CSE638 & AMS641: Advanced Algorithms **Lecturer:** Prof. Rezaul A. Chowdhury
Submitted by: Yoni Fogel, Arun Rathakrishnan and Amitav Paul **Date:** March 12, 2013

Homework # 1

1 Parallel BFS with Work Stealing

1.a

See Graph::serial _ bfs() in bfs.cc

1.b

- In *PARALLEL-BFS*, the following race conditions may occur, but will not affect the correctness of the output: - In *PARALLEL-BFS-THREAD*, multiple cores can be on line 5 with the same vertex v. It is then possible for two (or more) threads to call line 6 at the same time for v. This does not affect correctness because all threads are writing the same value. - In *PARALLEL-BFS-THREAD*, when stealing, we lock the thief and victim, but only protecting from other thieves. There can be a race condition if the victim is fast (or MIN-STEAL-SIZE $\approx |Q^{in}.q[\text{victim}]|$, where the victim explores some vertexes that were ALSO stolen. This does not affect correctness, it only slightly impacts performance.

1.c

- One vertex can be the end point of many vertexes. So, one vertex can be multiple times in Q^{in}. In case of stealing, due to overlapping of execution thief and victim may process the same vertex. So that same vertex can exist multiple times in Q^{in}.

- So, one vertex can not be more than one time in any $Q^{in}.q[i]$ because the check on line 5 of bfs-thread (and a single queue is serialized). It may end up in different queues (see race conditions above).

- We can add an additional field *c[v]* for each vertex *v* which will keep the track of the processor by which it discovered. i.e. $\mathbf{c}[\mathbf{v}] = \mathbf{i}$ in between line 6-7 in PARALLEL-BFS-THREAD.

And before expanding a vertex we check whether the vertex is discovered by this thread or not by checking *c[v]*. i.e. we can add ***if*** $(\mathbf{c}[\mathbf{v}] = \mathbf{i})$ before ***for*** loop at line 4 in PARALLEL-BFS-THREAD. So, only the processor who have won in the race condition for vertex v has right to expand it.
Note this does not affect correctness because exactly one processor will win the race (if it happens).

- Q^{in} holds the vertices in the current BFS level. So, any vertex *v* in Q^{in} implies that *d[v]* is not infinity. Now it may happen that *v* is adjacent to any other vertex in Q^{in}. Since, *d[v]* is not infinity so it can not be added to Q^{in} in any other successive iteration by any thread.

1.d

We consider phases of p steals each.
Let $F_i = \left\lceil \frac{|q_i|}{MIN\text{-}STEAL\text{-}SIZE} \right\rceil$
The potential of $Q_i = W_i = {F_i}^2$
Let

$$S_i = \begin{cases} 1, \text{No steal attempts on this processor during this phase} \\ 0, \text{otherwise} \end{cases}$$

When at least one steal attempt happens on a victim processor, the potential of the whole system drops by at least $W_i - (\frac{W_i}{4}) - (\frac{W_i}{4}) = \frac{W_i}{2}$ (Trivially for empty queues).

$X_i = W_i S_i$ is the portion of the potential that was involved in attempted steals. The potential drop of the whole system $\geq X_i/2$.

$E[S_i] = 1 - (1 - \frac{1}{p})^p \geq (1 - \frac{1}{e}) \geq \frac{1}{2}$
Now,

$$E\left[X\right] = \sum_{i=1}^{p} E\left[X_i\right] > \left(1 - \frac{1}{e}\right) \sum_{i=1}^{p} E[W_i] > \left(1 - \frac{1}{e}\right) E\left[W\right]$$

By Markov's inequality,

$$Pr\left(X < \beta[W]\right) = Pr\left(W - X > (1-\beta)[W]\right) < \frac{E\left[[W] - X\right]}{\left(1-\beta\right)E[W]} < \frac{1}{\left(1-\beta\right)e}$$

