CS 425 Homework 6

Mukai Yu

Wednesday, April 27, 2022

1. (a) (ai) For leaders of participant groups:

$$25 + 4 \times 10 + 25 + 4 \times 10 + 25 + 4 \times 10 = 195\ ms$$

For acceptors/learners of participant groups:

$$25 + 4 \times 10 + 25 + 4 \times 10 + 25 + 4 \times 10 + 10 = 205\ ms$$

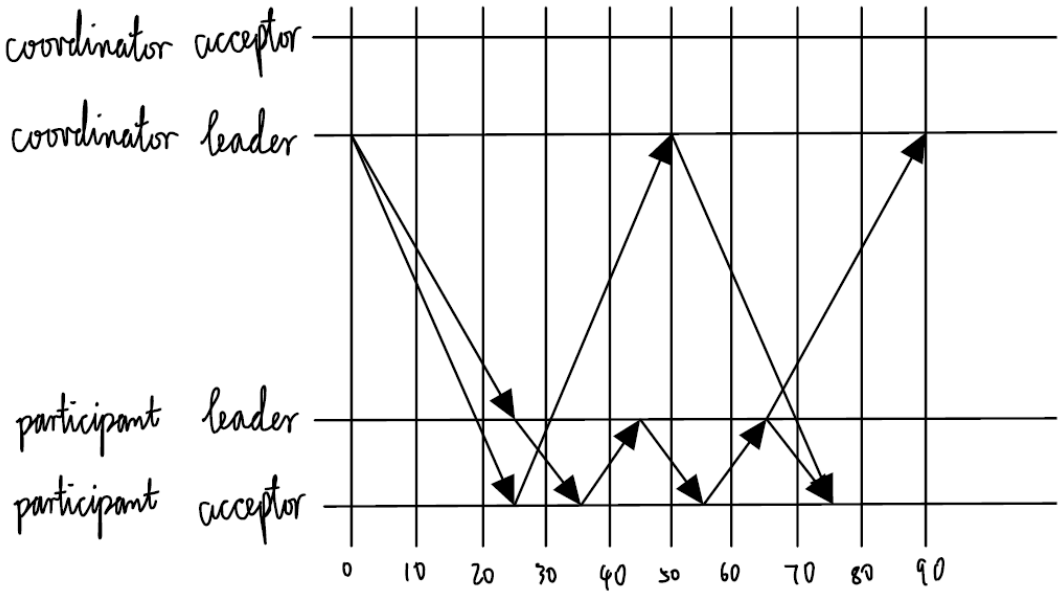
(aii)

$$3 + 3 \times 4 \times 5 + 3 + 4 \times 5 + 3 + 3 \times 4 \times 5 = 149$$

(b) Exactly when the coordinator group reaches consensus (distinguished learner receives accept messages) and its leader is about to send commit messages to leaders of participant groups:

$$25 + 4 \times 10 + 25 + 4 \times 10 = 130\ ms$$

(c)



The leaders of participant groups will always reach consensus faster than the leader of coordinator group, so the result will be the same as (ai):
For leaders of participant groups:

$$25 + 4 \times 10 + 25 + 4 \times 10 + 25 + 4 \times 10 = 195\ ms$$

For acceptors/learners of participant groups:

$$25 + 4 \times 10 + 25 + 4 \times 10 + 25 + 4 \times 10 + 10 = 205\ ms$$

2. (a)

index	value
0	50069
1	50069
2	50069
3	50069
4	50069
5	50069
6	50069
7	50069
8	50135
9	52086
10	52086
11	52086
12	54501
13	58569
14	1127
15	17102

(b) (i)

hop	node
1	49844
2	1127
3	9379
4	11967
5	12158

(ii)

hop	node
1	49844
2	17102
3	25642
4	28481
5	29112
6	29408

(c)

hop	node
1	49844
2	17102
3	25642
4	28481 (failed)
5	26842
6	28926
7	29112
8	29408

3. (a) Group values according to i and whether it belongs to $[V_1, V_2]$ or $[V_3, V_4]$:

function Map1((k, v)):
    if k.n is in [1, 2] **then**
        emit (((k.i, 1), v))
    else
        emit (((k.i, 2), v))
    end if
end function

Sum up $V_1[i] + V_2[i]$ and $V_3[i] + V_4[i]$ respectively:

function Reduce1((k, v)):
    emit ((k.i, sum(v)))
end function

Load-balancing, partition to 4 groups according to i, and compute product $(V_1[i]+V_2[i])\times (V_3[i] + V_4[i])$:

function Map2((k, v)):
    emit (((k % 4), product(v)))
end function

Sum up $(V_1[i] + V_2[i]) \times (V_3[i] + V_4[i])$ to $(V_1[j] + V_2[j]) \times (V_3[j] + V_4[j])$ according to its assigned partition group:

function Reduce2((k, v)):
    emit ((k, sum(v)))
end function

Finally sum up everything together:

fucntion Map3((k, v)):
    emit ((-, v))
end function

function Reduce3((k, v)):
    emit ((-, sum(v)))
end function

(b) Find all inwards edges and outwards edges:

```
function Map1((k, v)):
    for node in v do
        emit ( (node, ("in", k)) )
        emit ( (k, ("out", node)) )
    end for
end function
```

Don't need to care about empty v, because its k will only be the outwards node of some other k.

Create 2-hop edges:

```
function Reduce1((k, v)):
    for inNode in v.in do
        emit ( (k, ("in", inNode)) )        % pass on the inwards edges
        for outNode in v.out do
            emit ( (inNode, ("out2", outNode)) )
        end for
    end for
end function
```

Find 3-hope nodes:

```
function Map2((k, v)):
    for inNode in v.in do
        for out2Node in v.out2 do
            emit ( (inNode, out2Node) )        % final result before collection as list
        end for
    end for
end function
```

Result collection as list:

```
function Reduce2((k, v)):
    emit ( (k, v) )
end function
```