1 Overview

In the last lecture, we started introducing the various models of parallel computation. We first covered the shared-memory model, where each processor is connected to a common global memory. There are multiple forms of the shared-memory model, which are distinguished mainly by how it handles the situation where 2 or more processors are all attempting to either read or write to the same memory location. The next model we covered was the local-memory model, where each processor has its own private memory. Each processor is connected to an interconnection network, to allow each processor to communicate with each other. We then looked at the modular-memory model, where each processor is connected to an interconnection network, which is then connected to $k$-memory modules. All processors can access each of the $k$-memory modules. Lastly, we looked at mixed models, such as the parallel external memory (PEM) model and the GPU architecture. We finished up the last lecture by examining the different types of interconnection networks. Some examples are: ethernet, ring, and 2D mesh.

In this lecture, we will first introduce the circuit model. Then, we will look at how the circuit model is related to parallel models.

2 Circuit Model

The circuit model is comprised of gates; each gate has a constant fan-in and executes a primitive operation. Some examples of primitive operations of a circuit are addition, subtraction, multiplication, and boolean operations. Each gate works in parallel and starts its execution once all of its inputs are ready. Hence, we can combine gates together to compute [BM04]. The formal definition of a circuit is as follows:

**Definition 1.** A circuit for a particular problem, is a family of directed acyclic graphs (DAG) where each node is a primitive operation and each edge is a dependency between operations. [JaJ92]

Therefore, in the circuit model, the runtime is equal to the depth of the circuit, which is equivalent to the longest directed path in the DAG; and the work is equal to the size of the circuit, which is equivalent to the number of nodes in the DAG [BM04]. Notice that at most there will be $kn$ edges in the DAG, where $n =$ the number of nodes in the DAG and $k =$ fan-in of the DAG. Thus, if $k = O(1)$, number of edges in the DAG is $\Theta($number of nodes in the DAG$)$.
2.1 Relation to Parallel Models

**Theorem 2.** A circuit for a particular problem with time, \( T(n) \), and work, \( W(n) \), can be simulated on a CREW PRAM with \( p \) processors in time:

\[
t(n) \leq \frac{W(n)}{p} + T(n).
\]

**Proof.** We will consider the circuit level-by-level; let us define:

\[
\text{LEVEL}(g_i) = \begin{cases} 
1 & \text{if } g_i \text{ is fed by inputs} \\
1 + \max_{g_j \text{ feeds } g_i} \text{LEVEL}(g_j) & \text{otherwise}
\end{cases}
\]

where \( g_i \) is a gate. Let \( n_l \) be the number of gates at level \( l \). It will take \( \lceil \frac{n_l}{p} \rceil \) time to simulate level \( l \) using a \( p \) processor CREW PRAM (note: we need concurrent reads in order to handle cases where two or more gates all use a common input). By definition, we know that the number of levels in the circuit is \( T(n) \), hence we have:

\[
t(n) = \sum_{l=1}^{T(n)} \left\lceil \frac{n_l}{p} \right\rceil \leq \sum_{l=1}^{T(n)} \left( \frac{n_l}{p} + 1 \right) = \frac{1}{p} \sum_{l=1}^{T(n)} n_l + T(n).
\]

By definition, we have that \( \sum_{l=1}^{T(n)} n_l = W(n) \), therefore:

\[
t(n) \leq \frac{1}{p} \sum_{l=1}^{T(n)} n_l + T(n) = \frac{W(n)}{p} + T(n)
\]

Hence, by Brent’s Theorem, if we can create a circuit to solve a problem, then we can always simulate the circuit using a \( p \) processor CREW PRAM algorithm with \( O \left( \frac{W(n)}{p} + T(n) \right) \) runtime [BM04].

**Lemma 3.** Let \( A \) be the fastest sequential algorithm for some problem; and let \( T_A(n) \) be its runtime. Then,

\[
T_A(n) \leq W_c(n)
\]

where \( W_c(n) \) is the number of nodes in any circuit that solves the same problem.

**Proof.** We will prove by contradiction. Assume that there exists a circuit with size \( W_c(n) \), such that \( W_c(n) < T_A(n) \). By Brent’s Theorem, for \( p = 1 \) we have:

\[
T(n) \leq W_c(n) + T_c(n)
\]

\[
\Rightarrow T(n) \leq W_c(n) < T_A(n), \text{ since } W_c(n) < T_A(n).
\]

We arrived at contradiction, because \( A \) is the fastest sequential algorithm and, consequently, it must be that \( T_A(n) \leq T(n) \).
Therefore, we can never have a circuit smaller than the fastest sequential algorithm runtime.

We know that given a circuit, we can create a CREW PRAM algorithm by converting each edge into a memory read and each gate into an operation. Similarly, a \( p \)-processor CREW PRAM algorithm with work \( W(n) \) and time \( T(n) \) can be converted into a circuit with \( W(n) \) gates and depth \( T(n) \) by converting each memory read to an incoming edge, each memory write to an outgoing edge, and each operation into a gate.

3 Conclusion

In conclusion, we now know that given a \( p \) processor CREW PRAM algorithm with work \( W(n) \) and time \( T(n) \) can be converted into a CREW PRAM algorithm with \( p' < p \) processors that has work \( W(n) \) and time \( O\left(\frac{W(n)}{p'} + T(n)\right) \). In other words, if we design a CREW PRAM algorithm with \( p \) processors, then we can always scale the algorithm down for \( p' < p \) processors. Therefore, when designing parallel algorithms, we want to design it for the largest possible \( p \).

References