Supplementary Discussions and Solutions to Selected Problems in: Introduction to Parallel Computing by Vipin Kumar, Ananth Grama, Anshul Gupta, & George Karypis

John Weatherwax

Chapter 8

Analysis of Parallel Depth First Search Algorithms

Following the discussion in the book on this topic I will clarify some points that I had difficulty following. Hopefully, these comments will be of help to others as they learn this material.

In the presented analysis, there is a total of W work (searching) to be done and we initially have all of it assigned to a single processor p^*. This is in fact the way most parallel programs begin so our assumptions are in line with reality. Since we have not *specifically* specified the load balancing scheme we can be guaranteed that this W amount of work will be split up and distributed to the remaining processors if the load balancing scheme is "weakly fair". Intuitively, this means that after enough requests for work in the system (from any processor) have been issued *eventually* every processor will have work requested from it. In other words, no processor is excluded from being asked to provide work.

To quantify the notion of how many requests must take place before we can guarantee that every processor has had work requested from it. We define $V(p)$ to be the number of requests that must take place in the system before *we can guarantee* that every processor has been polled for work. Obviously, $V(p)$ depends on the specific load balancing scheme used be it global round robin, randomized polling etc. We see that at least $V(p) \geq p$ since if $V(p) < p$ we have not even generated enough polling to request from every existing processor. Specifically $V(p)$ will depend on the load balancing algorithm and will be specified when we discuss the explicit load balancing schemes below. In our initial situation, with all the work W at one processor, we must have $V(p)$ requests in our system before we can be guaranteed to to have requested work from p^*. With the first request that p^* accepts, the maximal work in the system drops to $(1-\alpha)W$. This is assuming an α-splitting as discussed in the book. After $2V(p)$ requests it drops to $(1-\alpha)^2 W$, etc. After $nV(p)$ total requests it drops to $(1-\alpha)^n W$. To determine the number n of $V(p)$ requests before our tasks are so small they can no longer be split we enforce the requirement that after these n work divisions, the maximal work left in the system be less than ϵ. This gives

$$(1-\alpha)^n W < \epsilon$$

Solving for n gives

$$n > \log_{1/(1-\alpha)}(\frac{W}{\epsilon})$$

Thus after n "chunks" of $V(p)$ requests we can guarantee that the maximal work at any given node in the system is less than ϵ. Thus the total number of work requests is $O(nV(p))$ and since

$$O(n) = O(\log_{1/(1-\alpha)}(\frac{W}{\epsilon})) = O(\log W)$$

the total number of work requests simplifies to $O(V(p)\log W)$ as given in the text. If we further assume that each request for work and the correspond work transfers occupy constant time t_{comm}, an upper bound on the amount of communication overhead can be given by $T_o = t_{\text{comm}} V(p) \log W$. For variations on communication that does not take place in constant time this see the problems from this chapter.

A Different Analysis of V(p) for Random Polling

Here is a different method of attack for obtaining the same answer for this question. As stated in the text, assume that we have p total boxes to mark and we continue to make random attempts to mark an unmarked box. Define

$f(q,n)$ to be the probability that q boxes are marked after n trials. Then the average number of trials needed to mark all of the boxes is given by

$$\bar{N} = \sum_{k=p}^{\infty} k f(p,k) \tag{1}$$

A simple partial difference equation for $f(q,n)$ is given by the following logic. The probability that $q+1$ boxes are marked after $n+1$ trials can be given by the sum of the two mutually exclusive events

- $q+1$ boxes are marked after n trials and the $n+1$ attempt to mark a box *fails*
- q boxes are marked after n trials and the $n+1$ attempt to mark a box *succeeds*

