To: Christina Taylor
From: Eli Case
Date: March, 10, 2023
Subject: CAAM 420/520 – Homework 3

Problem 1 (a) Nodes at a given level in the function tree can be processed at the same time. For example, we can begin traversing the function tree in Figure 1 at level 0. Once we reach the child nodes of level 0 at level 1, we can process the child nodes of level 1 simultaneously, as processing each child node is an independent operation.

(b) In evaluating the function tree, synchronization is required after the subtree for a given operator node has been processed. For example, Before evaluating the multiplication node at level 1, we must evaluate the subtree for the addition node at level 2 fist, for the function tree given in Figure 1.

(d) To find the parallel time complexity of evaluating the sun function using a function tree, we can make start by making the following observations.

i. If the number of nodes, n, in a given layer, L, is less than the number of total threads, N_T, then the time complexity to process that layer is $\mathcal{O}(1)$.

ii. If the number of nodes, n, in a given layer, L, is larger than or equal to the number of total threads, N_T, then the time complexity to process that layer is $\mathcal{O}(\frac{2^L}{N_T})$.

Now, given that the number of threads is a power of 2, as $N_T = 2^T$, and that $L = T$, when the number of nodes in a given layer equal the number of threads, we have the following equation for the time complexity of processing a given layer in parallel, `time_layer`, in the function tree.

$$\texttt{time\_layer} = \begin{cases} \mathcal{O}(1) & L < T \\ \mathcal{O}(\frac{2^L}{N_T}) & L \geq T \end{cases} \tag{1}$$

Subsequently, we can compute the total time complexity of processing the function tree, `time_tree`, by adding the time complexity for each layer, as given in Equation 1.

$$\texttt{time\_tree} = \mathcal{O}\left(\sum_{i=0}^{T-1} 1 + \frac{1}{N_T} \sum_{i=T}^{L_{\max}} 2^i \right) \tag{2}$$

We can rewrite Equation 2, in terms of the variables of interest, N_T and n, with the following formulas.

$$\begin{aligned} T &= log_2(N_T) \\ n &= 2m - 1 \\ L_{max} &= log_2(m) \end{aligned}$$

Rewriting Equation 2 and simplifying, we have the following equation for `time_tree`.

$$\texttt{time\_tree} = \mathcal{O}\left(log_2(N_T) + \frac{1}{N_T} \sum_{i=log_2(N_T)}^{log_2(\frac{n+1}{2})} 2^i \right) \tag{3}$$

Now, we can simplify Equation 3 by rewriting the finite sum of the geometric series. We have the following formula for the sum of a finite geometric series, S_n given by the following formula.

$$S_n = \sum_{i=0}^{n-1} r^i = \frac{r^n - 1}{r - 1}, \quad r > 1 \tag{4}$$

We would like to rewrite the following finite sum geometric of a geometric series.

$$\sum_{i=log_2(N_T)}^{log_2(\frac{n+1}{2})} 2^i \tag{5}$$

In order to apply Equation 4, we split the finite sum of a geometric series given in Equation 5 as follows.

$$\sum_{i=0}^{log_2(\frac{n+1}{2})} 2^i - \sum_{i=0}^{log_2(N_T)-1} 2^i \tag{6}$$

Now, using Equation 4, we rewrite Equation 6.

$$\frac{2^{log_2(\frac{n+1}{2})+1} - 1}{2-1} - \frac{2^{log_2(N_T)} - 1}{2-1}$$
$$2^{log_2(\frac{n+1}{2})+log_2(2)} - 2^{log_2(N_T)}$$
$$2^{log_2(n+1)} - 2^{log_2(N_T)}$$
$$n + 1 - N_T$$

Thus, we can now simplify Equation 3.

$$\texttt{time\_tree} = \mathcal{O}\bigg(log_2(N_T) + \frac{1}{N_T}\big(n + 1 - N_T\big)\bigg)$$
$$\texttt{time\_tree} = \mathcal{O}\bigg(log_2(N_T) + \frac{n+1}{N_T} - 1\bigg)$$

Finally, we obtain the time complexity for processing the function tree in parallel, as shown below.

$$\mathcal{O}\bigg(log_2(N_T) + \frac{n+1}{N_T} - 1\bigg) \tag{7}$$

(e) We were given that the time complexity for computing the sum with a parallel for loop was as follows.

$$\mathcal{O}\bigg(\frac{m}{N_T}\bigg)$$

Rewriting the above formula in terms of n, we obtain the following time complexity.

$$\mathcal{O}\bigg(\frac{n+1}{2N_T}\bigg) \tag{8}$$

We can determine when the parallel for loop implementation is slower than the function tree implementation in parallel by solving the following inequality.

$$\frac{n+1}{2N_T} > log_2(N_T) + \frac{n+1}{N_T} - 1 \tag{9}$$

Now, we simplify the inequality given in Equation 9 as follows.

$$log_2(N_T) + \frac{n+1}{2N_T} < 1 \tag{10}$$

If we consider the smallest function tree, $n = 3$, and only one thread, $N_T = 1$, it can be seen that inequality 10 is not satisfied. As a result, the function tree implementation of evaluating the sum function is always slower than the parallel for loop implementation, in both serial and parallel.