

SCHOOL OF STEM, COMPUTING & SOFTWARE SYSTEMS, UWB

CSS 534 Assignment 5

Yangde Li
December 11, 2018

Contents

0.1	Introduction	2
0.2	Parallelism Techniques	2
0.3	Performance Analysis	2
0.4	Programmability Analysis	2
0.5	Execution Snapshots or Demo with Graphical Results	2

0.1 Introduction

This report covers an assignment on implementing Ant Colony Optimization with MpiJava.

This is a compulsory exercise in the course CSS 534 Parallel Programming, given by the school of computing & software systems at the University of Washington, Bothell in 2018.

0.2 Parallelism Techniques

In this implementation with MpiJava, we mainly focus on two types of parallelism techniques: **Task** and **Data** decomposition. For the task decomposition, we split the ants to different computing nodes, thus to accelerate whole computing speed. In every optimization loop, the worker nodes send back the delta pheromone generated by the ants in this iteration to master, then master updates the whole pheromone matrix and scatter it to all workers. Another parallel technique is to make every worker run a complete ACO process and update the pheromone matrix using mean value, this is a way to increase the ants searching space and increase the possibility of finding the best path.

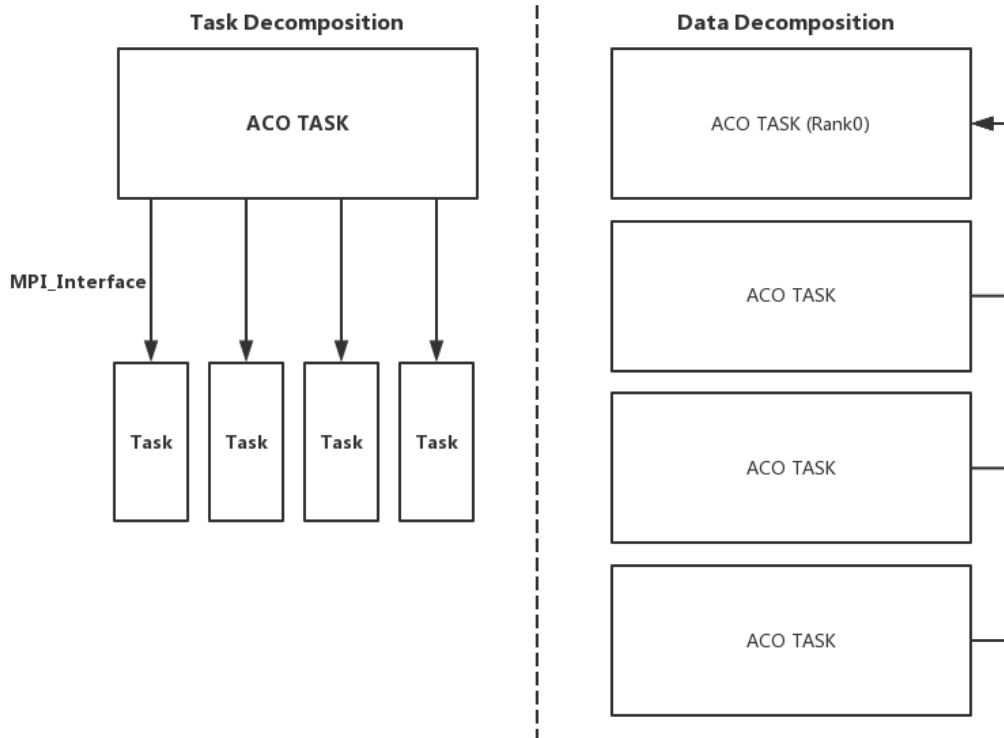


Figure 1: Two types of Parallel Pattern

0.3 Performance Analysis

For task parallelization, computing task is split into different nodes and the exchanged data is rather small so the performance improvement is obvious. However, the Data Decomposition duplicates the whole data set to run in every node, so the performance improvement should be little.

0.4 Programmability Analysis

0.5 Execution Snapshots or Demo with Graphical Results

```
ydli66_css534@cssmpi1:~/mpiJava

[ydli66_css534@cssmpi1 mpiJava]$ ./compile.sh
[ydli66_css534@cssmpi1 mpiJava]$ ./run_parallel2.sh 1000 4
BEST ROUTE:
source
VOI8NXGFA9K4THZJ0EC5W3R7LMP52DBQJ16Y
length: 911.3796427668566
Elapsed time: 12071ms
[ydli66_css534@cssmpi1 mpiJava]$ ./run_parallel2.sh 1000 8
BEST ROUTE:
source
V1Y270D3RXP94G06ALJUEHZ05CNSPMBKNT8I
length: 842.9515869913246
Elapsed time: 10129ms
[ydli66_css534@cssmpi1 mpiJava]$ ./run_parallel2.sh 1000 12
BEST ROUTE:
source
VQX48NT9KFA7LJ0E3RQBDI061YUZHSSQWMP2
length: 835.7741901857437
Elapsed time: 15175ms
[ydli66_css534@cssmpi1 mpiJava]$ ./run_parallel2.sh 1000 16
BEST ROUTE:
source
IO6G3RONS94T8NXKFA7LJ25HZYE0MDQBU1V
length: 818.2931493869263
Elapsed time: 17235ms
[ydli66_css534@cssmpi1 mpiJava]$ ./run_parallel2.sh 1000 20
BEST ROUTE:
source
IO6XN87ALR302PMSWCSUZEBOQT4FKG9JV1YH
length: 890.6644331344435
Elapsed time: 35342ms
[ydli66_css534@cssmpi1 mpiJava]$ ./run_parallel1.sh 1000 4
BEST ROUTE:
source
V1Y2H5CNSMPBQR3D7LAF9KXWGT80I2E0JU6
length: 638.101188553507
Elapsed time: 21135ms
[ydli66_css534@cssmpi1 mpiJava]$ ./run_sequential.sh 1000 4
BEST ROUTE:
source
V1Y2H5CNSMPBQR3D7LAF9KXWGT80I2E0JU6
length: 638.101188553507
Elapsed time: 23396ms
[ydli66_css534@cssmpi1 mpiJava]$
```

Figure 2: Execution Snapshots