Now at any stage, with q (of p) boxes marked, the chance we mark another box (a success) on the next trial is $\frac{p-q}{p}$ and the chance we do not (a failure) is given by $\frac{q}{p}$. With this background we can now express $f(q+1,n+1)$ as

$$f(q+1,n+1) = \frac{q+1}{p} f(q+1,n) + \frac{p-(q+1)}{p} f(q,n)\,. \tag{2}$$

For $q \geq 0$ and $n \geq 1$. Note that this is a linear partial difference equation and has many amenable method for its solution. For initial and boundary conditions for $f(q,n)$ we see from simple arguments that

$$\begin{aligned}
f(0,n) &= 0 \quad \text{for} \quad n \geq 1 && (3)\\
f(q,0) &= 0 \quad \text{for} \quad q \geq 1 && (4)\\
f(q,n) &= 0 \quad \text{for} \quad q > n && (5)\\
f(1,1) &= 1 && (6)\\
f(q,q) &= \frac{p^{(q)}}{p^q} = \frac{(p-1)(p-2)\dots(p-q+1)}{p^{q-1}} \quad \text{for} \quad q \geq 1 && (7)
\end{aligned}$$

We now desire the solution to the partial difference equation, Eq. 2. It will benefit us to write it decremented in q by one or as

$$f(q,n+1) = \frac{q}{p} f(q,n) + \frac{p-q}{p} f(q-1,n) \quad \text{for} \quad n \geq 1,\, q \geq 1\,. \tag{8}$$

Attempts to solve this partial difference equation are reported on below using a variety of techniques. Some were more successful that others and may indicate why the book choose the approach they took. Many reduce to the same calculations. In locations where I couldn't make more progress, I simply stopped, opting to come back to these derivations when my skills had improved more.

Operator Method

As a first attempt, lets try to solve Eq. 8 using the Operator Method. As such we define

$$\begin{aligned} E_1 f(q,n) &= f(q+1,n) \qquad (9)\\ E_2 f(q,n) &= f(q,n+1) \qquad (10)\end{aligned}$$

and our original equation becomes

$$E_2 f = \left(\frac{q}{p} - (1-\frac{q}{p})E_1^{-1} \right) f \qquad (11)$$

therefore recognizing the above as n iterations on $f(q,n)$ we have (by induction) and then the binomial theorem that

$$\begin{aligned} f(q,n) &= \left(\frac{q}{p} + (1-\frac{q}{p})E_1^{-1} \right)^n g(q) \qquad (12)\\ &= \sum_{k=0}^{n} \left(\begin{array}{c} n \\ k \end{array} \right) \left(\frac{q}{p}\right)^k \left(1-\frac{q}{p}\right)^{n-k} E_1^{-(n-k)} g(q) \qquad (13)\\ &= \sum_{k=0}^{n} \left(\begin{array}{c} n \\ k \end{array} \right) \left(\frac{q}{p}\right)^k \left(1-\frac{q}{p}\right)^{n-k} g(q-n+k) \qquad (14)\end{aligned}$$

As an interesting observation, that may shed light into how to proceed, notice that $\left(\begin{array}{c} n \\ k \end{array} \right) (\frac{q}{p})^k (1-\frac{q}{p})^{n-k}$ is the *Binomial* distribution. That is, the probability of k successes in n trials where the probability of a single success is $\frac{q}{p}$.

The Z-transform with respect to n

Taking the Z-transform with respect to n gives

$$zF(q,z) - zf(q,0) = \frac{q}{p}F(q,z) + F(q-1,z)\frac{p-q}{p} \quad \text{for} \quad q \geq 1 \qquad (15)$$

Now $f(q,0)=0$ when $q>0$ so the above can be solved for $F(q,z)$ giving

$$F(q,z)=\frac{1-\frac{q}{p}}{z-\frac{q}{p}}F(q-1,z) \quad \text{for} \quad q\geq 1 \tag{16}$$

Solving by iteration we obtain

$$F(q,z)=\frac{\prod_{l=1}^{q}(1-\frac{l}{p})}{\prod_{l=1}^{q}(z-\frac{l}{p})}F(0,z) \tag{17}$$

Note that

$$F(0,z)=Z(f(0,n))=\sum_{n\geq 0}f(0,n)z^{-n}=f(0,0)z^{0}=f(0,0) \tag{18}$$

At this point we have not determined $f(0,0)$. Limiting values of Eqs. 3 and 4 above would indicate that its value should be 0. This cannot be true, or else we end up with the trivial solution for $F(q,z)$. In addition, limiting values of the Eq. 7 require the value of $f(0,0)=1$, which is what we use in the analysis below.

To compute the inverse Z-transform of Eq. 17 we note that this expression can be separated into a sum of individual terms by partial fractions. To enable such a transformation we are looking for a set of coefficients A_l such that

$$\frac{1}{\prod_{l=1}^{q}(z-\frac{l}{p})}=\sum_{l=1}^{q}\frac{A_l}{z-\frac{l}{p}} \tag{19}$$

Multiplying both sides by

$$\prod_{l=1}^{q}\left(z-\frac{l}{p}\right)$$

in the standard way we can obtain the following for the coefficients A_l

$$A_l=\frac{1}{\prod_{m=1,m\neq l}^{q}\left(\frac{l}{p}-\frac{m}{p}\right)}=\frac{p^{q-1}}{\prod_{m=1,m\neq l}^{q}(l-m)} \quad \text{for} \quad l=1,2,\ldots,q \tag{20}$$

so that with this substitution $F(q,z)$ becomes

$$F(q,z)=p\left(\prod_{l=1}^{q}(1-\frac{l}{p})\right)\sum_{l=1}^{q}\frac{A_l}{l\left(\frac{z}{\frac{l}{p}}-1\right)} \tag{21}$$

Note that the product expression in the above equation can be written in terms of the factorial function as

$$\prod_{l=1}^{q}(1-\frac{l}{p}) = \prod_{l=1}^{q}(p-l)\frac{1}{p} \tag{22}$$

$$= \frac{1}{p^q}\prod_{l=1}^{q}(p-l) \tag{23}$$

$$= \frac{1}{p^q}\frac{p(p-1)(p-2)\dots(p-q)}{p} \tag{24}$$

$$= \frac{1}{p^{q+1}}p^{(q+1)} \tag{25}$$

so that at this point $F(q,z)$ becomes

$$F(q,z) = \frac{p^{(q+1)}}{p^q}\sum_{l=1}^{q}\frac{A_l}{l\left(\frac{z}{\frac{l}{p}}-1\right)} \tag{26}$$

To further compute the inverse Z-transform of this expression recall that the Z-transform of the Heavyside unit step (with jump occurring at 1) is given by (see Page 112 in the book by Kelly and Peterson [1]: Example 3.37)

$$\sum_{n\geq 0} u_n(1) = \frac{1}{z-1} \tag{27}$$

and recalling the geometric product formula for the Z-transform

$$Z(a^n y_n) = Y(\frac{z}{a}) \tag{28}$$

we can invert the above function obtaining

$$f(q,n) = \frac{p^{(q+1)}}{p^q}\sum_{l=1}^{q}\frac{A_l}{l}\left(\frac{l}{p}\right)^n u_n(1) \tag{29}$$

$$= \frac{p^{(q+1)}}{p^{q+n}}u_n(1)\sum_{l=1}^{q}A_l\, l^{n-1} \tag{30}$$

The Method of Generating Functions

In this method we multiply both sides of the equation by x^n and sum from $n \geq 0$. This gives

$$\sum_{n\geq 0} f(q,n+1)x^n = \sum_{n\geq 0}\frac{q}{p}f(q,n)x^n + \frac{p-q}{p}\sum_{n\geq 0} f(q-1,n)x^n \tag{31}$$

