Suppose we have a pagerank DAG like the whiteboard DAG:

               PR1\_1

           /       |

      /             |

    v              v

 PR2\_1  -->  PR2\_2L

where PR2\_1 is a leaf node with a fanout to PR2\_1 and a faninNB to PR2\_2L.

The DASK DAG task input for PR2+1 will be the tuple (PR1\_1) and the task input for

PR2\_2L will be tuple (PR1\_1, PE2\_1).

My code for executing a regular Dask (non-pagerank) task looks like:

Get the task\_inputs for the current task and check whether the task is a leaf task

            task\_inputs = state\_info.task\_inputs

            is\_leaf\_task = state\_info.task\_name in DAG\_info.get\_DAG\_leaf\_tasks()

 Since PR1\_1 is a leaf task, the arg value for the task is the value in task input, e.g., (1, )

args = task\_inputs

Get the task from a dictionary mapping the task name to the function to execute:

task = DAG\_tasks[state\_info.task\_name]

Execute the task:

output = execute\_task(task,args)

where:

     def execute\_task(task,args):

         logger.debug("input of execute\_task is: " + str(args))

         output = task(\*args)

         return output

Put the PA1\_1 task output in the data dictionary using "PR1\_1" as the key:

data\_dict[state\_info.task\_name] = output

Fanout task PR2\_1 with output, which means that PR2\_1 gets all of PR1\_1's output.

Do the faninNB for PR2\_2L with output, which means that PR2\_2L will also get all of PR1\_1's output.

For the execution of PR2\_1, the task input is (PR1\_1) and PR2\_1 is not a leaf.

            task\_inputs = state\_info.task\_inputs

            is\_leaf\_task = state\_info.task\_name in DAG\_info.get\_DAG\_leaf\_tasks()

Since PR2\_1 is not a leaf:

                args = pack\_data(task\_inputs, data\_dict)

                logger.debug(thread\_name + " argsX: " + str(args))

This will set args to PR1\_1's output. PR2\_1 is executed using:

output = execute\_task(task,args)

This is essentially output = task(\*args)

which unpacks args into PR2\_1's parameters.

For PageRank tasks, we want to split the outputs of a task like PR1\_1 into the output for PR2\_1 and the output for PR2\_2L.

The output for PR1\_1 is:

{'PR2\_1': [(2, 0.0075)], 'PR2\_2L': [(5, 0.010687499999999999)], 'PR2\_3': [(3, 0.012042187499999999)]}

This is a dictionary of list of tuples. The keys 'PR2\_1', 'PR2\_2L', and 'PR2\_3' are the fanouts/faninNBs/fanins for PR1\_1, and the list of tuples are the

outputs for the individual outputs for 'PR2\_1', 'PR2\_2L', and 'PR2\_3'.

When we execute leaf task PR1\_1, we do the usual:

            task\_inputs = state\_info.task\_inputs

            is\_leaf\_task = state\_info.task\_name in DAG\_info.get\_DAG\_leaf\_tasks()

and get the leaf task arg as usual:

                    args = task\_inputs

but now we will be executing the pagerank task instead of a Dask task. Note that the same PageRank\_Function\_Driver task (Python function) is used

for all of the tasks. The parameters of PageRank\_Function\_Driver are:

- task name: Since we use a single PageRank\_Function\_Driver  function, we need the task\_name to determine which DAG task we are executing. This task name will

  be used as the name of the file that contains the partition of graph nodes for this pagerank task.

- total number of graph nodes: the pagerank computation needs to know how many nodes are in the complete graph

- result\_dictionary: this is a dictionary of outputs received from the other tasks. For example,PR2\_1 receives output (above) from PR1\_1. Using the key 'PR2\_1', the

  output of PR1\_1 that is intended for PR2\_1 is the list of tuples:  [(2, 0.0075)].

So before we execute task PR2\_1, we need to extract its input value from the  output of PR1\_1. To do this,

we put PR1\_1's output in the data dictionary using "qualified task names" instead of the simple Dask DAG names.

That is, For each key in PR1\_1's output PR2\_1', 'PR2\_2L', and 'PR2\_3 we use 'PR1\_1-PR2\_1', 'PR1\_1-PR2\_2L', and 'PR1\_1-PR2\_3", respectively

as the keys for the data dictionary. So the data dictionary will have:

'PR1\_1-PR2\_1' :  [(2, 0.0075)]

'PR1\_1-PR2\_2L' : [(5, 0.010687499999999999)]

'PR1\_1-PR2\_3' : [(3, 0.012042187499999999)]

So PR1\_1's output has been split.

