CS 838

Assignment 1

Karan Bavishi, Hasnain Ali Pirzada
Group 15

Table of Contents

Question 1

Q1 Part A: Query run time comparison

The major factors that can affect the performance of an engine are:

A. Available network bandwidth
B. Disk read/write speed
C. DAG structural properties

Hive/MR is more susceptible to A than Hive/Tez. This is because Hive/MR uses a lot more network bandwidth owing to the limitations of the MR programming model - where each Map-Reduce stage has to be followed by writing the output to HDFS. On the other hand, Hive/Tez does not suffer from this problem. A lot of tasks (including Map tasks) can read input from intermediate files and not just from HDFS. This means that MR ends up using much more network bandwidth than Tez, and thus is impacted more in case of a **network bottleneck**.

Tez is affected more by B than MR. This happens because in most Tez DAGs, only the final stage writes its output to HDFS. All other stages read their input from local intermediate files. This can sometimes lead to **disk contention** because both output generating tasks and input-reading downstreaming tasks can end up competing for disk resources because the files are only available on one node. On the other hand, since Map tasks in MR fetch their inputs from HDFS and there are multiple replicas available, its performance is not hampered as much by the disk speed.

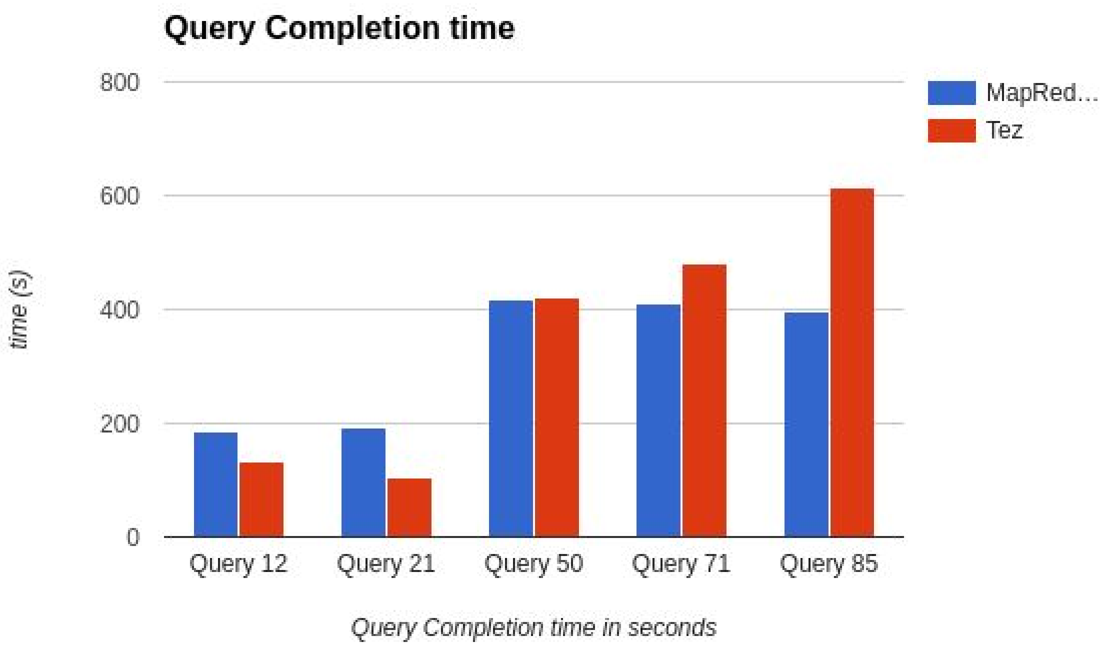
C affects both MR and Tez in different ways. The impact differs based on the nature of the DAGs generated by both engines:

- *More barriers in MR because of programming model limitations* - More *barriers* exist in MR because of the restriction of writing to HDFS after each stage. This can hamper performance if each stage writes a lot of output to HDFS, because there will be long periods where very few tasks are running.

- *Lack of barriers in Tez can lead to better utilization, but also sometimes contention* - Since most tasks in Tez can fetch their input from intermediate files, tasks in subsequent stages are not blocked by barriers. This can generally lead to better utilization of resources, and thus better performance. However, sometimes this can also lead to contention on specific resources and create hotspots because of a lack of a uniform data distribution. Eg. In certain cases, the disk may become a bottleneck because a lot of tasks are writing their output and reading their input from the same disk on a node.

Given this, we can anticipate that the question of which engine would perform better would depend on various parameters. There would be no one clear winner in all situations.

	Query Completion Time (seconds)				
Engine	*Query 12*	*Query 21*	*Query 50*	*Query 71*	*Query 85*
MapReduce	185.49	194.42	416.29	411.52	396.57
Tez	135.42	107.08	420.23	482.36	613.41



After running the queries, we see that no engine outperforms the other in all the queries. The performance of both Hive/MR and Hive/Tez seems to depend a lot on the query being run. The following observations can be made:

1. Hive/Tez performs better than Hive/MR in query 12 and query 21. This is probably because Hive/MR ends up using more network bandwidth than Hive/Tez:
    a. For query 12, MR transfers 7.53 GB of data compared to 4.42 GB by Tez
    b. For query 21, MR transfers 2.61 GB of data compared to 0.71 GB by Tez.

2. Both perform roughly equally on query 50 - Although MR transfers more bytes over the network than Tez (25.05 GB vs. 19.04 GB), both queries have roughly equal completion times. An interesting observation is that MR spawns many more tasks compared to Tez (188 tasks vs. 91 tasks). MR's disadvantage due to higher network usage is probably offset by better resource utilization offered by spawning more tasks

3. Hive/MR performs better than Hive/Tez on query 71 and query 85 - This difference in performance is due to the disk becoming a bottleneck for Hive/Tez. In general, Tez constructs a DAG in which only the final stage writes its output to HDFS and all the other stages use intermediate files. Using intermediate files instead of writing to

HDFS allows Tez to eliminate barriers, but this can come at the expense of disk becoming a bottleneck. Tasks writing their output to intermediate files, and tasks trying to read inputs from intermediate files end up competing for disks. This can be verified by comparing the disk bytes read / written on Tez vs. MR:

a. For query 71, Tez reads/writes 82.25 GB of data to disk compared to 44.41 GB by MR.
b. For query 85, Tez reads/writes 104.36 GB of data to disk compared to only 13.56 GB by MR.

