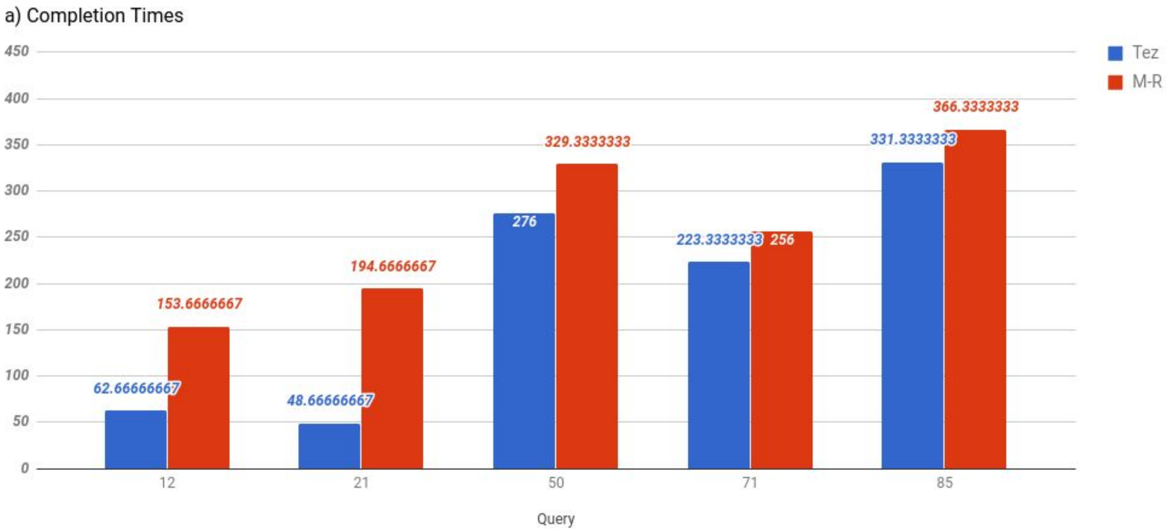
PArt A
1a.

Query	Tez	M-R
12	62.66666667	153.6666667
21	48.66666667	194.6666667
50	276	329.3333333
71	223.3333333	256
85	331.3333333	366.3333333



Tez is always faster than Map reduce.
Due to the architecture of Tez the job's steps are computed before execution and the system can cache intermediate job results in memory. But, in MapReduce all intermediate data between MapReduce phases are written to HDFS i.e. disk adding latency.

Part 1b:
Network bandwidth in bytes

Tez		Avg
	12	6,361,292,950,666,670,000
	21	1,157,242,779,666,670,000
	50	24,489,100,656,333,300,000
	71	35,492,911,377,000,000,000
	85	18,967,835,682,666,700,000
MR	12	8,009,428,532,500,000,000
	21	6,430,727,694,333,330,000
	50	30,626,058,147,666,700,000
	71	35,426,918,164,000,000,000
	85	14,537,329,973,000,000,000

Disk bandwidth:

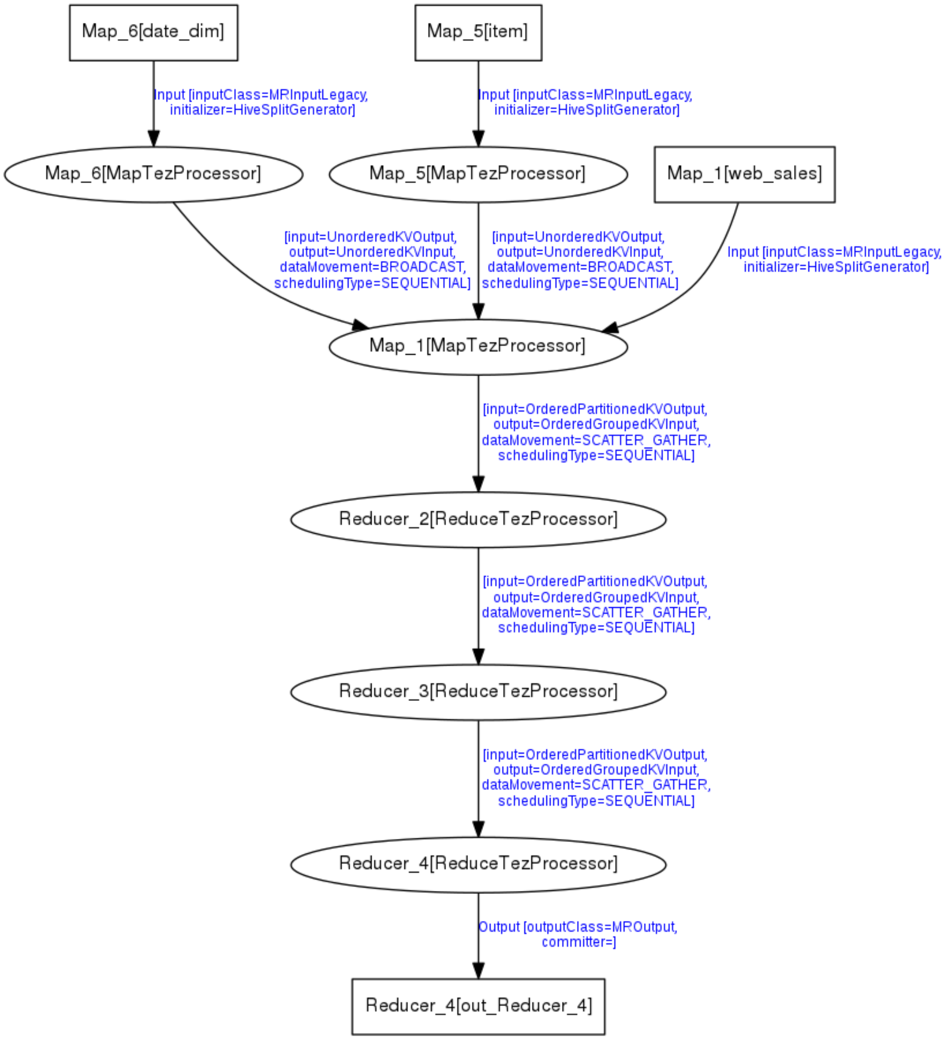
Query	TEZ	MR
12	418181120	5736886272
21	22323200	4653715456
50	8594477056	19173908480
71	69925629952	10406600704
85	122368	7858442240

Map Reduce disk usage > Tez Disk usage by a considerable margin.
This is because MR writes the result of intermediate steps to disk requiring read at the next stage, whereas tez analyses the task DAG and optimises to maintain the intermediate results in memory to speed up execution.

