Titolo appropriato

Anna Bonaldo Vassilikì Menarin

Email: anna.bonaldo@studenti.unipd.it vassiliki.menarin@studenti.unipd.it

***Abstarct* - Write something interesting here**

1 Introduction

1.1 Work Staling Scheduler

Work stealing is a scheduling strategy designed to efficiently manage a dynamically multi-threaded computation using a fixed number of processors. The work stealing scheduler bounds the number of concurrently active threads within a limit, thus affecting the memory requirements too. This causes an improvement in the execution time and memory usage. Moreover, the scheduler tries to mantain related threads on the same processor, minimizing the communication between different threads. Work stealing differs from work sharing because underutilized processors take the initiative, without relying soley on the scheduler: each processor owns a deque and, when it-s empty, it tries to steal work from others. The number of migrations is lower in work stealing, because it is done only when absolutely necessary.

We built a work stealing scheduler that runs a Quick Sort application. We performed some analysis on the results, comparing the clock-time and CPU time for a work stealing and a sequential scheduling, searching for a set of paramentrs that minimizes wall-clock time.

2 Our work

We developed a work-stealing scheduler and a Quick-Sort application using Java 8. The main routine takes three parameters, and we tried assigning different values to them to find a set that minimizes wall-clock time when compared to sequential sorting. These parameters are:

- `arraySize`: sets the size of the array tp sort;
- `numServers`: sets the number of servers;
- `cutOff`: sets the array size under which to perform sequential sorting and it is used to improve the performances of the work stealing scheduler.

The classes that implement the algorithm are `Scheduler`, `Tasklet` and `QuickSort` We then use `BatchQuickSortExecutors`, `IOFromCSVFile` and `Statistics` to efficiently gather and compute data.

2.1 The Scheduler class

2.2 The Tasklet class

2.3 The QuickSort class

3 Analysis performed

3.1 Results of analysis

4 Conclusion