Q1 Part B: Network & Disk Usage Analysis

<u>Network Usage: Expectations & Anomalies</u>

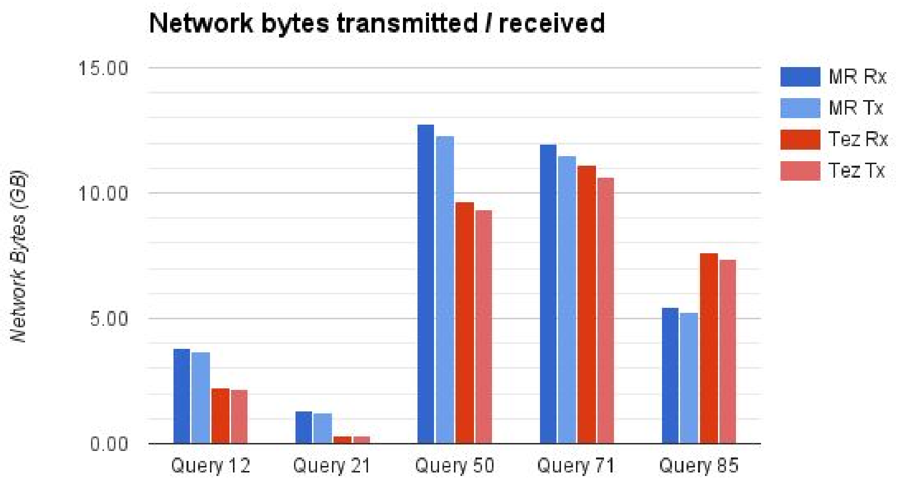
- In general, we expect MR to use a lot more network bandwidth than Tez because it has more stages writing their output to HDFS and thus should end up sending a lot more data (replicas) over the network.

- Our expectations are met for query 12, 21, 50 and 71.

- Query 85 is an anomaly - Tez ends up using more than 40% more bandwidth than MR. Sometimes Tez can end up with unusually high network usage numbers because of its decision to have the HDFS writing stage only at the end, and use intermediate files for communication semantics.

  Generally, writing to HDFS means that network usage will be more because of replicas being sent out. However, this also makes it possible for subsequent Map tasks to do data local computations. Since Tez has a lot of stages further down in the DAG which have to fetch their data from multiple inputs over the network, there are more remote input reads than local input reads.

	Network Usage (GBs)				
Engine	*Query 12*	*Query 21*	*Query 50*	*Query 71*	*Query 85*
MR Rx	3.84	1.33	12.77	11.97	5.46
MR Tx	3.69	1.28	12.28	11.48	5.24
Tez Rx	2.26	0.36	9.70	11.12	7.65
Tez Tx	2.17	0.35	9.34	10.67	7.36



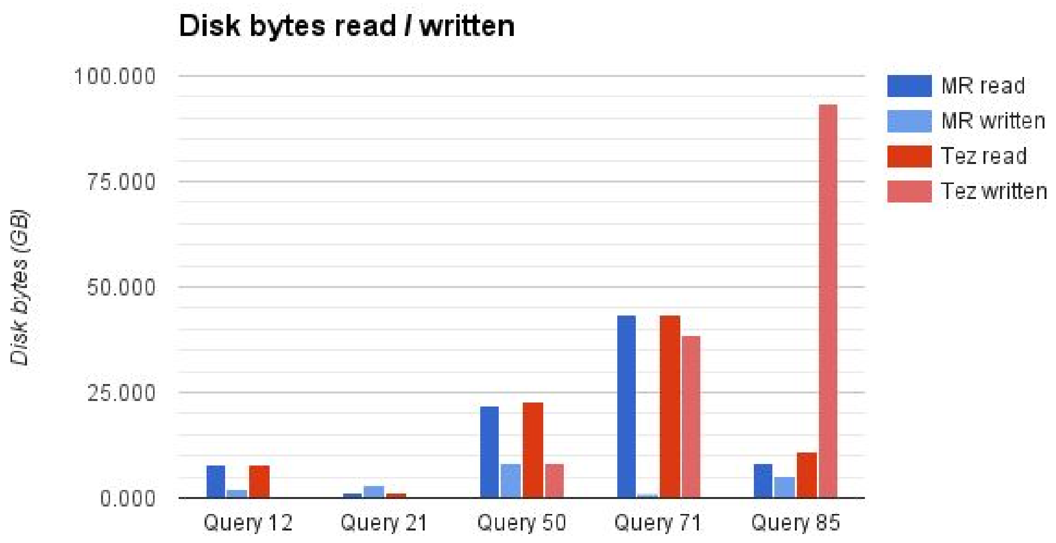
Disk Usage: Expectations & Anomalies

- In the general case, we expect the disk usage numbers for MR to be slightly higher than those for Tez, on account of more stages writing their output to HDFS. Since HDFS maintains two replicas for each block, it will end up writing more bytes to disk. Therefore we expect MR disk usage numbers to be higher than that of Tez.

- The numbers match our expectations for query 12, 21 and 50 - MR bytes read/written are higher or almost equal to the Tez figures.

- Query 71 and query 85 are anomalies to this trend: Tez ends up with 1.85x and 7.7x disk usage numbers than MR for query 71 and 85 respectively. These are probably due to sub-optimal query plans made by Tez, which results in various of stages generating a lot of intermediate data, and thus result in large number of "*shuffled bytes*". Large shuffle sizes can inflate the number of bytes being written / read from disk.