Part 1c:

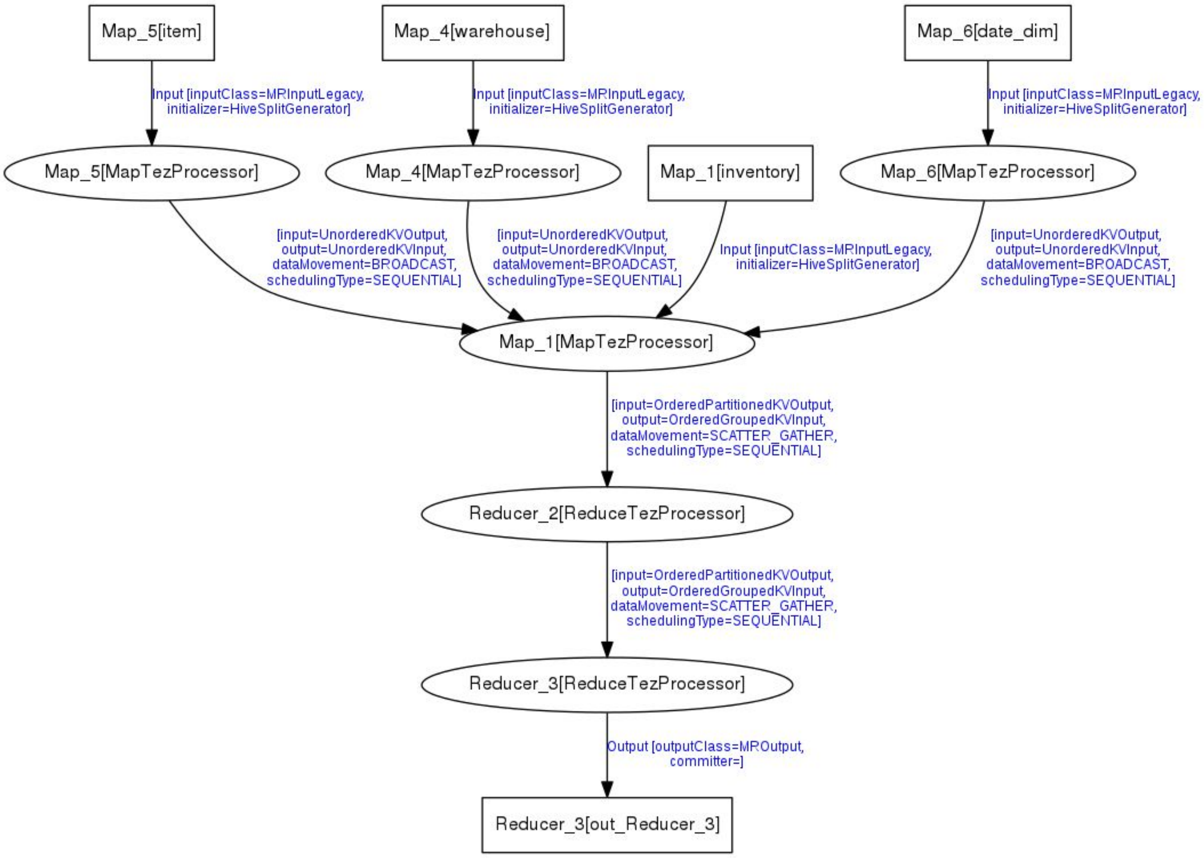
Tasks		Aggregate(red)	Read from HDFS(map)	Ratio	Total
MR	12	3	39	0.07692307692	42
	21	2	20	0.1	22
	50	2	98	0.02040816327	100
	71	2	173	0.01156069364	175
	85	8	51	0.1568627451	59
Tez	12	7	3	2.333333333	10
	21	7	4	1.75	11
	50	9	5	1.8	14
	71	8	14	0.5714285714	22
	85	8	14	0.5714285714	22

Part 1D:
Tez:
Query 12:


Map_6[date_dim]
Map_5[item]
Input [inputClass=MRInputLegacy, initializer=HiveSplitGenerator]
Input [inputClass=MRInputLegacy, initializer=HiveSplitGenerator]
Map_6[MapTezProcessor]
Map_5[MapTezProcessor]
Map_1[web_sales]
[input=UnorderedKVOutput, output=UnorderedKVInput, dataMovement=BROADCAST, schedulingType=SEQUENTIAL]
[input=UnorderedKVOutput, output=UnorderedKVInput, dataMovement=BROADCAST, schedulingType=SEQUENTIAL]
Input [inputClass=MRInputLegacy, initializer=HiveSplitGenerator]
Map_1[MapTezProcessor]
[input=OrderedPartitionedKVOutput, output=OrderedGroupedKVInput, dataMovement=SCATTER_GATHER, schedulingType=SEQUENTIAL]
Reducer_2[ReduceTezProcessor]
[input=OrderedPartitionedKVOutput, output=OrderedGroupedKVInput, dataMovement=SCATTER_GATHER, schedulingType=SEQUENTIAL]
Reducer_3[ReduceTezProcessor]
[input=OrderedPartitionedKVOutput, output=OrderedGroupedKVInput, dataMovement=SCATTER_GATHER, schedulingType=SEQUENTIAL]
Reducer_4[ReduceTezProcessor]
Output [outputClass=MROutput, committer=]
Reducer_4[out_Reducer_4]

