Prefix Sum

1 Introduction

The prefix sum problem, also known as the scan problem, is a fundamental operation in parallel computing. It involves computing all the partial sums of a sequence of numbers. The input will be n numbers:

$$x_0, x_1, x_2, ..., x_{n-1} \tag{1}$$

And the output will be:

$$S_0, S_1, S_2, ..., S_{n-1} \tag{2}$$

The relation could be expressed as:

$$S_i = \sum_{j=0}^{i} x_j \tag{3}$$

2 Sequential Algorithm

The best sequential algorithm we can get is:

Algorithm 1 Sequential algorithm

```
S_0 = x_0
for i = 1 to n − 1 do
    S_i = S_{i−1} + x_i
end for
```

The runtime of this algorithm will be:

$$T(n, 1) = \Theta(n) \tag{4}$$

3 Parallel Algorithm

3.1 Alg-0

Assume the problem size as n, the number of processors is p. In this algorithm, we want to calculate each S_i at the same time, which means:

1. $S_0 = x_0$, use one processor
2. $S_1 = x_0 + x_1$, use two processors, each processor store one value. Notice that these two processors are not including the processor for S_0, they are new.
3. $S_2 = x_0 + x_1 + x_2$, use three processors.
4. $S_3 = x_0 + x_1 + x_2 + x_3$, use four processors.
5. And so on...

Therefore, the number of processors required:

$$p = 1 + 2 + 3 + ... + n = \frac{n(n+1)}{2} \tag{5}$$

Now if we want to use n processors to add up n numbers in parallel, the runtime is $\Theta(\log n)$. For example, for S_3 calculation:

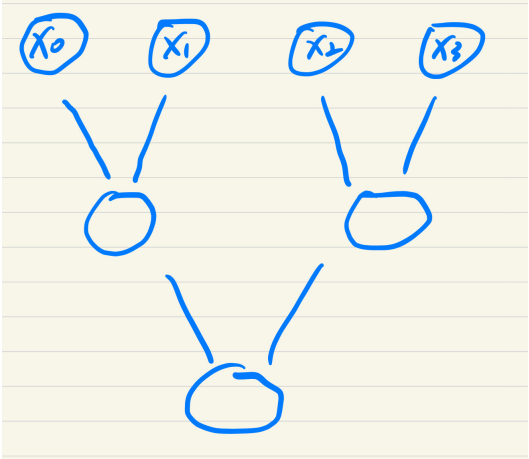

Figure 1: Alg-0

So the number of operations keeps decreasing by order of 2, so the runtime will be $\Theta(\log n)$.

Therefore, we have the following relationships:

$$T(n, \frac{n(n+1)}{2}) = \Theta(\log n) \tag{6}$$

$$T(n, \Theta(n^2)) = \Theta(\log n) \tag{7}$$

$$T(n, p) = \Theta(\frac{n^2 \log n}{p}) \tag{8}$$

3.2 Alg-1

In this algorithm, we use divide-and-conquer method to develop new algorithm. The mechanism is shown below:

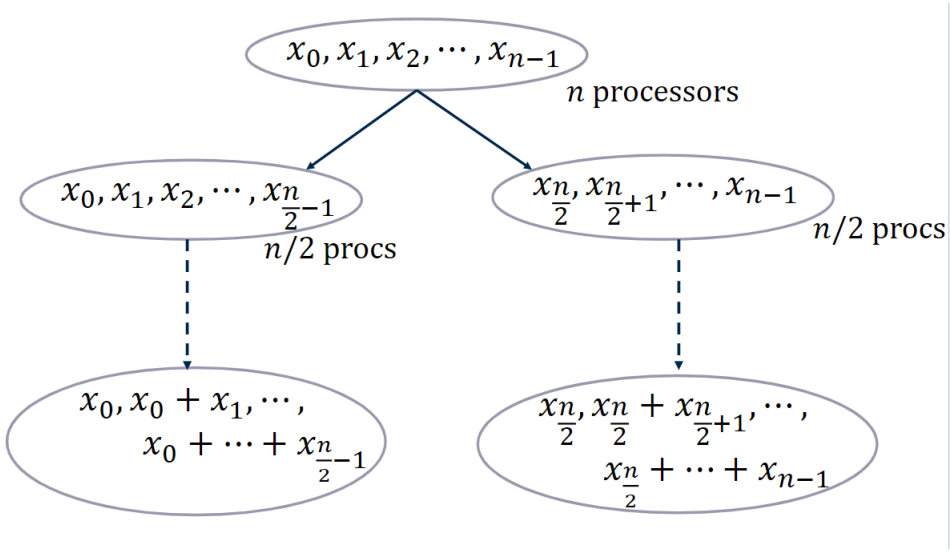

Figure 2: Alg-1

Notice here the problem size n keeps being divided by 2, until it reaches every processor has only 1 value (which means it only needs one operation), with the depth as $\log n$. The key part is merging $S_{\frac{n}{2}-1}$ on the left with all the processors on the right. Notice that this step is not just 1 operation, because we could not communicate one processor with all the processors on the right at the same time, if in serial the time will be $\Theta(n)$. However, we can use **tree-based communication**, one processor passes the value to another, than those two processors pass the value to another 4 and the merge step runtime is $\Theta(\log n)$. Therefore the total runtime would be:

$$\text{depth} * \text{time for each level} = \log n \cdot \Theta(1 + \log n) = \Theta(\log^2 n) \tag{9}$$

3.3 Alg-2

The mechanism is explained here:

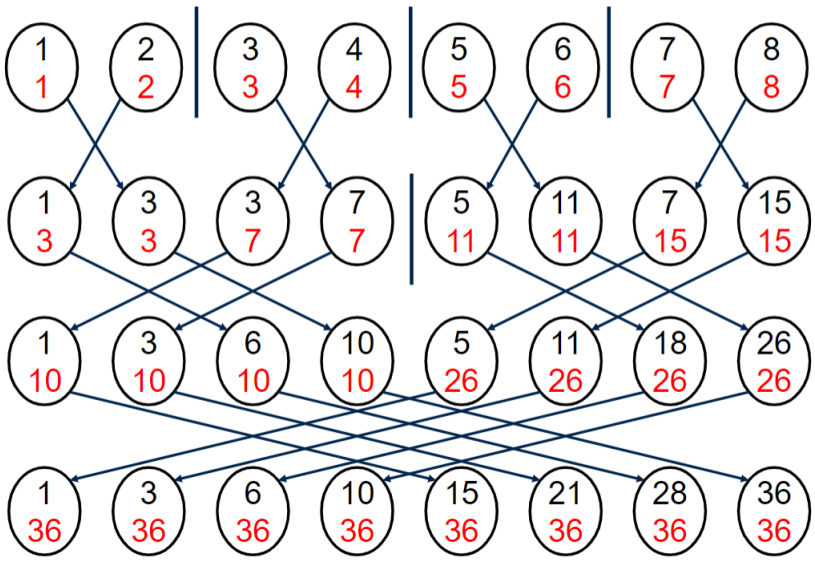

Figure 3: Alg-2

```
Algorithm (for P_i)
total_sum ← prefix_sum ← local_number
for j=0 to d-1 do
   rank' ← rank XOR 2^j
   send total_sum to rank'
   receive received_sum from rank'
   total_sum ← total_sum + received_sum
   if (rank > rank')
      prefix_sum ← prefix_sum + received_sum
endfor
```

Figure 4: Alg-2-algorithm

In words, at each level, the left hand side (**this keeps changing!**) processor will save the current local value and the result of the sum, the right hand side processor will change the local value to the sum. Also at each level, use hypercubic permutation to determine the combination. Because in this algorithm, at each level the operation is just constant (each processor send one and receive one), so the runtime will fully depend on the depth, which is $\log n$, Therefore, the runtime will be:

$$T(n, n) = \Theta(\log n) \tag{10}$$

If the number of processor is less than n, then we need to reassign the values:

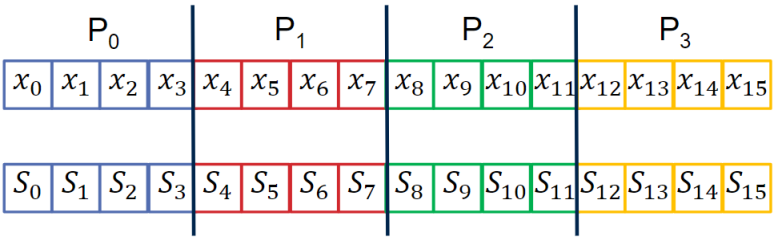

Figure 5: Alg-2, less processors

Recall the Brent's Lemma, we have:

$$T(n,p) = \Theta(\frac{n}{p}\log n) \tag{11}$$

3.4 Alg-3

This algorithm includes steps:

1. Compute prefix sum locally on each processor
2. Perform parallel prefix sum (Alg-2) using the local prefix sum on each processor
3. Add the result of parallel prefix sum on a processor to each of its local prefix sum

3.4.1 First processor prefix sum as non-zero

The mechanism is explained below:

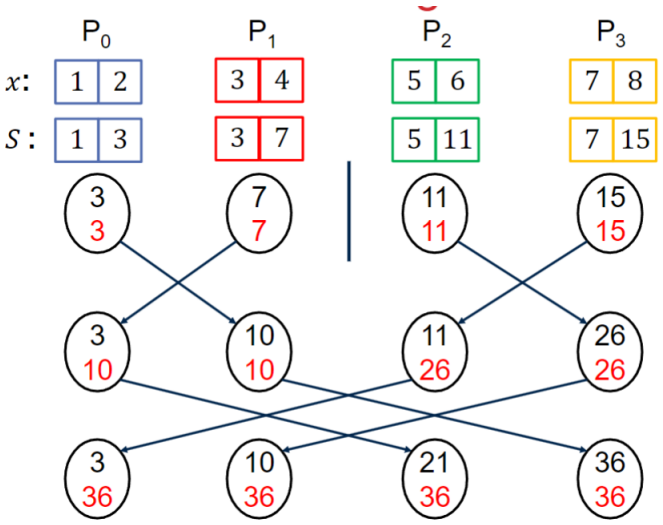

Figure 6: Alg-3, Non-zero Start

Inside the processor, we still use divide-and-conquer method, then we can get the prefix sum after adding each value inside the processor. But only the last prefix sum will be recorded for the next step. Then we basically still follow the conquer method. Here the saved value inside each processor is the value after the calculation inside the processor.

3.4.2 First processor prefix sum as zero

The mechanism is explained below:

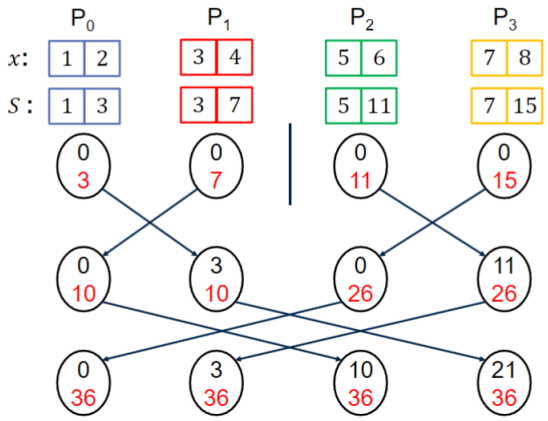

Figure 7: Alg-3, Zero Start

Basically is the same thing as the previous one, but now the processor saves the value before the calculation inside the processor.

3.4.3 Special Cases

If n i s not divisible by p, then we need to assign $\lceil \frac{n}{p} \rceil$ to each processor. If p is not power of 2, then we need to find:

$$p' = \text{a power of 2} \tag{12}$$

And:

$$\frac{p'}{2} < p < p' \tag{13}$$

Then run the code like if we have p' processors, ignore communications to/from non-existing processors:

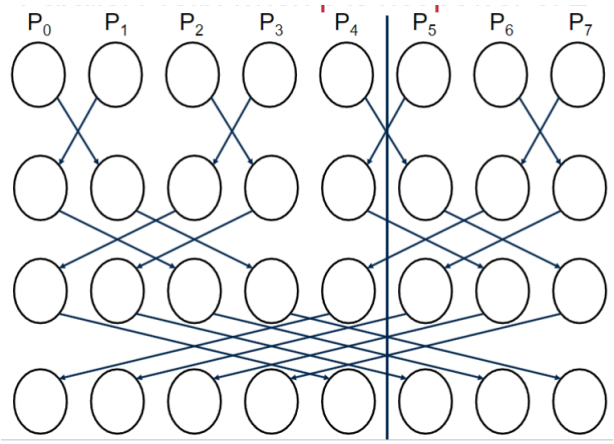

Figure 8: Special Case, the vertical line will be the cutoff line