	Disk Usage (GBs)				
Engine	*Query 12*	*Query 21*	*Query 50*	*Query 71*	*Query 85*
MR read	7.785	1.221	22.062	43.367	8.352
MR written	2.312	3.271	8.351	1.047	5.207
Tez read	7.892	1.332	22.665	43.474	11.082
Tez written	0.034	0.011	8.271	38.776	93.279

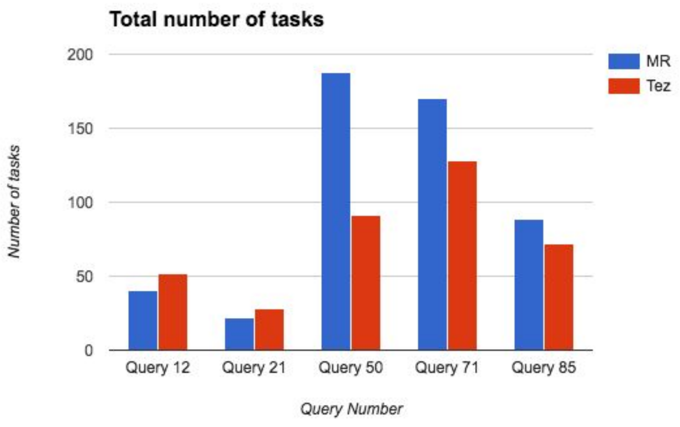


Q1 Part C: Task statistics & Analysis

<u>Comparison of total tasks between MR and Tez</u>

The following table shows the total number of tasks that were run for both the engines and for all the queries. The values have also been shown in a chart for easier comparison

	Total number of tasks				
Engine	*Query 12*	*Query 21*	*Query 50*	*Query 71*	*Query 85*
MR	40	22	188	170	89
Tez	52	28	91	128	72

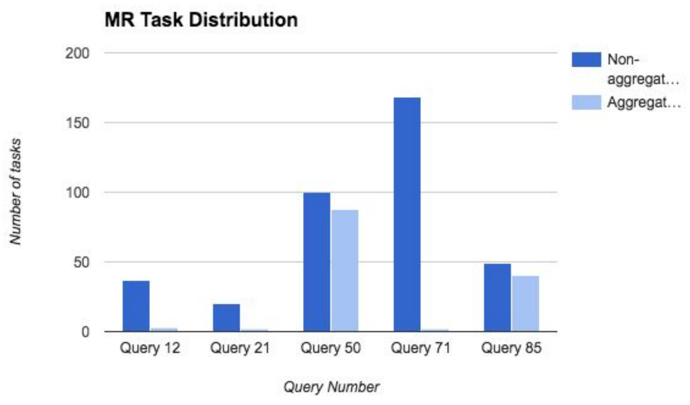


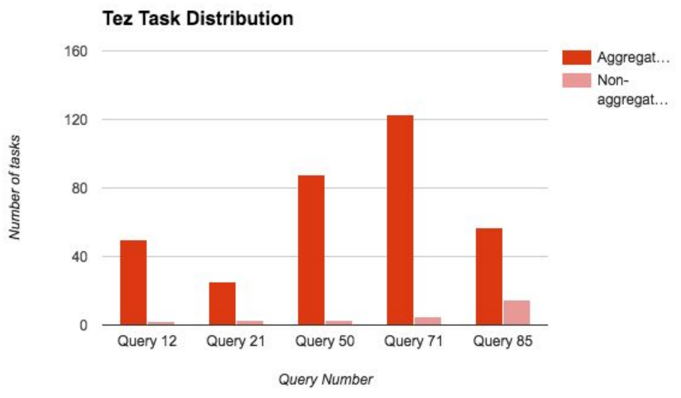
<u>Comparison of task ratios between MR and Tez</u>

The ratio of aggregator tasks to non-aggregator tasks for both engines has been described in the following table.

	Ratio of aggregator tasks to non-aggregator tasks				
Engine	*Query 12*	*Query 21*	*Query 50*	*Query 71*	*Query 85*
MR	0.08	0.10	0.88	0.01	0.82
Tez	25.00	8.33	29.33	24.60	3.80

The charts below depict the task distributions for both engines.





Correlation Analysis for total tasks and task ratios

The *r-squared* values for the two abovementioned metrics and performance have been shown in the following table:

R-squared values		
Property	*Number of tasks*	*Task Ratio*
MR query run time	0.7905	0.3560
Tez query run time	0.4915	0.0148

As can be seen, the correlation between the metrics and the performance depends on the engine. The following conclusions can be made:

- Number of tasks:
    - MR - There is a significant correlation (0.79 R^2) between the total number of tasks and completion time for Hive/MR. In other words, the more tasks we seem to have, the longer it takes for MR to finish. This can be explained in the following ways:
        - It can be argued that having more tasks should improve performance because of parallelism. However in MR, parallelism has restricted impact because of the nature of the programming model. Computation

has to be performed in sequential stages and the stages can't be run in parallel because of dependencies

        - Since parallelism doesn't help performance, having more tasks can only result in a higher completion time.

    - Tez - There seems to be a much weaker correlation (0.49 R^2) between the number of tasks and completion time for Hive/Tez. This can be explained by the following reasons:
        - Impact of parallelism: Since Tez models computation as a DAG, it doesn't suffer from the computational limitations of MR and thus can reduce completion time by parallelizing more tasks.

        - However having more tasks run in parallel can increase resource contention and thus impact query completion time. The disk contention is even more pronounced in Tez compared to MR because most stages read their inputs from intermediate files rather than HDFS (which has replicas).

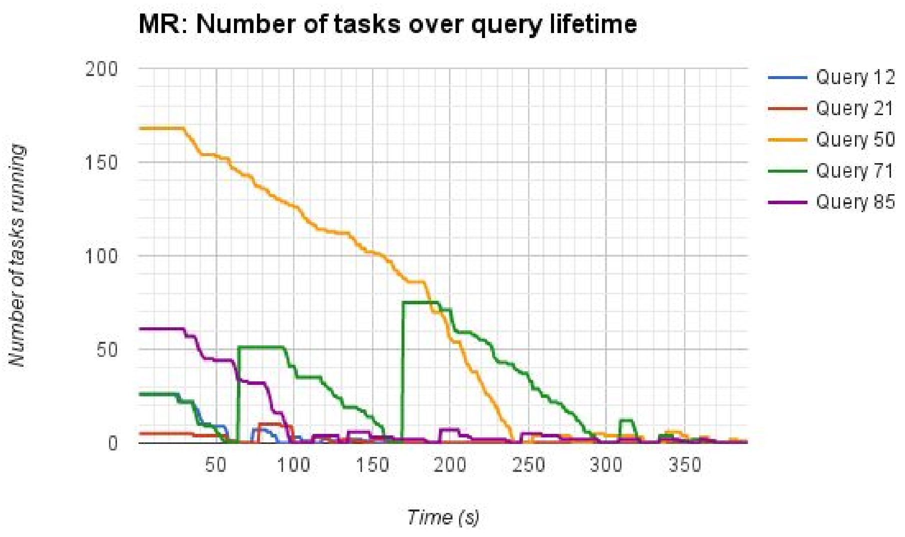
- Ratio of aggregator tasks to non-aggregators
    - MR: There seems to be little correlation between the ratio of tasks and performance - 0.35 R^2. Increasing the ratio would mean increasing the number of reduce tasks relative to map tasks. Having more reduce tasks can reduce completion time, but not in the following cases:
        - Reduce operator is not associative - In this case, the reduce tasks have to wait for all map tasks to complete. Eg. Median computation.