Now when PR1\_1 does a fanout for PR2\_1, instead of sending the entire PR1\_1 output to PR2\_1, we just send the part of PR1\_1's output

that is intended for PR2\_1.

Note that I send a dictionary of results to PR2\_1 when I fanout PR2\_1. In the normal Dask DAG case, this is:

dict\_of\_results[calling\_task\_name] = output

But for pagerank fanouts I extract PR2\_1's individual output from PR1\_1's output using PR2\_1's name, and then

create the dictionary of results using the qualified name "PR1\_1-PR2\_1'

                    qualfied\_name = str(calling\_task\_name) + "-" + str(name)

                    dict\_of\_results[qualfied\_name] = output[name]

Now PR2\_1 will only receive its part of PR1\_1's output.

PR2\_1 has to put its input (part of its payload) in its data dictionary. Recall its input is the dictionary of results:

        dict\_of\_results = payload['input']

        for key, value in dict\_of\_results.items():

            data\_dict[key] = value

The key will be "PR1\_1-PR2\_1" and the value will be [(2, 0.0075)].

To execute PR2\_1, we need to create the "result\_dictionary" parameter for function  PageRank\_Function\_Driver. We start as usual with:

            task\_inputs = state\_info.task\_inputs

            is\_leaf\_task = state\_info.task\_name in DAG\_info.get\_DAG\_leaf\_tasks()

However, the task\_input as set by DASK for the custom DAG I sent previously is: PR1\_1, which is the unqualified name. So we need

to create a task input that uses qualified names, e.g., 'PR1\_1-PR2\_1' since these names will be used as keys into the data dictionary and

the keys in the data dictionary are qualified names, e.g., "PR1\_1-PR2\_1".

So first create a new task\_input tuple that has qualified names instead of the Dask DAG simple names. (I do not currently do this step

since when I create my DAGs I use qualified names for task\_inputs instead of simple names.)

Assume task\_inputs for PR2\_1 is ('PR1\_1-PR2\_1'). I use:

            result\_dictionary =  {}

            if not is\_leaf\_task:

                logger.debug("Packing data. Task inputs: %s. Data dict (keys only): %s" % (str(task\_inputs), str(data\_dict.keys())))

                # task\_inputs is a tuple of task\_names

                args = pack\_data(task\_inputs, data\_dict)

                logger.debug(thread\_name + " argsX: " + str(args))

                if tasks\_use\_result\_dictionary\_parameter:

                    logger.debug("Foo1a")

                    # task\_inputs = ('task1','task2'), args = (1,2) results in a resultDictionary

                    # where resultDictionary['task1'] = 1 and resultDictionary['task2'] = 2.

                    # We pass resultDictionary of inputs instead of the tuple (1,2).

                    if len(task\_inputs) == len(args):

                        logger.debug("Foo1b")

                        result\_dictionary = {task\_inputs[i] : args[i] for i, \_ in enumerate(args)}

                        logger.debug(thread\_name + " result\_dictionaryX: " + str(result\_dictionary))

Since task\_inputs uses qualified names, the args are constructed as usual:

args = pack\_data(task\_inputs, data\_dict)

but we are not passing args directly to PageRank\_Function\_Driver. Instead we need to construct the result\_dictionary parameter

for PageRank\_Function\_Driver. We do that using

result\_dictionary = {task\_inputs[i] : args[i] for i, \_ in enumerate(args)}

which just matches each key in task\_inputs with its corresponding arg. So the result\_dictionary for PR2\_1 is

'PR1\_1-PR2\_1' : [(2, 0.0075)]

As another example, if PR2\_1's output is the dictionary: {'PR2\_2L': [(0, 0.0075)]} then when we execute faninNB task PR2\_2L, its

result\_dictionary (as collected by the faninNB for PR2\_2L) will be PR2\_2L's inputs from PR1\_1 and PR2\_1 (with qualified names for keys):

     {'PR2\_1-PR2\_2L': [(0, 0.0075)], 'PR1\_1-PR2\_2L': [(5, 0.010687499999999999)]}

Again, the faninNB for PR2\_2L collects a dictionary of the inputs it receives from PR1\_1 and PR2\_1 and this dictionary is given to the Lambda that executes

the faninNB task PR2\_2L. This lambda will put these fanin results in its data dictionary as described above and then execute faninNB task PR2\_2L as

just described by using the task\_inputs for PR2\_2L and the data dictionary to generate args and using the task\_inputs and args to generate the

result\_dictionary for executing PageRank\_Function\_Driver.