ubuntu_20170929224946_4ddfe079_b48e_43ea_a622_86af9c16cb3e_1

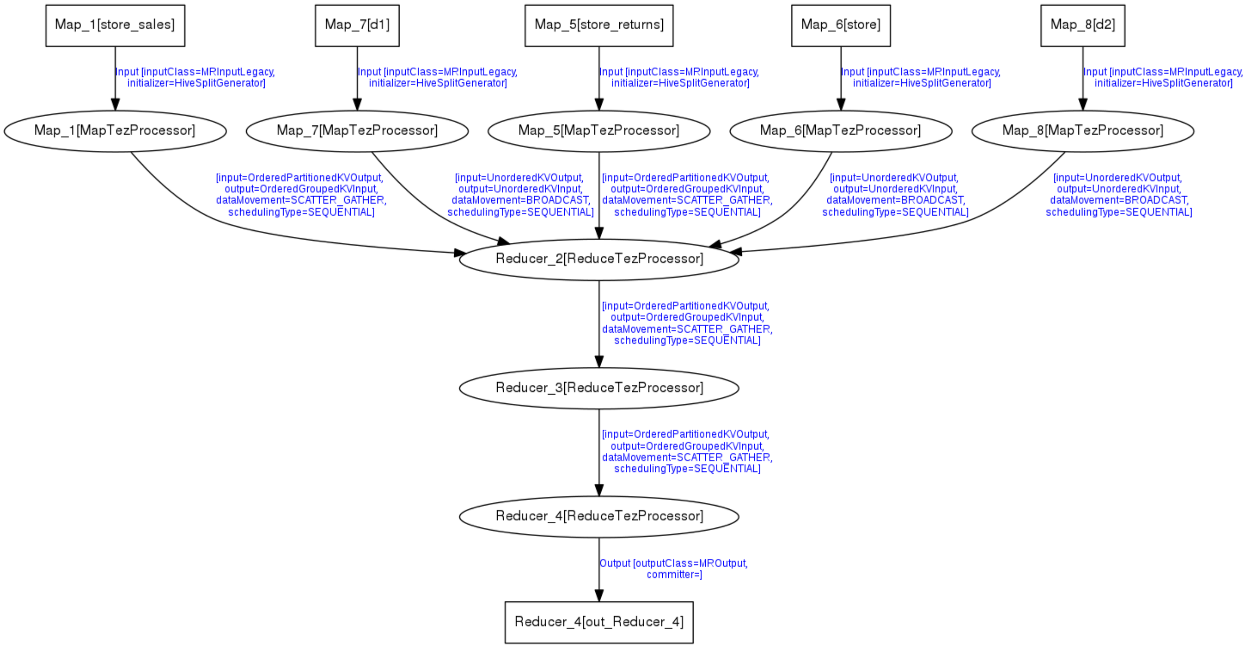
Query 21:



Map_5[item]
Map_4[warehouse]
Map_6[date_dim]
Input [inputClass=MRInputLegacy,
initializer=HiveSplitGenerator]
Input [inputClass=MRInputLegacy,
initializer=HiveSplitGenerator]
Input [inputClass=MRInputLegacy,
initializer=HiveSplitGenerator]
Map_5[MapTezProcessor]
Map_4[MapTezProcessor]
Map_1[inventory]
Map_6[MapTezProcessor]
[input=UnorderedKVOutput,
output=UnorderedKVInput,
dataMovement=BROADCAST,
schedulingType=SEQUENTIAL]
[input=UnorderedKVOutput,
output=UnorderedKVInput,
dataMovement=BROADCAST,
schedulingType=SEQUENTIAL]
Input [inputClass=MRInputLegacy,
initializer=HiveSplitGenerator]
[input=UnorderedKVOutput,
output=UnorderedKVInput,
dataMovement=BROADCAST,
schedulingType=SEQUENTIAL]
Map_1[MapTezProcessor]
[input=OrderedPartitionedKVOutput,
output=OrderedGroupedKVInput,
dataMovement=SCATTER_GATHER,
schedulingType=SEQUENTIAL]
Reducer_2[ReduceTezProcessor]
[input=OrderedPartitionedKVOutput,
output=OrderedGroupedKVInput,
dataMovement=SCATTER_GATHER,
schedulingType=SEQUENTIAL]
Reducer_3[ReduceTezProcessor]
Output [outputClass=MROutput,
committer=]
Reducer_3[out_Reducer_3]