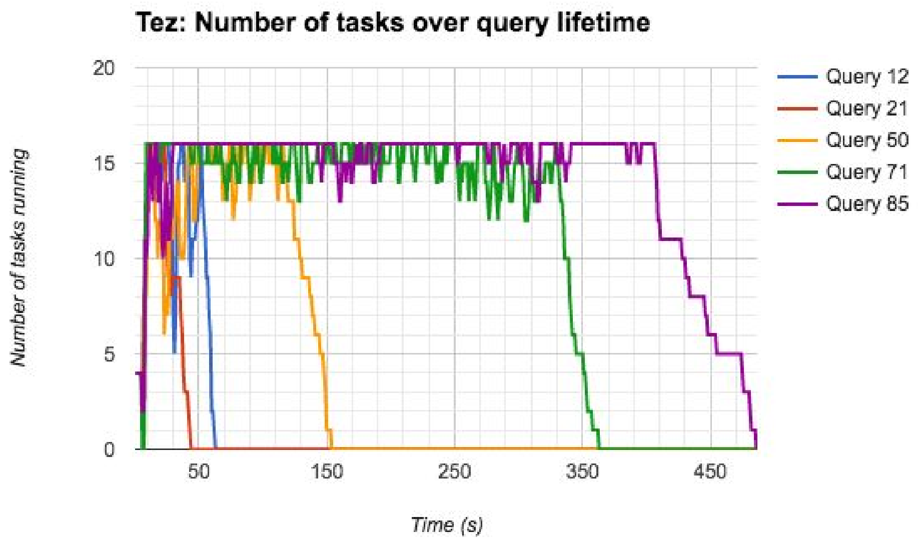
        - Not much intermediate data to process - For certain MapReduce tasks, the amount of intermediate data is very little. Eg: Distributed grep.

    - Tez: There seems to be almost zero correlation between the ratio of tasks and performance for Tez - 0.01 R^2. Increasing the number of aggregator tasks relative to non-aggregators wouldn't guarantee improvement in completion time in all cases because there may not be many records to process.

<u>**Task distribution over query lifetime for MR and Tez: Analysis and correlation**</u>

The following charts depict the number of running tasks over the query lifetime for both the engines.





The following observations can be made:

- Hive/MR:
  - For certain queries with large number of stages (eg. query 71), we see quite a few intermediate periods of no task activity ie. “barriers”. This is because of the inherent limitations of the MR model where each stage has to write its output to HDFS before the subsequent stages can begin

- Hive/Tez
  - There seems to be an upper limit on the total number of tasks that can be running at a time in Tez (16). As soon as one task finishes, a new task is started very soon. This is possible because of the DAG structure of Tez which avoids writing output to HDFS and uses intermediate files for most stages. This allows tasks from subsequent stages to be run immediately instead of

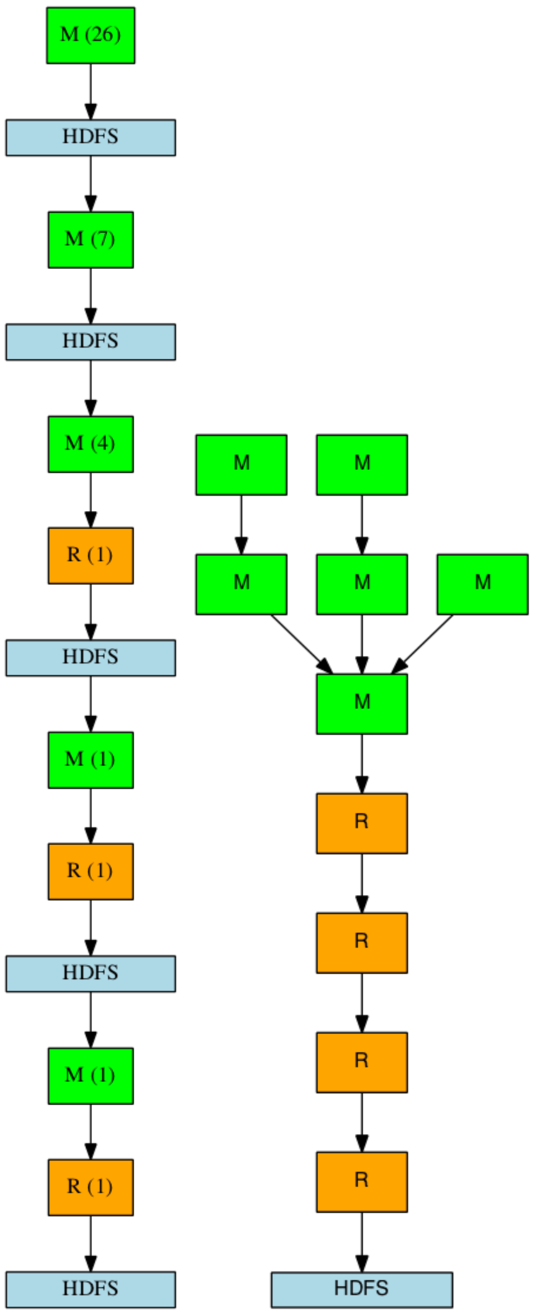
being blocked waiting for the HDFS writing stage to complete.

- Correlation with performance
    - For Hive/MR, this task distribution timeline can be used to explain the slower performance due to “barriers”. As can be seen for query 71, there are significant periods worth tens of seconds where very few tasks are being run.
    - For Hive/Tez, this chart is unable to provide any clues to why performance may be suffering. This is consistent with our earlier observation of the r-squared values between number of tasks and performance for Tez.

