15210: Parallel and Sequential Data Structures and Algorithms

CilkLab

Zikang Wang (zikangw)

6.1

$W = O\ (nlogn),\ S = O\ (n)$

6.2

Core	4	8	10	12	14	16	20	24
SpeedUp	3.69	5.49	6.43	5.97	6.22	6.58	5.75	5.75

Speed up is the average of 3 runs. Sometimes the speed up among the 3 runs are quite different (e.g. 5.09x, 6.07x, 6.08 for 24 cores).

In general, the speed up is highest between 10 cores to 16 cores. The maximum speed up is around 6.5x.

It is obvious the speed up is not linear. Quick sort has nlogn work and n span, therefore the parallelism is logn, less than n (linear).

6.3

Spawning new workers for parallel version requires an overhead time. When the number of elements is small, this overhead time may become longer than calculation time. In such case, it is more efficient to use a non-parallel algorithm locally.

For the serial version, it needs to call another function, which involves some operations like memory allocation and stack operations. They would consume some overhead time as well. Therefore for similar reason it is also more efficient to switch in sequential version.

6.4

Step	Work	Span
Choosing pivots	$O\ (\sqrt{n}logn)$	$O\ (\sqrt{n}logn)$
Bucketing elements	$O\ (nlogn)$	$O\ (logn)$
Redistributing elements	$O(n)$	$O(\sqrt{n})$
Local sort	$O(nlogn)$	$O(\sqrt{n}logn)$

Overall work is $O(nlogn)$ and span is $O(\sqrt{n}logn)$. We have $\sqrt{n}$ parallism.

$$W = \sum Work = O(nlogn)$$

$$S = max\ \{Span\} = O\ (\sqrt{n}logn)$$

6.5

From 6.4, we have $\sqrt{n}$ parallism. So for 100M data, $\sqrt{100\text{M}} = 10{,}000 < 100{,}000$. We can only utilize 10,000 cores, so only 10,000 elements can be processed at the same time. Some computation power will be wasted.

7.1

This could work, but it didn't take race condition into consideration. There is a possibility that two workers would write and read at the same time. In this case, it is unpredictable what result we will get. Given a large data set the probability is small but it's not 0.