GPU Teaching Kit

Accelerated Computing

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Module 9 – Parallel Computation Patterns (Reduction)

Lecture 9.1 - Parallel Reduction

Objective

- To learn the parallel reduction pattern
  - An important class of parallel computation
  - Work efficiency analysis
  - Resource efficiency analysis

“Partition and Summarize”

- A commonly used strategy for processing large input data sets
  - There is no required order of processing elements in a data set (associative and commutative)
  - Partition the data set into smaller chunks
  - Have each thread to process a chunk
  - Use a reduction tree to summarize the results from each chunk into the final answer
- E.G., Google and Hadoop MapReduce frameworks support this strategy
- We will focus on the reduction tree step for now

Reduction enables other techniques

- Reduction is also needed to clean up after some commonly used parallelizing transformations
- Privatization
  - Multiple threads write into an output location
  - Replicate the output location so that each thread has a private output location (privatization)
  - Use a reduction tree to combine the values of private locations into the original output location

What is a reduction computation?

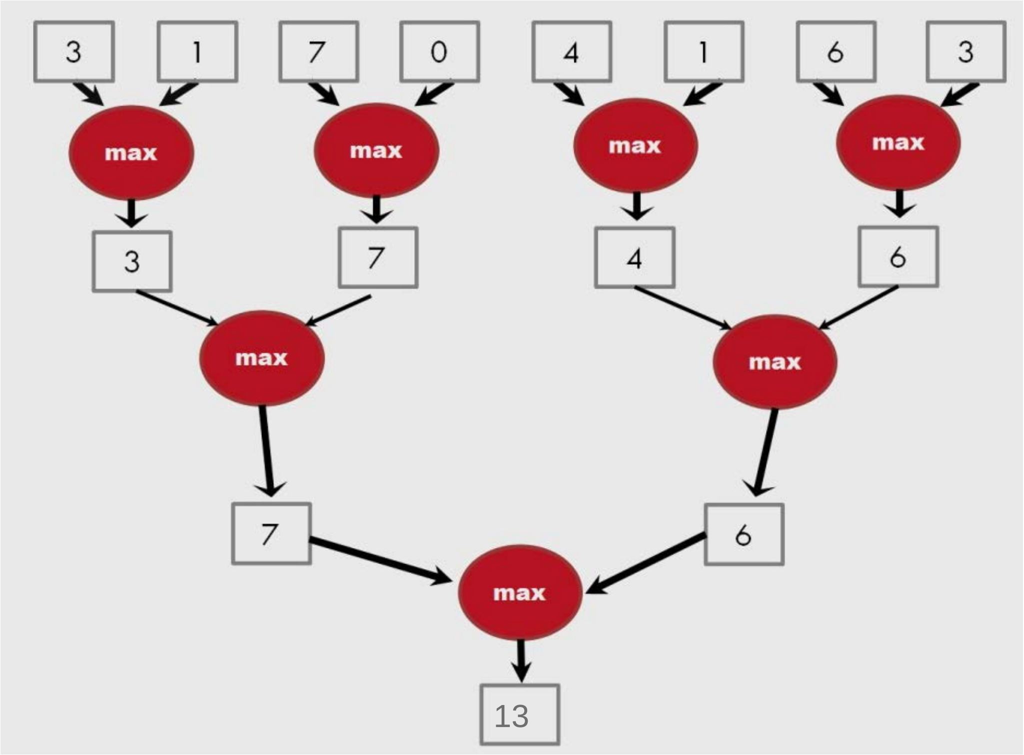
- Summarize a set of input values into one value using a “reduction operation”
  - Max
  - Min
  - Sum
  - Product
- Often used with a user defined reduction operation function as long as the operation
  - Is associative and commutative
  - Has a well-defined identity value (e.g., 0 for sum)
  - For example, the user may supply a custom “max” function for 3D coordinate data sets where the magnitude for the each coordinate data tuple is the distance from the origin.

An example of “collective operation”

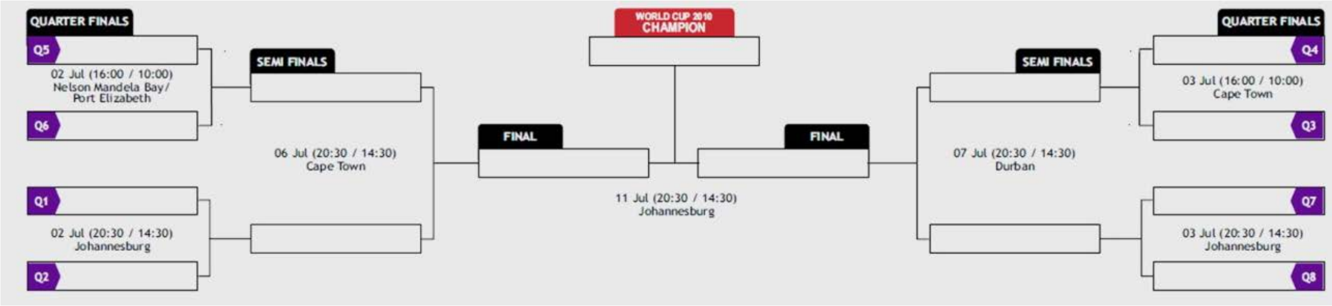
An Efficient Sequential Reduction O(N)

- Initialize the result as an identity value for the reduction operation
  - Smallest possible value for max reduction
  - Largest possible value for min reduction
  - 0 for sum reduction
  - 1 for product reduction
- Iterate through the input and perform the reduction operation between the result value and the current input value
  - N reduction operations performed for N input values
  - Each input value is only visited once – an O(N) algorithm
  - This is a computationally efficient algorithm.

A parallel reduction tree algorithm performs N-1 operations in log(N) steps



A tournament is a reduction tree with “max” operation



A Quick Analysis

- For N input values, the reduction tree performs
  - (1/2)N + (1/4)N + (1/8)N + … (1)N = (1- (1/N))N = N-1 operations
  - In Log (N) steps – 1,000,000 input values take 20 steps
    - Assuming that we have enough execution resources
  - Average Parallelism (N-1)/Log(N))
    - For N = 1,000,000, average parallelism is 50,000
    - However, peak resource requirement is 500,000
    - This is not resource efficient
- This is a work-efficient parallel algorithm
  - The amount of work done is comparable to the an efficient sequential algorithm
  - Many parallel algorithms are not work efficient

GPU Teaching Kit

Accelerated Computing