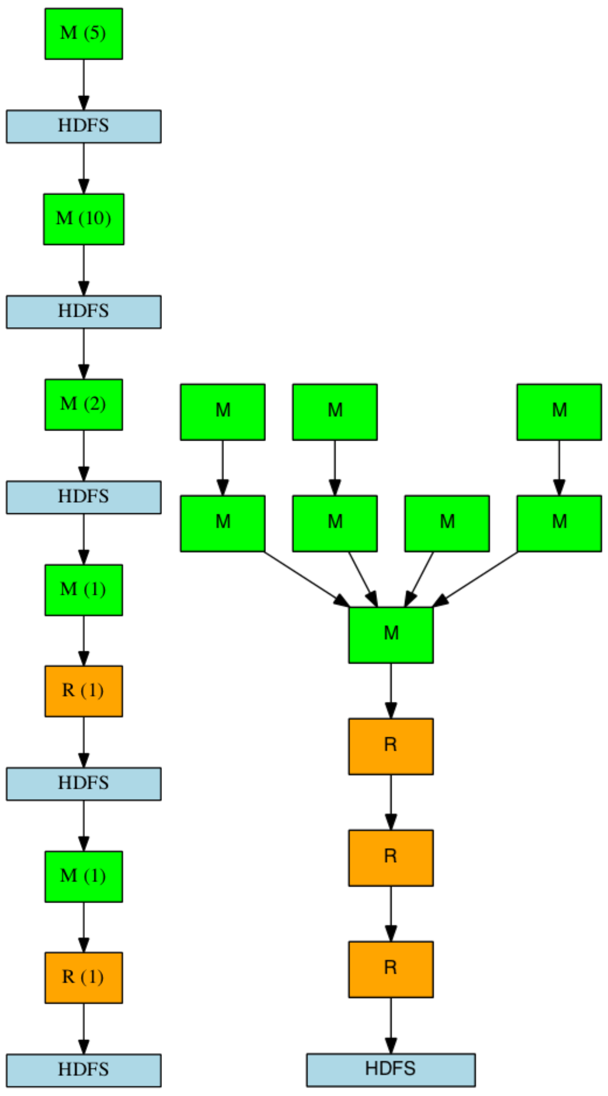
Q1 Part D: DAG analysis

The DAGs for each of the queries in both the engines have been shown below. Each group of DAGs in a different page to avoid tiny images. We have tried to show the number of tasks running in the Map/Reduce phase in MR wherever possible.

