Parallelism

Parallelism is defined as the ratio of the amount of work that can be done in parallel to the total amount of work required to complete a task. The work is the running time on a single processor, denoted by T\_1, while the span is the running time if we could run each strand on its own processor, denoted by T\_Inf. **P = T\_1 / T\_Inf**

Example:

You are cleaning up a park, there are 3 different tasks, each task must be done by at most one person. Task 1: Pick up trash (2 hours), Task 2: Sweep the paths (3 hour), Task 3: Empty Trash (1 hours)

If you work alone, it will cost 6 hours to finish all the tasks. But if you have enough people to help you, i.e. 3 people, it will cost you 3 hours. Therefore, the Parallelism is: P = T\_1 / T\_Inf = 6 / 3 = 2

Slackness

We deﬁne the (parallel) slackness of a multithreaded computation executed on an ideal parallel computer with P processors to be the ratio： **S = (T\_1 / T\_Inf \* P)**

Example:

Similar to the previous example, If you work alone, it will take you 6 hours to finish all the tasks.

But if there are 5 people helping you, the overall time consumption is still 3 hours(each task must be done by at most one person).

Therefore, the Slackness is: S = (T\_1 / T\_Inf) / 6 = (6 / 3) / 6 = 0.33

Spawn, Sync

- Spawn: When we call a procedure, the parent procedure may continue to execute in parallel.

parent node spawn child node TA, TB, TC. don’t need to wait for child processes, executing in parallel.

-Sync: procedure must wait for all its spawned children to complete

Parallel Divide and Conquer

The input array is divided into smaller sub-arrays, which are sorted independently using multiple threads. Each thread handles a sub-array and sorts it using the classic merge sort algorithm. Once all sub-arrays are sorted, they are merged together using the second part of the algorithm.

Parallel Merging

The sorted subarrays are merged together in a parallel fashion. This is achieved using a priority queue and multiple threads. Each thread handles one or more sub-arrays and merges them together using the priority queue. The resulting sorted array is the final output of the algorithm.