The trace of the above code is:

[2023-04-13 09:28:16,058] [Thread\_leaf\_3] DEBUG: Thread\_leaf\_3 execute task: PR2\_2L

[2023-04-13 09:28:16,059] [Thread\_leaf\_3] DEBUG: is\_leaf\_task: False

[2023-04-13 09:28:16,061] [Thread\_leaf\_3] DEBUG: task\_inputs: ('PR2\_1-PR2\_2L', 'PR1\_1-PR2\_2L')  
[2023-04-13 09:28:16,064] [Thread\_leaf\_3] DEBUG: Packing data. Task inputs: ('PR2\_1-PR2\_2L', 'PR1\_1-PR2\_2L'). Data dict (keys only): dict\_keys (['PR1\_1-PR2\_1', 'PR1\_1-PR2\_2L', 'PR1\_1-PR2\_3', 'PR2\_3-PR3\_3', 'PR2\_1-PR2\_2L'])  
[2023-04-13 09:28:16,065] [Thread\_leaf\_3] DEBUG: Thread\_leaf\_3 argsX: ([(0, 0.0075)], [(5, 0.010687499999999999)])  
[2023-04-13 09:28:16,065] [Thread\_leaf\_3] DEBUG: Foo1a  
[2023-04-13 09:28:16,066] [Thread\_leaf\_3] DEBUG: Foo1b  
[2023-04-13 09:28:16,067] [Thread\_leaf\_3] DEBUG: Thread\_leaf\_3 result\_dictionaryX: {'PR2\_1-PR2\_2L': [(0, 0.0075)], 'PR1\_1-PR2\_2L': [(5, 0.010687499999999999)]}  
[2023-04-13 09:28:16,068] [Thread\_leaf\_3] DEBUG: argsZ: ([(0, 0.0075)], [(5, 0.010687499999999999)])  
[2023-04-13 09:28:16,071] [Thread\_leaf\_3] DEBUG: Thread\_leaf\_3 result\_dictionaryZ: {'PR2\_1-PR2\_2L': [(0, 0.0075)], 'PR1\_1-PR2\_2L': [(5, 0.010687499999999999)]}  
[2023-04-13 09:28:16,072] [Thread\_leaf\_3] DEBUG: Thread\_leaf\_3: execute\_task\_with\_result\_dictionary: input of execute\_task is: {'PR2\_1-PR2\_2L': [(0, 0.0075)], 'PR1\_1-PR2\_2L': [(5, 0.010687499999999999)]}

After executing a task, we need to put its output in the data dictionary. (output is a dictionary). In the following code,

state\_info.task\_name

is the name of the executed task.

#   Example: task PR1\_1 produces an output for fanouts PR2\_1

            #   and PR2\_3 and faninNB PR2\_2.

            #       output = {'PR2\_1': [(2, 0.0075)], 'PR2\_2': [(5, 0.010687499999999999)], 'PR2\_3': [(3, 0.012042187499999999)]}

                for (k,v) in output.items():

                    # example: state\_info.task\_name = "PR1\_1" and

                    # k is "PR2\_3" so data\_dict\_key is "PR1\_1-PR2\_3"

                    data\_dict\_key = str(state\_info.task\_name+"-"+k)

                    data\_dict\_value = v

                    data\_dict[data\_dict\_key] = data\_dict\_value

Also, when we execute a leaf task like PR1\_1, we need a result\_dictionary for PageRank\_Function\_Driver. Note that pagerank leaf tasks

have no inputs (so task\_inputs would be an empty tuple). Thus, for leaf tasks we use:

result\_dictionary['DAG\_executor\_driver\_0'] = ()

The PageRank\_Function\_Driver always ignores the actual key values of the result\_dictionary it receives. It simple

concatenates all the values of he result\_dictionary to create a single list of input values for PageRank\_Function:

def PageRank\_Function\_Driver(task\_file\_name,total\_num\_nodes,results\_dictionary):

    input\_tuples = []

    for (\_,v) in results\_dictionary.items():

        if not v == ():

            input\_tuples += v

    output = PageRank\_Function(task\_file\_name,total\_num\_nodes,input\_tuples)

    return output

The input tuples are the pagerank values of  any parent nodes that are needed to compute the pagerank values of the nodes in this partition.