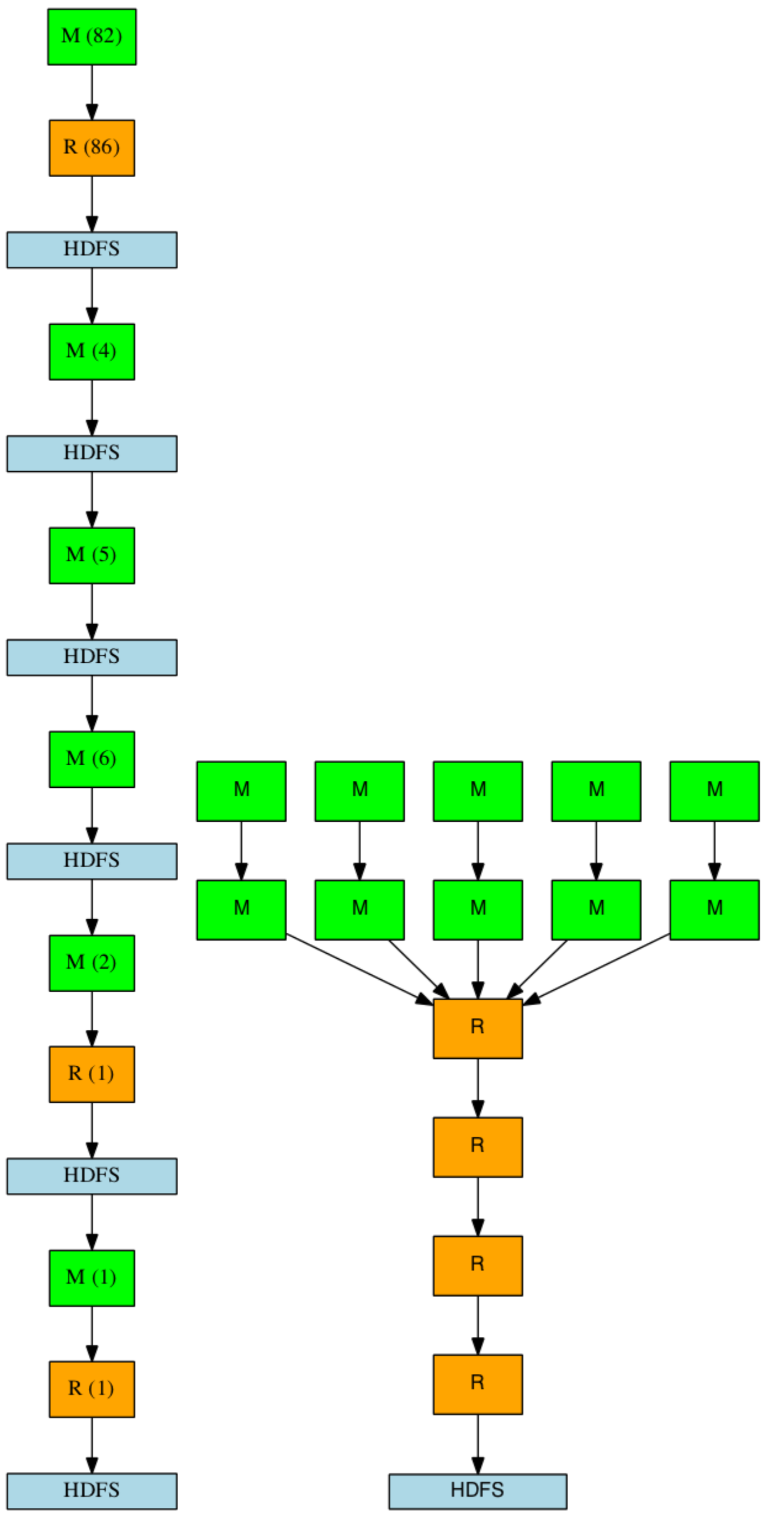
Query 12



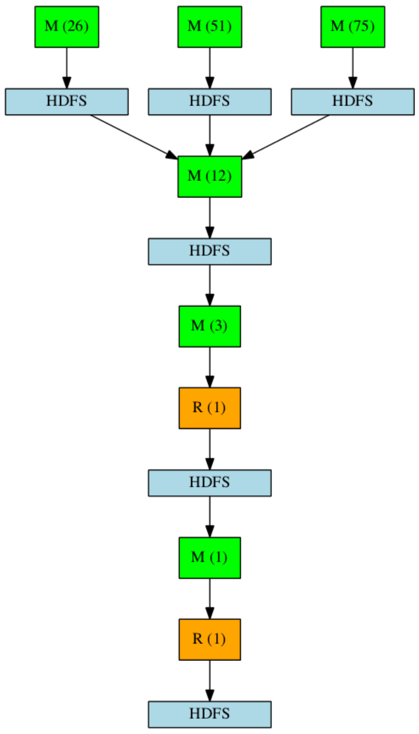
Query 21

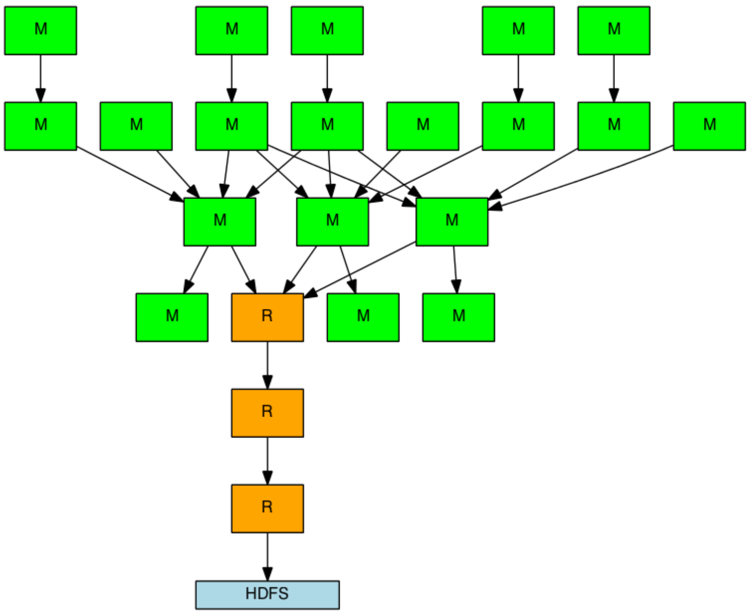

M (5)
HDFS
M (10)
HDFS
M (2)
HDFS
M (1)
R (1)
HDFS
M (1)
R (1)
HDFS
M
M
M
M
M
M
M
M
R
R
R
HDFS

Query 50

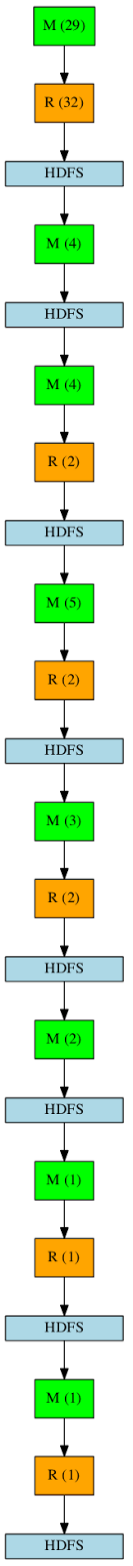

M (82)
R (86)
HDFS
M (4)
HDFS
M (5)
HDFS
M (6)
HDFS
M (2)
R (1)
HDFS
M (1)
R (1)
HDFS
M
M
M
M
M
M
M
M
M
M
R
R
R
R
HDFS

Query 71

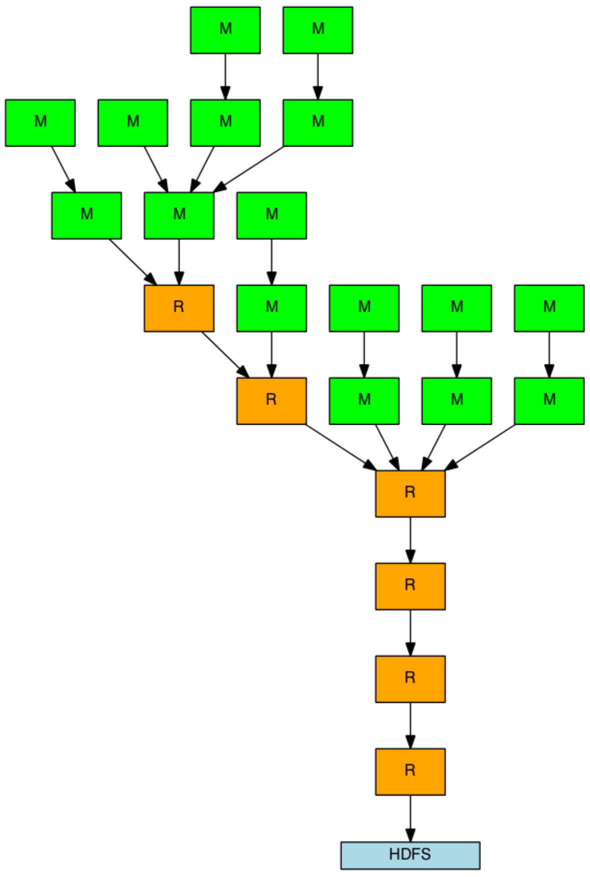

M (26)
M (51)
M (75)
HDFS
HDFS
HDFS
M (12)
HDFS
M (3)
R (1)
HDFS
M (1)
R (1)
HDFS


M
M
M
M
M
M
M
M
M
M
M
M
M
M
M
M
M
M
R
M
M
R
R
HDFS

Query 85



M (29)
R (32)
HDFS
M (4)
HDFS
M (4)
R (2)
HDFS
M (5)
R (2)
HDFS
M (3)
R (2)
HDFS
M (2)
HDFS
M (1)
R (1)
HDFS
M (1)
R (1)
HDFS



M
R
HDFS

Differences in MR and Tez DAGs

We can see the following important differences in the DAG structures for MR and Tez:

- *More HDFS stages in MR* - Each Map-Reduce stage in Hive/MR has to be followed by writing the final output to HDFS, since the subsequent Map tasks can only read their inputs from HDFS. On the other hand, Map tasks in Hive/Tez have no such restriction and can read their inputs from intermediate files.

- *MR DAGs more linear in structure* - Because the Map tasks in Hive/Tez have no restriction on its source of input, the DAG structures in Hive/Tez are much more branched or fanned-out as compared to the Hive/MR DAGs which are more linear in nature.

Impact of DAG structure on performance

The nature of the two DAG structures can impact performance in the following ways:

- *More network bandwidth usage in Hive/MR* - Since there are more stages in Hive/MR where output is being written to HDFS, this leads to more bandwidth utilization in Hive/MR as compared to Hive/Tez because of the need to keep replicas. If the network links are a bottleneck, it will cause the Hive/MR performance to degrade much more than Hive/Tez.

