Foundations of Parallel Programming CSE502 - Project report

Steal Tree: Low-Overhead Tracing of Work Stealing Schedulers

Akshita Sawhney(MT17143)
Ankit Sharma (MT16121)

26 April 2018

1 Introduction

Work stealing is a distinct and a dynamic load balancing strategy. The flexibility of work stealing leads to less structured mapping of tasks to threads. There are two phases in this strategy which are the working phase and the stealing phase. The purpose of this paper is to track the steal operation which is an important component of help first schedulers. Identification of the key properties of the above policy accounts for the randomness. This allows us to trace the execution of tasks with low time and space overhead which in turn leads to reduced cost of data-race detection. There are two algorithms, one of which helps in tracing the steals by forming a steal tree and the other which replays the program using this steal tree formed.

2 Experiments and validation

2.1 Overhead of tracing compared to default cotton runtime

As in the paper, our implementation of steal tree(NQueens 12) also reflects low overheads for tracing as shown in Figure 2. The average execution time for conventional help first is 0.987 secs as compared to trace based implementation which takes 1.033 secs (illustrated in Figure 1).

2.2 Verification of replay and steal-tree using the special implementation of NQueens

As illustrated from Figure 2, we were able to get the exact distribution for most of the cases.

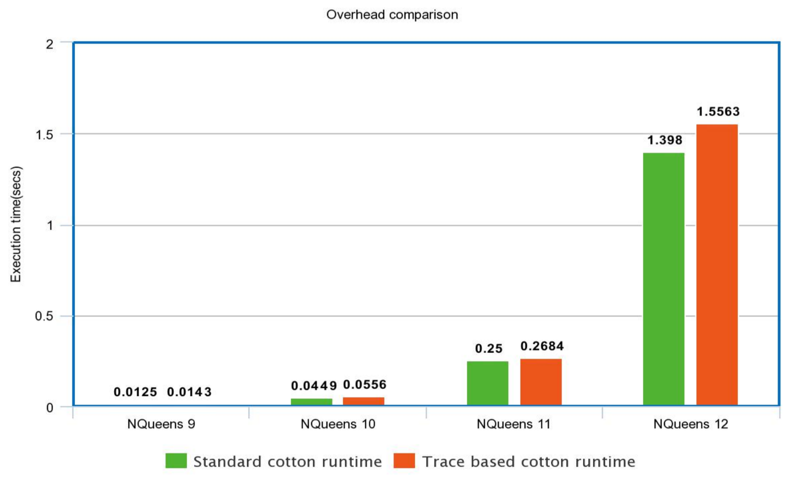
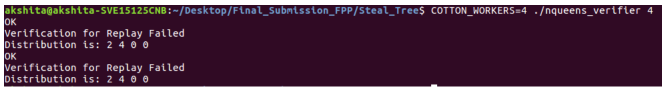


Figure 1: Overhead comparison



```
akshita@akshita-SVE15125CNB:~/Desktop/Final_Submission_FPP/Steal_Tree$ COTTON_WORKERS=4 ./nqueens_verifier 4
OK
Verification for Replay Failed
Distribution is: 2 4 0 0
OK
Verification for Replay Failed
Distribution is: 2 4 0 0
```

Figure 2: Replay verification