CS60040 Parallel and Distributed Algorithms, Autumn 2017–2018

Class Test

15–November–2017 | CSE 107/108, 11:00–12:00 hours | Maximum marks: 20

Roll no: _____ **Name:** _____

[*Write your answers in the question paper itself. Be brief and precise. Answer all questions.*
If you use any algorithm/result/formula covered in the class, just mention it, do not elaborate.]

1. In this exercise, you work out a parallel implementation of the quick-sort algorithm on a CREW PRAM. Let $A = (a_1, a_2, \ldots, a_n)$ be an array of n integers, that we want to sort. We choose a_1 as the pivot for partitioning.

(a) Propose an $O(\log n)$-time parallel partitioning algorithm. **(10)**

Solution

1. Set the pivot: $p = a_1$. /* $O(1)$ time */
2. For $i = 2, 3, 4, \ldots, n$ pardo: /* $O(1)$ time */
   (a) If $a_i \leqslant p$, set $LIDX[i] = 1$ and $RIDX[i] = 0$,
   (b) else set $LIDX[i] = 0$ and $RIDX[i] = 1$.
3. Compute the prefix sums of $LIDX[1 \ldots n]$ in the same array. /* $O(\log n)$ time */
4. Compute the prefix sums of $RIDX[1 \ldots n]$ in the same array. /* $O(\log n)$ time */
5. Set $n_1 = LIDX[n]$ and $n_2 = RIDX[n]$. /* $O(1)$ time */
6. For $i = 2, 3, 4, \ldots, n$ pardo: /* $O(1)$ time */
   (a) If $a_i \leqslant p$, set $L[LIDX[i]] = a_i$,
   (b) else set $R[RIDX[i]] = a_i$.
7. For $i = 1, 2, \ldots, n_1$, pardo: Copy $L[i]$ to $A[i]$. /* $O(1)$ time */
8. Set $A[n_1 + 1] = p$. /* $O(1)$ time */
9. For $i = 1, 2, \ldots, n_2$, pardo: Copy $R[i]$ to $A[n_1 + 1 + i]$. /* $O(1)$ time */

(b) How do you handle the recursive calls? **(2)**

Solution Run quick sort on $A[1 \ldots n_1]$ and $A[n_1+2 \ldots n_1+n_2+1]$ **in parallel**.

(c) Deduce the best-case and worst-case running times of this parallel quick-sort algorithm. **(4)**

Solution Best-case: $T(n) \approx T(n/2) + \mathrm{O}(\log n) = \mathrm{O}(\log^2 n)$.
Worst-case: $T(n) = T(n-1) + \mathrm{O}(\log n) = \mathrm{O}(n \log n)$.

(d) Propose a strategy to make the parallel quick sort run in worst-case $\mathrm{O}(\log^2 n \log\log n)$ time. **(4)**

Solution We can solve the selection problem in A in $\mathrm{O}(\log n \log\log n)$ time. Thus, we can choose the median as the pivot. This gives $T(n) \approx T(n/2) + \mathrm{O}(\log n \log\log n) = \mathrm{O}(\log^2 n \log\log n)$.

Note: Using only one iteration of the size-reduction loop in the selection algorithm, we can compute the median of medians in $\mathrm{O}(\log n)$ time. If we use this element as the pivot, we have $T(n) \leqslant T(3n/4) + \mathrm{O}(\log n)$. Although this is poorer than the best case, we still achieve $T(n) = \mathrm{O}(\log^2 n)$, that is, we can force the parallel quick sort to run in worst-case $\mathrm{O}(\log^2 n)$ time.