Consider time steps, i and j such that $j > i$ and at least p steal attempts occur between time steps i (inclusive) and j (exclusive) then,

$$Pr\left(\Phi_i - \Phi_j > \frac{\Phi_i}{4}\right) > \frac{1}{4}.$$

Let each processor correspond to a bin and each steal attempt to a throw of a ball. Let Q be the set of processors which were victims of the steal attempts. Let $X_q = \phi_i(q)$ for each $q \in Q$ and 0 otherwise.Let $X = \sum\limits_{q=1}^{p} X_q$.
Setting $\beta = \frac{1}{2}$ in Lemma **??**, we get,

$$Pr\left(X < \frac{1}{2}\Phi_i\right) < \frac{2}{e} \implies Pr\left(X < \frac{1}{2}\Phi_i\right) \geq \left(1 - \frac{2}{e}\right) = \frac{1}{4}$$

That is the weight of queues of victim processors at time i exceed half the weight of entire set of processors by $\frac{1}{4}$.
From **??**, $\Phi_i - \Phi_j \geq \frac{1}{2}X$. Combining both we get,

$$Pr\left(\Phi_i - \Phi_j \geq \frac{1}{4}\Phi_i\right) \geq \frac{1}{4}$$

1.e

Using the balls and bins, the probability that any processor choose the wrong victim = $(1-\frac{1}{p})^{cp\log p} = ((1-\frac{1}{p})^p)^{clnp} = (\frac{1}{e})^{clnp} = \frac{1}{p^c}$ So, $cp\log p$ is a good choice for MAX-STEAL-ATTEMPTS w.h.p.

Moreover, in reference of *Task 1(d)*, it can be shown that $cp\log p$ is more than expected steal attempts to steal from a victim.

1.f

For initialization: $D\log p$

For stealing: $O(min\{D^2 \log \frac{\Delta}{MIN-STEAL-SIZE} \log p, D\log \frac{n}{MIN-STEAL-SIZE} \log p\} + Dp\log p\}$

For exploring: $\Delta \times MIN - STEAL - SIZE + \frac{n}{p} + \frac{m}{p}$

For synchronization: $D\log p$

Assuming MIN-STEAL-SIZE = $\Theta(1)$

So, $T_p =$

Lower Bound for size of the input graph:

1.g

Let assume a case, when a victim has less than MIN-STEAL-SIZE vertexes. Then a thief will try to steal the edges i.e. it will calculate the PREFIX-SUM of the out degrees of the vertexes and try to figure out whether it can steal(PREFIX-SUM is greater than the MIN-STEAL-SIZE) or not. In the successive steal attempts by other thieves, each one can determine whether it is going to steal or not by checking the PREFIX-SUM.

Therefore, total steal attempts (w.h.p.) = $p\log p \times (\log q_l + \log\Delta)$ [since MIN-STEAL-SIZE = $\Theta(1)$]
= $p\log p \log \Delta q_l$

1.h

We can replace ***for*** by ***parallel for*** in line 10 of *figure 2*. So, the line 10 will be

10. ***parallel for*** i=1 ***to*** p-1 ***do***

1.i

See Graph::parallel _ bfs() in bfs.cc

1.j

Input File	RT_{SBFS}	$RT_{PBFS^{(f)}}$	$RT_{PBFS^{(g)}}$	$SF_{PBFS^{(f)}}$	$SF_{PBFS^{(g)}}$
cage15	1	2	3	4	5
cage14	1	2	3	4	5
freescale	1	2	3	4	5
Wikipedia	1	2	3	4	5
kkt-power	1	2	3	4	5
RMAT100M	1	2	3	4	5
RMAT1B	1	2	3	4	5

Table 1: RT = Running Time, SBFS = Serial BFS, $PBFS^{(f)}$ = Parallel BFS for task 1(f), $PBFS^{(g)}$ = Parallel BFS for task 1(g) and SF = Speed Factor

1.k

Answer:

2 Lockfree Parallel BFS

2.a

Answer:

2.b

Answer: