EE 382V: Parallel Algorithms **Summer 2017**

Lecture 5: Parallel Prefix

Lecturer: Vijay Garg *Scribe: Van Quach*

5.1 Parallel Prefix

Parallel prefix is the scan operation, which takes an associated binary operator, and a set of n elements and returns a . Let exam an example of a parallel prefix. Given an array A:

A	2	11	5	8
$PrefixSum$	2	13	18	26

Algorithm 1 Prefix sum - sequential version

```
INPUT: A_k, ∀k = 0...n−1
OUTPUT: C_k = Σ_{i=0}^{k} A_i, ∀k = 0...n−1
procedure PREFIX_SUM()
    for i ← 0, n−1 do
        C[i] = A[i]
    end for
    for i ← 0, n−1 do
        C[i] = C[i−1] + A[i]
    end for
end procedure
```

We can illutrate sequential version of the algorithm by a tree as below:

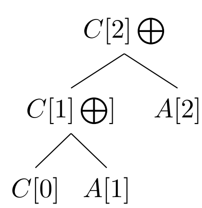

Figure 5.1: Tree graph for sequential version of prefix sum

	$T(n)$	$W(n)$
Sequential	$\mathcal{O}(n)$	$\mathcal{O}(n)$

Table 5.1: Time and Work for sequential version

We can improve the sequential version of prefix by a parallel recursive version (see professor's note). Parallel version of prefix sum:

Algorithm 2 Prefix sum - parallel version

```
procedure PREFIX_SUM()
    for i ← 0, n − 1 do in parallel
        C[i] = A[i]
    end for
    for d ← 1, n − 1 by d × 2 do
        for i ← 1, n − 1 do in parallel
            if i − d > 0 then
                C[i] = C[i] + C[i − d]
            end if
        end for
    end for
end procedure
```

$Index$	0	1	2	3
A	2	11	5	8
C	2	11	5	8
$C, d = 1$	2	13	16	13
$C, d = 2$	2	13	18	26

Figure 5.2: C's values for each step

	$T(n)$	$W(n)$
$Sequential$	$\mathcal{O}(\log(n))$	$\mathcal{O}(n \times \log(n))$

Table 5.2: Time and Work for parallel version

We can improve the parallel version to obtain work optimal algorthim by cascading technique. In cascading techinque, we can break input array into $\frac{n}{\log(n)}$ segments of size $\mathcal{O}(\log(n))$. We save this implementation for homework