ubuntu_20170929230408_8559a1d0_97bd_47e1_ad0e_4f038dac4393_1

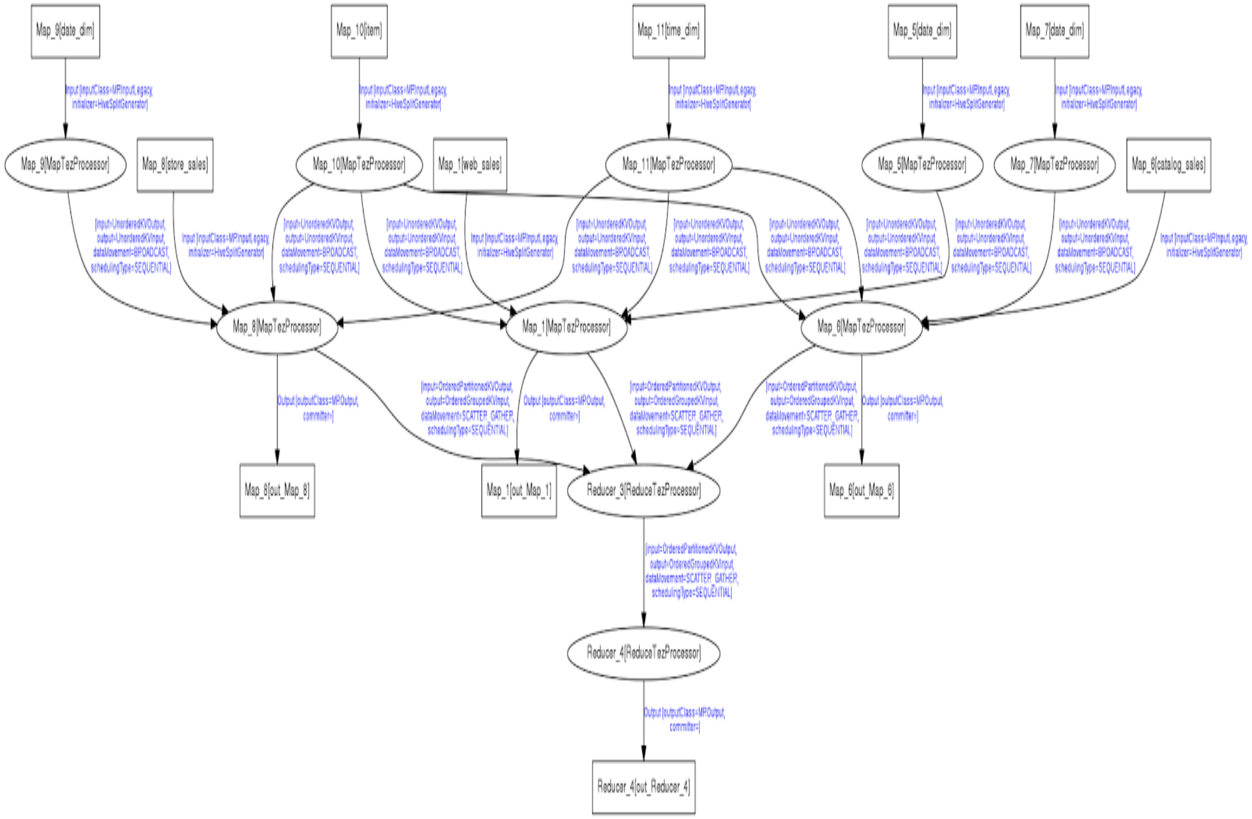
Query 50



Map_1[store_sales]
Map_7[d1]
Map_5[store_returns]
Map_6[store]
Map_8[d2]
Input [inputClass=MRInputLegacy,
initializer=HiveSplitGenerator]
Input [inputClass=MRInputLegacy,
initializer=HiveSplitGenerator]
Input [inputClass=MRInputLegacy,
initializer=HiveSplitGenerator]
Input [inputClass=MRInputLegacy,
initializer=HiveSplitGenerator]
Input [inputClass=MRInputLegacy,
initializer=HiveSplitGenerator]
Map_1[MapTezProcessor]
Map_7[MapTezProcessor]
Map_5[MapTezProcessor]
Map_6[MapTezProcessor]
Map_8[MapTezProcessor]
[input=OrderedPartitionedKVOutput,
output=OrderedGroupedKVInput,
dataMovement=SCATTER_GATHER,
schedulingType=SEQUENTIAL]
[input=UnorderedKVOutput,
output=UnorderedKVInput,
dataMovement=BROADCAST,
schedulingType=SEQUENTIAL]
[input=OrderedPartitionedKVOutput,
output=OrderedGroupedKVInput,
dataMovement=SCATTER_GATHER,
schedulingType=SEQUENTIAL]
[input=UnorderedKVOutput,
output=UnorderedKVInput,
dataMovement=BROADCAST,
schedulingType=SEQUENTIAL]
[input=UnorderedKVOutput,
output=UnorderedKVInput,
dataMovement=BROADCAST,
schedulingType=SEQUENTIAL]
Reducer_2[ReduceTezProcessor]
[input=OrderedPartitionedKVOutput,
output=OrderedGroupedKVInput,
dataMovement=SCATTER_GATHER,
schedulingType=SEQUENTIAL]
Reducer_3[ReduceTezProcessor]
[input=OrderedPartitionedKVOutput,
output=OrderedGroupedKVInput,
dataMovement=SCATTER_GATHER,
schedulingType=SEQUENTIAL]
Reducer_4[ReduceTezProcessor]
Output [outputClass=MROutput,
committer=]
Reducer_4[out_Reducer_4]

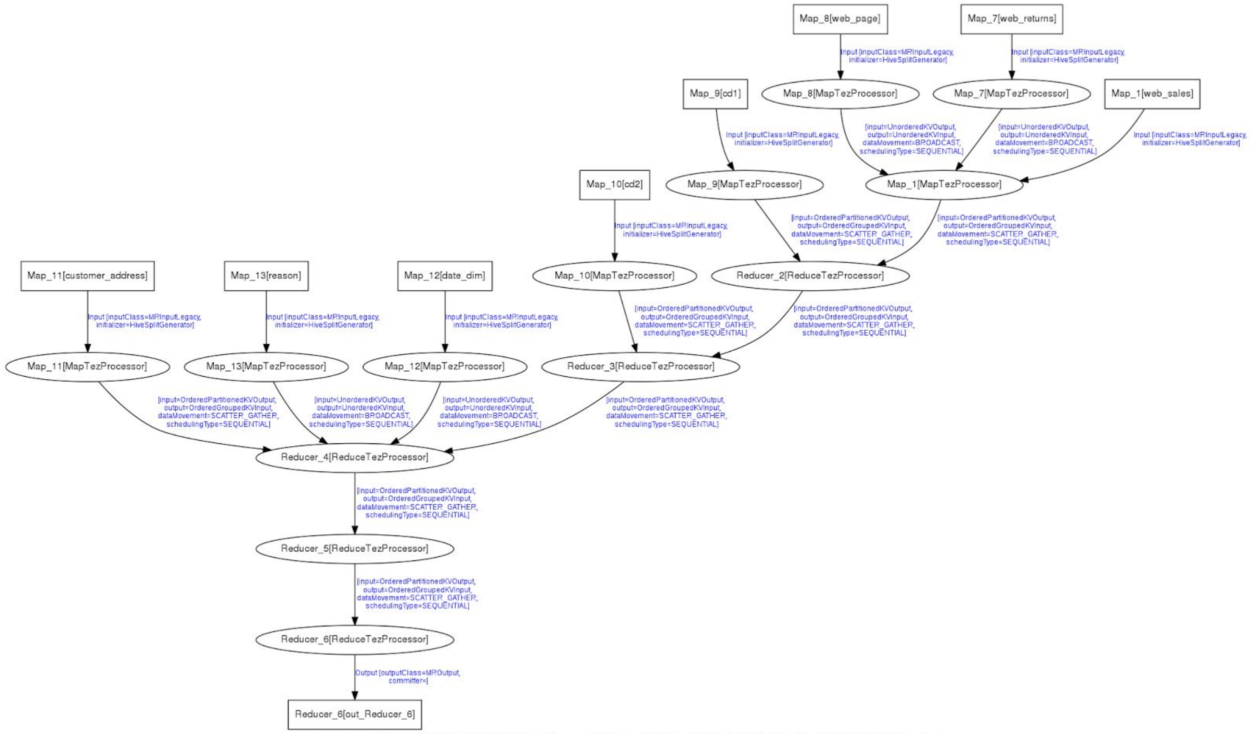
ubuntu_20170930035816_2f1a22e2_1100_4b1a_8422_172470fde80f_1

