APPD
TD n°9 - Scheduling (2)

Adrien Durier Oguz Kaya

10/01/2017

Part 1

Task graph scheduling

Consider scheduling the following task graph on 3 identical processors.

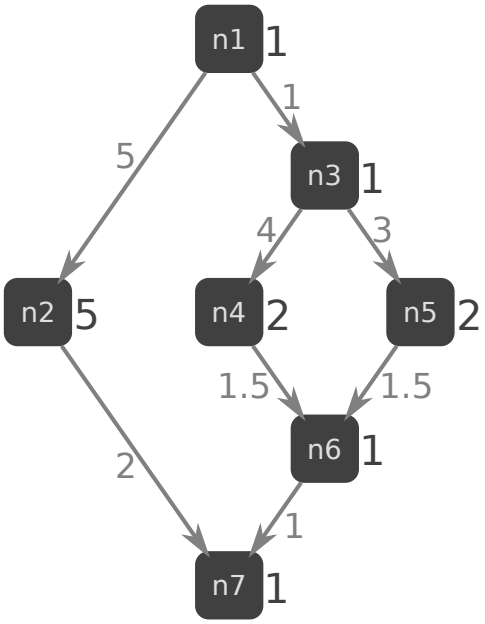

Figure 1: A task graph

1.1

Scheduling without communications

We first disregard the labelling of the edges and assume communications to come for free.

Question 1

a) Compute the bottom level for each node.

b) Schedule the task graph on our 3 processors using a list heuristic. What is the makespan of our schedule? Is it optimal?

1.2

Critical path scheduling

From now on we will consider the communication costs which have to be accounted for when two adjacent tasks are scheduled on different processors.

Question 2

a) How communications should be taken into consideration when computing the bottom level?

b) Compute the bottom level for each node.

c) Schedule the task graph on our 3 processors using the critical path heuristic. What is the makespan of our schedule?

1.3

Modified critical path scheduling

Sometimes, it is worth waiting to schedule a task on a busy processor rather then using the first processor available.

Question 3

a) Which wrong decision, made in the previous section, would be avoided by using this new heuristic?

b) Using this approach, propose a new schedule for our task graph. What is its makespan?

1.4

Clustering

List heuristics are simple but not very efficient. In order to increase the performance of distributed systems, many clustering have been developed. By analysing the graph, Sarkar's algorithm removes the communication cost of some edges by forcing the neighbouring tasks to be executed on the same processor. Clustering algorithms may vary in how they decide which edge to remove (or which cluster to merge).

Question 4

a) Apply the clustering algorithms you saw in class to the example.

b) Schedule the execution of the clusters you obtain by Sarkar's algorithm on our 3 processors. Use the natural load balancing and the free task scheduling as a subheuristics. What is the makespan of this schedule?

c) Is this schedule optimal?

Part 2

Loop parallelization

Considering the following code.

for $i = 1$ *to* N **do**
  for $j = i + 2$ *to* N **do**
    S_1: $a(i+1, j-2) \leftarrow c(i+1, j-1)$
    S_2: $b(i, j+3) \leftarrow a(i+2, j-6)$
    S_3: $c(i+1, j) \leftarrow d(i-2, j+1) + c(i+1, j-5)$
    S_4: $d(i+1, j) \leftarrow a(i+1, j-3) + c(i, j+2)$
    S_5: $e(i+1, j) \leftarrow a(i+1, j-2)$

Question 5

a) Give the dependancy graph, with edges labelled with the type of dependancy and the direction vector.

Question 6

a) Apply Allen and Kennedy's algorithm to the loop nest.

Question 7

a) Recall what condition must a Lamport vector $\pi = (a, b)$ satisfy in order to correctly schedule the loop nest (i.e., execute the iteration I at step number $\pi \cdot I$)?

b) How can one find algorithmically such a vector in the general case?

Question 8

a) Find a Lamport vector minimizing the execution time.

Question 9

a) Between the two, what was the most efficient strategy to parallelize the loop? Can you do better?