- *Barriers in Hive/MR* - Having more HDFS writing stages also leads to more barriers which can hamper performance because these are periods where very few tasks are run. If the amount of bytes being written to HDFS is quite high, then we will have long periods where very few tasks are running. (Eg. See query 71 task distribution timeline)

- *Less data locality in Hive/Tez* - Since most tasks (including both Map and Reduce) in Tez have to read their input from intermediate files, this leads to lower data locality. In other words, more bytes need to be shuffled in order for the tasks to begin computation.

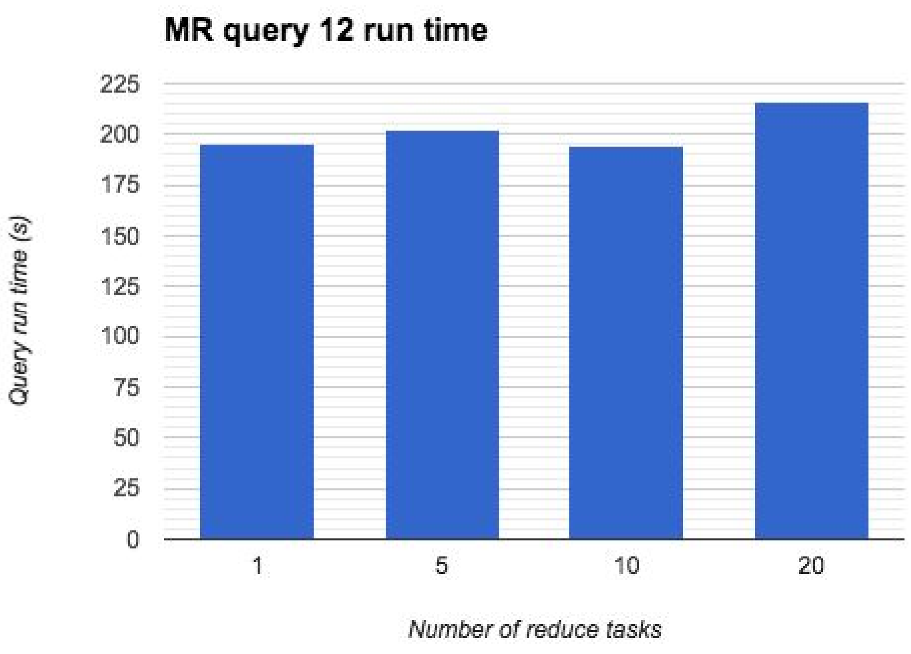
- *Higher probability of hotspots and "incast problem" in Tez* - Another consequence of most tasks taking inputs from intermediate files is the higher probability of the "incast problem" - where certain hotspots arise because multiple nodes are trying to connect to a single source. This problem can lead to high contention for the disk on the single source and thus hamper performance for Tez

Question 2

Q2 Part A: Tuning MR parameters

Tuning the number of reduce tasks

The following chart shows the performance times when we vary the number of reduce tasks for query 12 in Hive/MR.

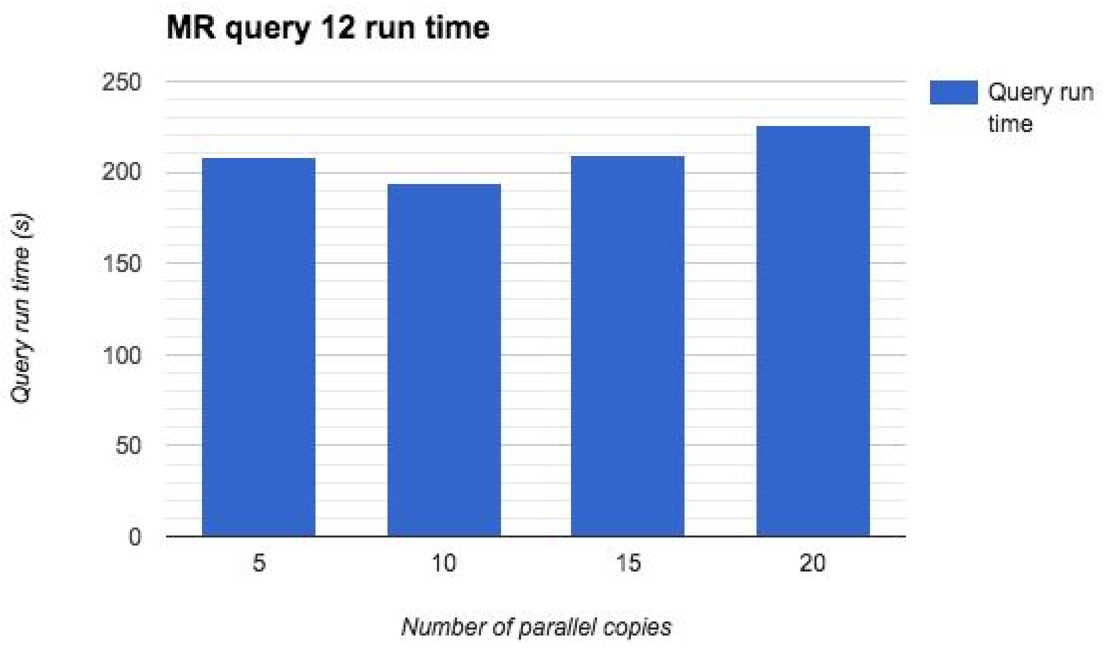


Observations:

- The performance variation is not a lot. The worst case performance (216.4 seconds) is observed when we set the number of reduce tasks to 20. This is still within ~15% of the best case performance of 194.5 seconds with 10 reduce tasks.