Query 71:


Map_9[date_dim]
Map_10[item]
Map_11[time_dim]
Map_5[date_dim]
Map_7[date_dim]
Map_9[MapTezProcessor]
Map_8[store_sales]
Map_10[MapTezProcessor]
Map_1[web_sales]
Map_11[MapTezProcessor]
Map_5[MapTezProcessor]
Map_7[MapTezProcessor]
Map_6[catalog_sales]
Map_8[MapTezProcessor]
Map_1[MapTezProcessor]
Map_6[MapTezProcessor]
Map_8[out_Map_8]
Map_1[out_Map_1]
Reducer_3[ReduceTezProcessor]
Map_6[out_Map_6]
Reducer_4[ReduceTezProcessor]
Reducer_4[out_Reducer_4]
Input [inputClass=MRInputLegacy, initializer=HiveSplitGenerator]
[input=UnorderedKVOutput, output=UnorderedKVInput, dataMovement=BROADCAST, schedulingType=SEQUENTIAL]
[input=OrderedPartitionedKVOutput, output=OrderedGroupedKVInput, dataMovement=SCATTER_GATHER, schedulingType=SEQUENTIAL]
Output [outputClass=MROutput, committer=]

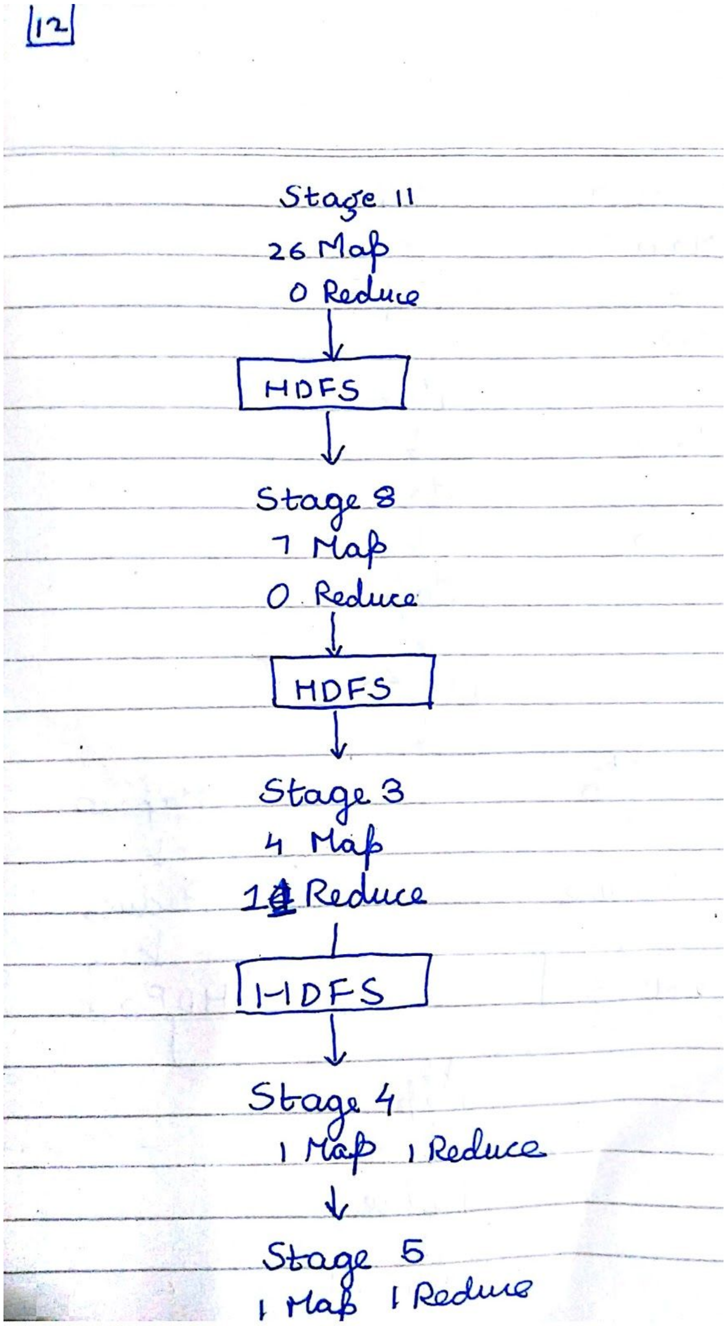
ubuntu_20170930041441_5fbae3d1_c017_49ac_a0ef_989e3a52aa57_1