Defining $F(q,x) = \sum_{n\geq 0} f(q,n)x^n$ the above becomes

$$\sum_{n\geq 1} f(q,n)x^{n-1} = \frac{q}{p}F(q,x) + (1-\frac{q}{p})F(q-1,x) \tag{32}$$

or

$$\frac{1}{x}F(q,x) = \frac{q}{p}F(q,x) + (1-\frac{q}{p})F(q-1,x) \tag{33}$$

or

$$F(q,x) = \frac{(1-\frac{q}{p})}{\frac{1}{x}-\frac{q}{p}} \tag{34}$$

The Method of Separation of Variables

To use Separation of Variables we define $f(q,n) = F_q G_n$ and insert this into our partial difference above to get

$$F_q G_{n+1} = \frac{q}{p}F_q G_n + (1-\frac{q}{p})F_{q-1}G_n \tag{35}$$

Now dividing by $G_n F_q$ we obtain

$$\frac{G_{n+1}}{G_n} = \frac{q}{p} + \left(1-\frac{q}{p}\right)\frac{F_{q-1}}{F_q}\,. \tag{36}$$

Since the left hand side is a function *only* n and the right hand side is *only* a function of q they each must equal a separation constant we denote α. With this substitution the equation for G_n then becomes

$$G_{n+1} = \alpha G_n \tag{37}$$

with a solution of

$$G_n = G_0 \alpha^n \tag{38}$$

Similarly F_q must satisfy

$$\frac{q}{p} + (1-\frac{q}{p})\frac{F_{q-1}}{F_q} = \alpha \tag{39}$$

or

$$F_q = \frac{\frac{q}{p}-1}{\frac{q}{p}-\alpha}F_{q-1} \tag{40}$$

Derivation of a difference equation for $\bar{N}$

This may yet be another method that could yield manageable calculations.

Chapter 8 Solutions

Problem 1

To evaluate the performance of this scheme we will follow the discussion given in the book and compute the overhead due to communication i.e. resulting from work requests and work transfers necessitated by the load balancing algorithm; be it asynchronous round robin, global round robin, or random poling. From the the discussions in the book, the communication overhead is given by

$$T_0 = t_{\text{comm}} V(p) \log W \,. \tag{41}$$

Assuming a constant communication time (t_{comm}) independent of the distance between processors and the amount of data sent.

Problem 4

Part (a): Assume as in the discussion in the text that W amount of work is initially placed at a single processor. In $V(p)$ requests and assuming an α-splitting we reduce the maximum work at any given node to $(1-\alpha)W$. We also require distributing either αW or $(1-\alpha)W$ to the requesting processor (in a real implementation where communication bandwidth is limited one would want to transmit the smaller of those two expressions). This transmission requires a communication time proportional to either $\sqrt{\alpha W}$ or $\sqrt{(1-\alpha)W}$ and is bounded above by the larger of these two. Since we assume that $0 < \alpha < 0.5$ this is $\sqrt{(1-\alpha)W}$. This total negotiation for work results in at most $V(p)$ requests each with t_{comm} communication time and the additional transfer of work which is bounded above by $t_w\sqrt{(1-\alpha)W}$. Giving a total communication overhead of

$$T_o^{(1)} = t_{\text{comm}} V(p) + t_w \sqrt{(1-\alpha)W} \,. \tag{42}$$

A second application of the same logic results in

$$T_o^{(2)} = T_o^{(1)} + t_{\text{comm}} V(p) + t_w \sqrt{(1-\alpha)^2 W} \tag{43}$$

$$= \quad 2t_{\text{comm}}V(p) + t_w(\sqrt{(1-\alpha)W} + \sqrt{(1-\alpha)^2W}) \tag{44}$$

Induction to multiple blocks of work requests on the above pattern (until the work at all processors is less than the minimum split amount ϵ) gives for the total overhead due to communication (transfer of data plus requests) of

$$T_o = t_{\text{comm}}V(p)\log W + t_w\sqrt{W}\sum_{i=1}^{n}(1-\alpha)^{i/2} \tag{45}$$

