ICS 643: Advanced Parallel Algorithms	Fall 2016

Lecture 7 — September 26, 2016

Prof. Nodari Sitchinava *Scribe: Kyle Berney*

1 Overview

In the last lecture, we introduced prefix sums and examined a parallel algorithm for it with $O(\log n)$ runtime and $O(n \log n)$ work. Although we used the addition operator (+), prefix sums can be applied to any associative operator or function.

In this lecture, we will examine the work-optimal prefix sum algorithm. For the scope of this lecture, we will define $\oplus$ to be any associative operator and I to be the identity of this operator, i.e. $a \oplus I = a$ for all a. For convenience, we will also use the summation notation ($\sum$) to denote the application of $\oplus$ to multiple arguments.

2 Work-Optimal Prefix Sums

2.1 Recursive Version

Algorithm 1 PREFIXSUM($A[0, ..., n-1]$)

1: **if** $n == 1$ **then**
2: $\quad A[0] = A[0] \oplus I$
3: **else**
4: $\quad$ **for** $i = 0$ to $\frac{n}{2} - 1$ **in parallel do**
5: $\quad\quad B[i] = A[2i] \oplus A[2i+1]$
6: $\quad$ **end for**
7: $\quad$ PREFIXSUM($B[0, ..., \frac{n}{2} - 1]$)
8: $\quad$ **for** $i = 0$ to $n - 1$ **in parallel do**
9: $\quad\quad$ **if** $i == 0$ **then**
10: $\quad\quad\quad A[0] = A[0] \oplus I$
11: $\quad\quad$ **else if** $i \% 2 == 0$ **then**
12: $\quad\quad\quad A[i] = B[\frac{i}{2} - 1] \oplus A[i]$
13: $\quad\quad$ **else**
14: $\quad\quad\quad A[i] = B[\frac{i-1}{2}]$
15: $\quad\quad$ **end if**
16: $\quad$ **end for**
17: **end if**

Claim 1. *After line 7 executes, for $i \in \{0, ..., \frac{n}{2} - 1\}$,*

$$B[i] = \sum_{j=0}^{2i+1} A[j].$$

Proof. **Base Case:** $i = 0$. We have that:

$$\begin{aligned} B[0] &= A[2 \cdot 0] \oplus A[2 \cdot 0 + 1] \\ &= A[0] \oplus A[1] \\ &= \sum_{j=0}^{1} A[j] = \sum_{j=0}^{2 \cdot 0 + 1} A[j]. \end{aligned}$$

Inductive Step: Assume that our hypothesis is true for some $i > 0$, we will show that our hypothesis holds for $i' = i + 1$. We have that:

$$\begin{aligned} B[i'] &= B[i' - 1] \oplus B[i'] \\ &= B[i] \oplus B[i + 1] \\ &= \sum_{j=0}^{2i+1} A[j] \oplus (A[2(i+1)] \oplus A[2(i+1) + 1]) \\ &= \sum_{j=0}^{2i+1} A[j] \oplus (A[2i + 2] \oplus A[2i + 3]) \\ &= \sum_{j=0}^{2i+3} A[j] = \sum_{j=0}^{2(i+1)+1} A[j] = \sum_{j=0}^{2i'+1} A[j]. \end{aligned}$$

□

Hence, after line 7 executes, $B[i]$ will contain the prefix sum of all the first $2i + 1$ entries in the array A. In other words, B contains the prefix sum of all the odd indices in A. From this, we can deduce the prefix sum for the even indices by performing a $\oplus$ operation on the previous element's prefix sum and itself. The final for loop, starting on line 8, does exactly what we have described. Thus, the above algorithm correctly computes the prefix sum of the array A.

The runtime of this algorithm is given by the recurrence relation:

$$T(n) = \begin{cases} T(\frac{n}{2}) + O(1) & \text{if } n > 1 \\ O(1) & \text{otherwise} \end{cases} = O(\log n),$$

using the Master Theorem case 2, with $k = 0$. The recurrence relation for work is:

$$W(n) = \begin{cases} W(\frac{n}{2}) + O(n) & \text{if } n > 1 \\ O(1) & \text{otherwise} \end{cases} = O(n),$$

using the Master Theorem, case 3. Therefore, since our work is equal to the work of the best sequential algorithm, we have a work-optimal algorithm for prefix sum.

2.2 Iterative Version

Algorithm 2 PREFIXSUM($A[0, ..., n-1]$)

1: **for** $i = 0$ to $n-1$ **in parallel do**
2: $\quad B[0][i] = A[i]$
3: **end for**
4: **for** $h = 1$ to $\log n$ **do**
5: $\quad$ **for** $i = 0$ to $\frac{n}{2^h} - 1$ **in parallel do**
6: $\quad\quad B[h][i] = B[h-1][2i] \oplus B[h-1][2i+1]$
7: $\quad$ **end for**
8: **end for**
9: $C[\log n][0] = I$
10: **for** $h = \log n - 1$ down to 0 **do**
11: $\quad$ **for** $i = 0$ to $\frac{n}{2^h} - 1$ **in parallel do**
12: $\quad\quad$ **if** i % $2 == 0$ **then**
13: $\quad\quad\quad C[h][i] = C[h+1][i/2]$
14: $\quad\quad$ **else**
15: $\quad\quad\quad C[h][i] = C[h+1][\frac{i-1}{2}] \oplus B[h][i-1]$
16: $\quad\quad$ **end if**
17: $\quad$ **end for**
18: **end for**
19: **for** $i = 0$ to $n-1$ **in parallel do**
20: $\quad A[i] = A[i] \oplus C[0, i]$
21: **end for**

We can see that the for loop on line 1 has runtime $O(1)$, the for loop on line 4 has runtime $O(\log n)$, the for loop on line 10 has runtime $O(\log n)$, and the last for loop on line 19 has runtime $O(1)$. Hence, the runtime for the whole algorithm is $O(\log n)$. Similarly, the for loop on line 1 has work $O(n)$, the for loop on line 4 has work $O(n)$, the for loop on line 10 has work $O(n)$ (since, $\sum_{h=1}^{\log n} \sum_{i=1}^{\frac{n}{2^h}} 1 = \sum_{h=1}^{\log n} \frac{n}{2^h} = n(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + ... + \frac{1}{n}) \leq n$), and the last for loop on line 19 has work $O(n)$. Thus, the work for the whole algorithm is $O(n)$. In the next lecture, we will prove its correctness.