Query 85:

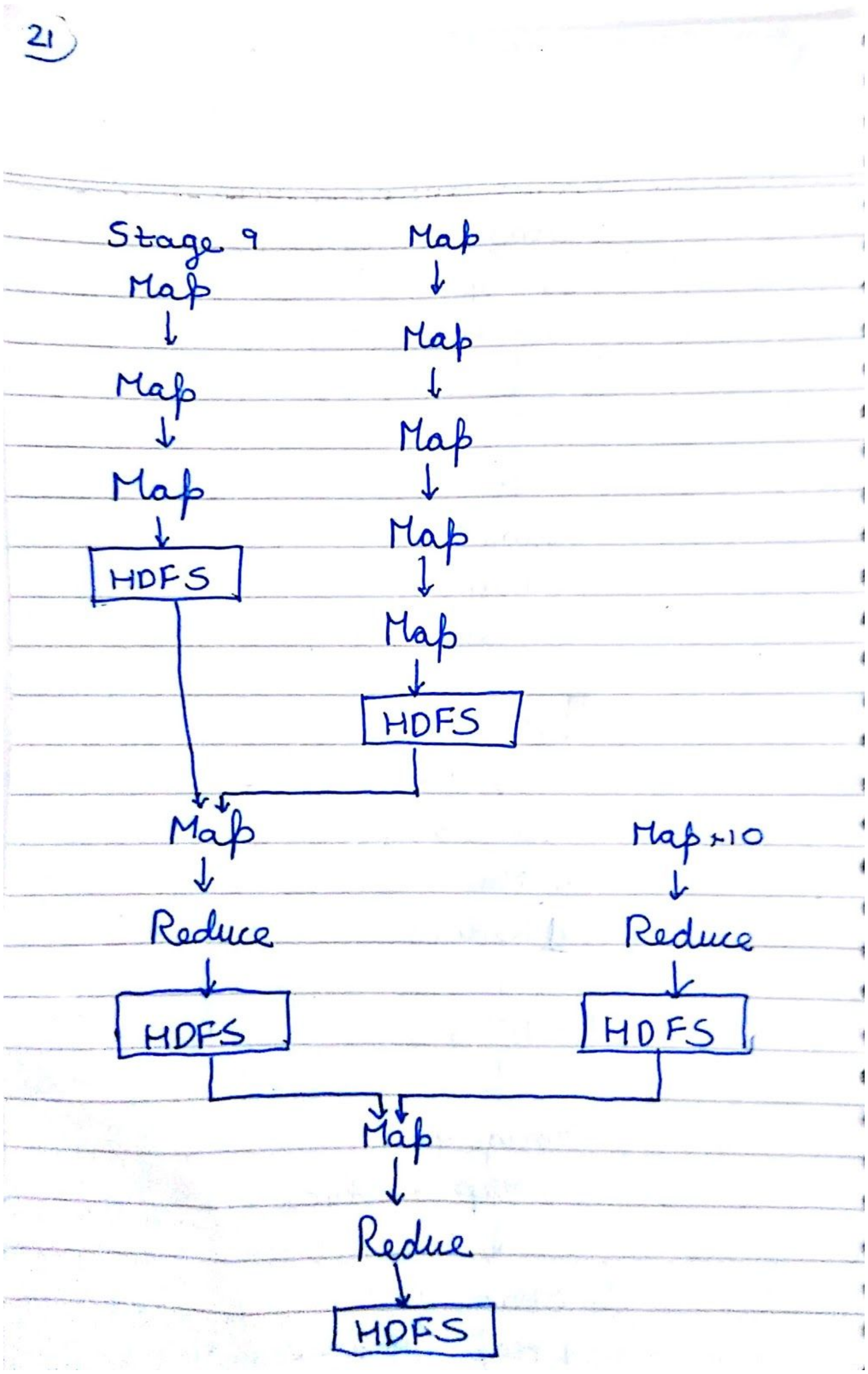

Map_8[web_page]
Map_7[web_returns]
Map_9[cd1]
Map_8[MapTezProcessor]
Map_7[MapTezProcessor]
Map_1[web_sales]
Map_10[cd2]
Map_9[MapTezProcessor]
Map_1[MapTezProcessor]
Map_11[customer_address]
Map_13[reason]
Map_12[date_dim]
Map_10[MapTezProcessor]
Reducer_2[ReduceTezProcessor]
Map_11[MapTezProcessor]
Map_13[MapTezProcessor]
Map_12[MapTezProcessor]
Reducer_3[ReduceTezProcessor]
Reducer_4[ReduceTezProcessor]
Reducer_5[ReduceTezProcessor]
Reducer_6[ReduceTezProcessor]
Reducer_6[out_Reducer_6]
Input [inputClass=MRInputLegacy, initializer=HiveSplitGenerator]
[input=UnorderedKVOutput, output=UnorderedKVInput, dataMovement=BROADCAST, schedulingType=SEQUENTIAL]
[input=OrderedPartitionedKVOutput, output=OrderedGroupedKVInput, dataMovement=SCATTER_GATHER, schedulingType=SEQUENTIAL]
Output [outputClass=MROutput, committer=]