Where as in the notes for this section

$$n > \log_{1/(1-\alpha)}(\frac{W}{\epsilon})\,.$$

To determine the total communication overhead we must perform the summation in Eq. 45. The simplest way to evaluate this expression is to consider the limit when n tends to infinity. In that case, the summation above is a geometric series and converges nicely to a constant. Giving in total that the total overhead is given by

$$T_0 = t_{\text{comm}}V(p)\log(W) + O(t_w\sqrt{W}) \tag{46}$$

Now Eq. 46 represents the communication required by the load balancing strategy through its dependence on $V(p)$. For load balancing by Global Round Robin (GRR) the function $V(p) = p$ since we must have *at least* p work requests in the system before we can be gaurrenteed that every processor has received at least one request and Eq. 46 becomes

$$T_0 = t_{\text{comm}}p\log(W) + C_1t_w\sqrt{W} + C_0 \tag{47}$$

For the hypercube architecture the average distance between processors is given by $\Theta(\log(p))$ so that the above becomes

$$T_0 = O(p\log(p)\log(W)) + O(t_w\sqrt{W})\,. \tag{48}$$

To derive the iso-efficiency function we ballance the communication overhead T_0 with the total work to be done W giving

$$W = O(p\log(p)\log(W)) + O(\sqrt{W}) \tag{49}$$

and we desire to solve the above asymptotically for $W = W(p)$. Asymptotically, $W \gg \sqrt{W}$, and we drop the $O(\sqrt{W})$ term from the above giving

$$W \approx p\log(p)\log(W) \quad \text{when} \quad W \to \infty$$

giving

$$\begin{aligned} W &= O(p\log(p)\log(p\log(p\log(W)))) && (50)\\ &= O(p\log(p(\log(p)+\log(\log(p))+\log(W)))) && (51)\\ &\approx O(p\log(p)^2) && (52)\end{aligned}$$

As discussed in the text for global round robin, the iso-efficiency due to contention for the shared resource "counter" processor is $O(p^2\log(p))$ and this dominates the iso-efficiency above.

Part (b): In this case, after the system has "seen" $V(p)$ work requests the maximal work at any one processor is reduced to $W(1-\alpha)$, after two blocks of $V(p)$ work requests the maximal work in any one processor is given by $W(1-\alpha)^2$. As before we have, after n blocks of $V(p)$ work requests the maximal work in any one processor is given by $W(1-\alpha)^n$. The assumptions of this part of the problem assume that the communication time for the transfer of work is upper bounded by

$$\log(W(1-\alpha)^n)\,. \qquad (53)$$

So in total, the total time required for the data transfered is bounded above by

$$\begin{aligned} \sum_{i=1}^{n} t_w \log(W(1-\alpha)^i) &= t_w \sum_{i=1}^{n} \log(W) + i\log(1-\alpha) && (54)\\ &= t_w n\log(W) + t_w\log(1-\alpha)\sum_{i=1}^{n} i && (55)\end{aligned}$$

with t_w the per word data transfer time (time required to transfer a word of data). Since $n = O(\log(W))$ as discussed above we observe that the above is bounded above by

$$t_w(\log(W))^2 + t_w\log(1-\alpha)n^2 = O(\log(W)^2)\,. \qquad (56)$$

Thus the overhead due to communication of work requests is

$$T_0 = t_{\text{comm}}V(p)\log(W) + (\log(W))^2 \qquad (57)$$

since for Global Round Robin $V(p) = p$ and we obtain

$$T_0 = t_{\text{comm}}p\log(W) + (\log(W))^2 \qquad (58)$$

For the hypercube architecture $t_{\text{comm}} = \log(p)$ and we have

$$T_0 = p\log(p)\log(W) + (\log(W))^2 \,. \tag{59}$$

To derive the iso-efficency function set $T_0 = W$ and solve for W as a function of p. The assignment of $T_0 = W$ gives

$$W = p\log(p)\log(W) + (\log(W))^2 \tag{60}$$

Since $(\log(W))^2$ is subdominant to W (the left hand side) we obtain equating W with the $\log(p)$ term the expression

$$W = p\log(p)\log(W) \tag{61}$$

which is the same result as in Part (a) and gives an iso-efficiency function of $W = O(p(\log(p))^2$ as before. Since for both of these situations the communication time due to data transfer did not affect the iso-efficiency function we see a motivation for not including it in the discussion presented in the book.

References

[1] W. G. Kelley and A. C. Peterson. *Difference Equations. An Introduction with Applications.* Academic Press, New York, 1991.