-SCHEDULER runs for T steps.

Since half of the first $2\log_\beta m$ classes are good for all T time steps then at least one of those classes must be good for $T/2$ of the steps by a pigeonhole argument. Call one such class C_k. We will group the steps when C_k is good into *phases*. Denote the number of processors in

$C_{>k}$ at the start of phase i by x_i and the number of neighboring processors outside of $C_{>k}$ at the start of phase i by y_i. Recall, on the steps when C_k is good at least half of the neighbors of $C_{>k}$ are in $C_{<k}$. We'll say that phase i ends when one of the following two conditions occur.

1. The number of neighbors of processors in $C_{>k}$ that are outside of $C_{>k}$ has dropped below $y_i/2$.
2. There have been $4\mathcal{L} - 4dk/y_i$ steps completed in phase i.

If the first condition happens on phase i, then we start phase $i+1$ with y_{i+1} equal to the current number of neighbors outside of $C_{>k}$.

If the second condition happens in phase j then we claim that no processor has more than $2\mathcal{L} + 2dk$ jobs. Notice that

$$\sum_{i|p_i \in C_{>k}} s_i \leq y_j \mathcal{L}$$

at the start of phase j by Lemma 1. Furthermore, the amount of surplus over $\mathcal{L} + 2dk$ in $C_{>k}$ at the start of phase j is $y_j\mathcal{L} - 2x_j dk$. Thus, if we show that at least $y_j\mathcal{L} - 2x_j dk$ jobs are passed out of $C_{>k}$ then we can conclude that no processor is in $C_{>k}$ and therefore no processor has more than $\mathcal{L} + 2dk$ surplus.

Since $4\mathcal{L} - 4dk/y_j$ steps occur such that on each step at least $y_j/4$ jobs are passed out of $C_{>k}$ then at least

$$(y_j/4)(4\mathcal{L} - 4dk/y_j) = y_j\mathcal{L} - dk > y_j\mathcal{L} - 2x_j dk$$

jobs are passed out of the region. Since the number of jobs passed out is greater than the amount of surplus over $\mathcal{L} + 2dk$ we can conclude that there are no processors remaining in $C_{>k}$.

Notice that

$$\sum_{i|p_i \in C_{>k}} j_i - (\mathcal{L} + 2dk)$$

does not increase even though some processor can be reclassified as being in $C_{>k}$. This is because the job that caused a processor to be reclassified must have come from $C_{>k}$.

So, since each phase has no more than $4\mathcal{L} - 4dk/y_i$ steps and there are no more than $\log m$ phases, the maximum number of steps needed to get every processor below $2\mathcal{L} + 2dk$ jobs is $(4\mathcal{L} - 4dk/y_i)\log m$. Recall that we have only been counting the steps when C_k is good. Class C_k is bad for at most $T/2$ steps so the actual bound on the total number of steps is $(8\mathcal{L} - 8dk/y_i)\log m$.

Now we know it will take no longer than $2\mathcal{L} + 2dk$ to finish the remaining work by Lemma 4. Thus, the total time is

$$(8\mathcal{L} - 8dk/y_i)\log m + 2\mathcal{L} + 2dk \leq O(\log m)\mathcal{L}$$

if $\mathcal{L} > (2dk)/(\log m)$.

Recall that $k \leq 2\log_\beta m$. So, in the worst case the expansion is $1/m$, thus $k = m\log m$, and the maximum degree in the network is m then $\mathcal{L}$ needs to be at least m^2. Many networks, including random networks and uniform degree networks, have constant expansion. In these cases, $\mathcal{L}$ only needs to be larger than m. □

We note that this analysis provides a weak bound in some cases. We know from Lemma 1 and Lemma 4 that this algorithm is a $(d+1)$-approximation algorithm. Thus, for $d < \log m$ the analysis above is loose, but it does guarantee that in any network EAGER-SCHEDULER produces schedules within a $\log m$ factor of optimal given a sufficiently long optimal schedule length.

References

[1] W. Aiello, B. Awerbuch, B. Maggs, and S. Rao. Approximate load balancing on dynamic and asynchronous networks. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 632–641, 1993.

[2] B. Ghosh and S. Muthukrishnan. Dynamic load balancing on parallel and distributed networks by random matchings. In *Proceedings of the 6th ACM Symposium on Parallel Algorithms and Architectures*, pages 226–235, 1994.

[3] B. Maggs and F. T. Leighton, September 1994. Private communication.