ubuntu_20170930042057_11aed344_655f_473d_89d4_0a61716921cd_1

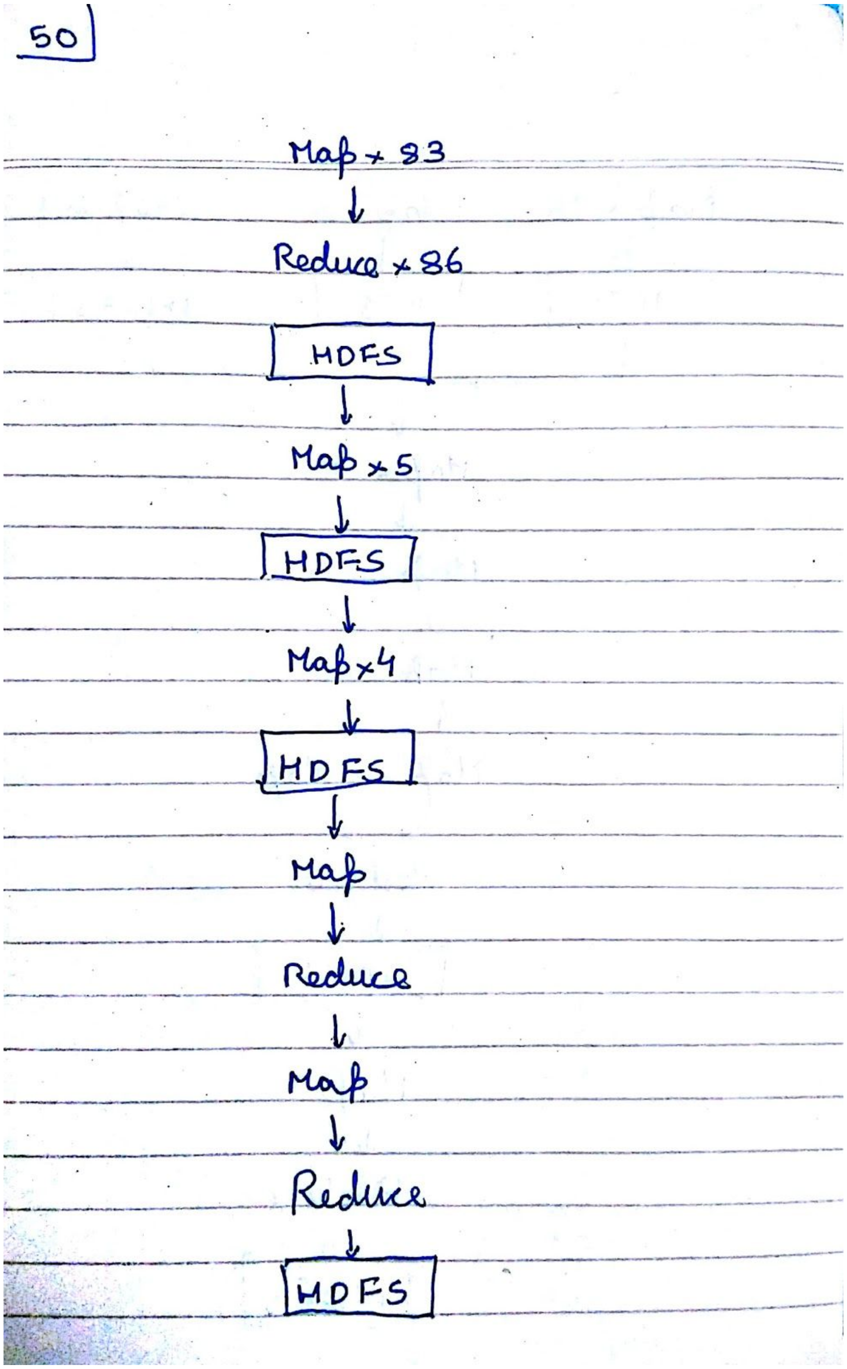
Query 12:


12
Stage 11
26 Map
0 Reduce
HDFS
Stage 8
7 Map
0 Reduce
HDFS
Stage 3
4 Map
1 Reduce
HDFS
Stage 4
1 Map 1 Reduce
Stage 5
1 Map 1 Redure

Query 21:

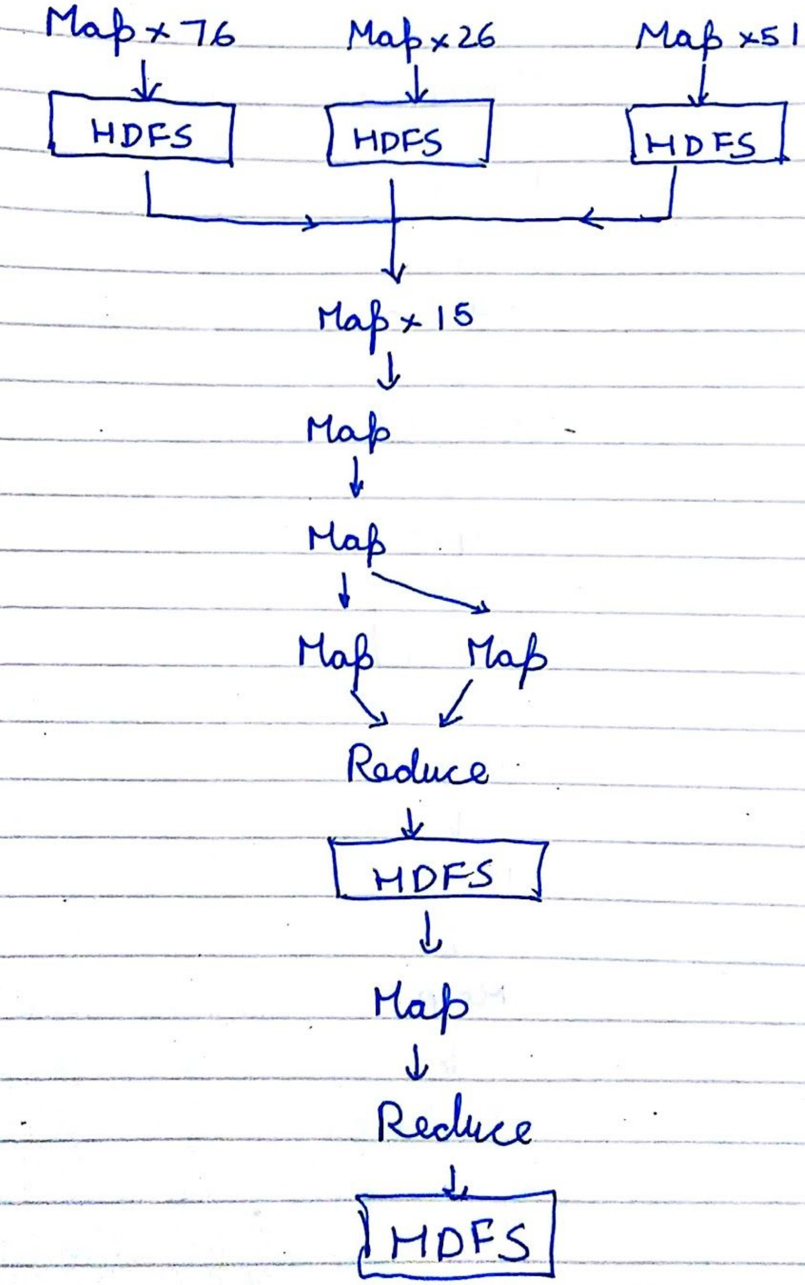

21
Stage 9
Map
Map
Map
HDFS
Map
Map
Map
Map
Map
HDFS
Map
Reduce
HDFS
Map x10
Reduce
HDFS
Map
Redue
HDFS

Query 50:

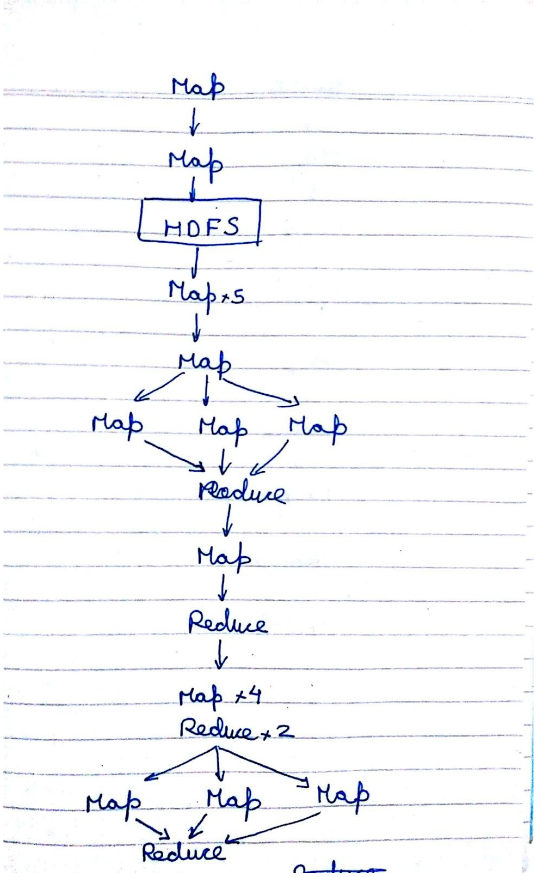


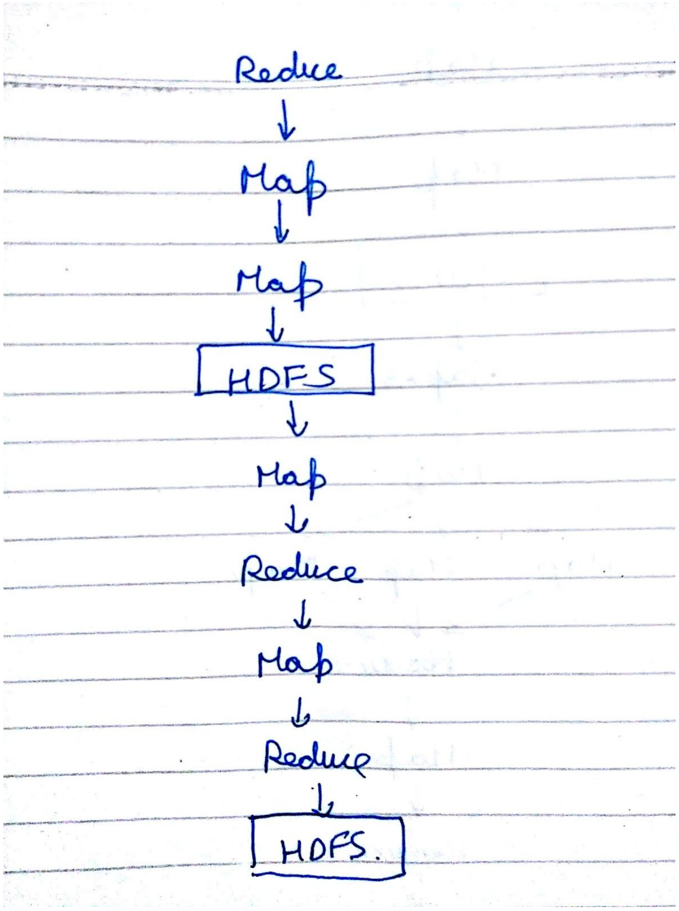
Query 71:

71)


Map x 76
Map x 26
Map x 51
HDFS
HDFS
HDFS
Map x 15
Map
Map
Map
Map
Reduce
HDFS
Map
Reduce
HDFS

Query 85:


Map
Map
HDFS
Map×5
Map
Map
Map
Map
Reduce
Map
Reduce
Map ×4
Reduce×2
Map
Map
Map
Reduce
~~Reduce~~


Reduce
Map
Map
HDFS
Map
Reduce
Map
Reduce
HDFS.

Part 2:
Percent at which data node is killed vs Execution time

Framework		Execution time
Tez	Normal	220
	Killed at 25%	216
	Killed at 75%	225
MR	Normal	317
	Killed at 25%	320
	Killed at 75%	350

We observe a small but proportionally insignificant increase in execution times when 1 datanode is killed at various times.
This is due to two reasons:

1. Replication factor of 2 implies there are multiple copies of the data from the failing data node. Thus the framework does not lose any necessary inputs
2. Speculative execution: Hadoop schedules redundant copies of the same task. So if the tasks fail at one node, there are redundant executions of the same whose output can be used.