  Inordinately increasing the number of reduce tasks does not always boost the performance because of the following possible reasons:
  - There is not much intermediate data to process
  - Reduce operator is not associative

<u>Tuning the number of parallel transfers</u>



Observations:

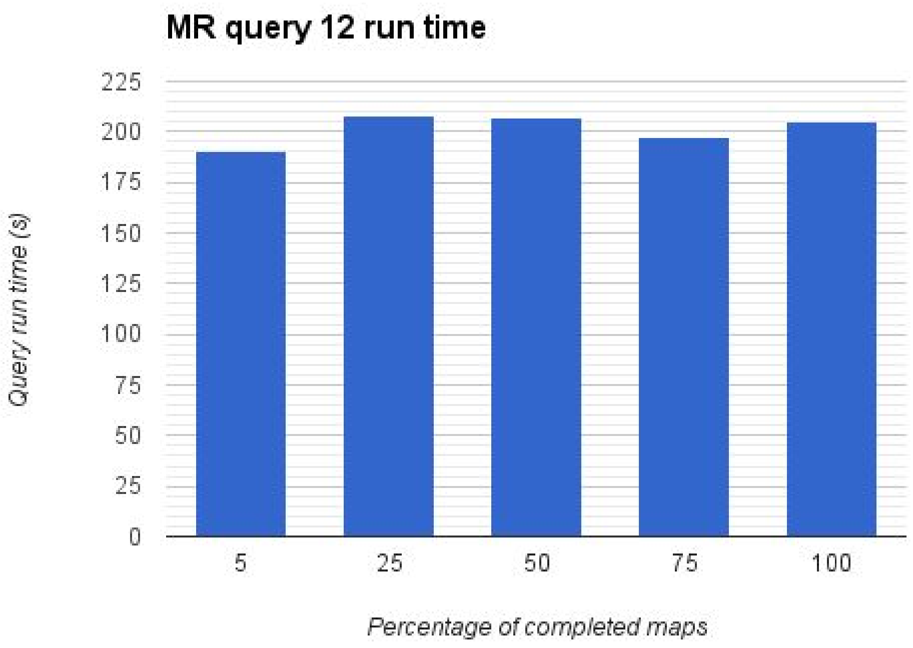
- The performance variation is again not a lot. The worst performance (225.3 seconds) is observed when we have 20 parallel transfers, and the best performance (193.9 seconds) is seen when we use 10 parallel transfers.

- An interesting thing to note here is that increasing the number of parallel transfers from 5 to 10 improves the performance slightly. However increasing it further actually worsens the performance.

  Because of the programming model limitations, MR has several “shuffle phases” where nothing else happens. Since there are no computational tasks running and using the network, we can improve performance by increasing the number of parallel transfers.

  However, having too many parallel transfers results in increased competition between TCP flows. This competition, combined with the congestion avoidance and slow start behaviour of TCP, results in performance degradation for all flows. Thus increasing the number of parallel transfers by too much leads to an increase in the query completion time.

Tuning the fraction of completed map tasks



Observations:

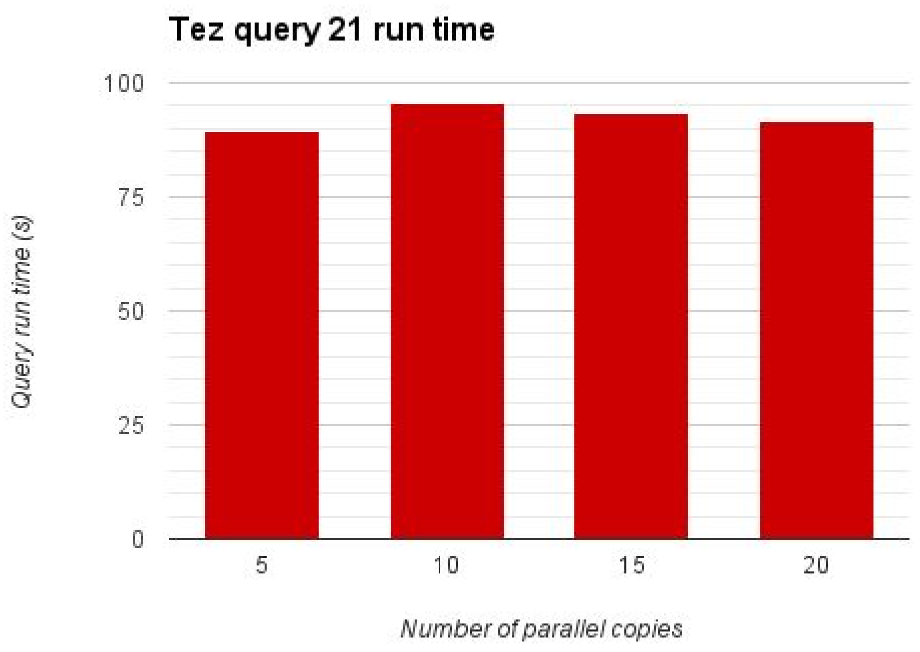
- The worst performance (207.9 seconds) is observed when we set reduce tasks to start when 25% map tasks are complete. Similar run times are seen when we set the fraction to 50% (206.8 seconds) and 100% (205.34 seconds)

Overall observations on tuning

- The best performance seen by tuning various parameters is 190.85 seconds. It is achieved by using the following parameter values:
    - Number of reduce tasks: 10
    - Number of parallel copies: 10
    - Fraction of completed map tasks: 5%

- The best performance observed via tuning is only 3% better than the one seen by using default values in Q1. This suggests that tuning may not help performance in all cases.

Q2 Part B: Tuning Tez parameters



Observations:

- The best performance in Tez for query 21 is 89.6 seconds and is seen when 5 parallel transfers are done during the copy phase.

- The worst performance (95.6 seconds) is seen when we have 10 parallel transfers. Overall, we see an improvement of roughly 17% over the default case running time. However most of the improvement here can be attributed to the decision to reuse containers.

- Increasing the number of parallel transfers does not result in improved performance - in fact we see a decrease of 6-7%. In Tez, there could be multiple “shuffle” phases occurring in parallel at various times. This is different from MR where there is a dedicated “shuffle” period when nothing else happens. In such a scenario, increasing the number of parallel transfers will only lead to more network contention. Multiple TCP flows end up competing with one another. This competition, combined with the TCP congestion avoidance and slow start behaviour leads to degraded performance for all flows.

Q2 Part C: Plugging tuned values into different queries

<u>MR performance comparison</u>

The following table lists the observed run times when queries 12 and 50 were run with the best performing settings for query 21 in MR. The best performing settings for query 12 and 50 are also listed for comparison

	MR Query 12 (using query 21 settings)	MR Query 12 (best performing settings)	MR Query 50 (using query 21 settings)	MR Query 50 (best performing settings)
Number of reduce tasks	10	10	10	10
Number of parallel copies	10	10	10	10
Percentage of completed map tasks	5	50	5	75
Run time	202.003293	182.113307	442.8752291	403.1476531

<u>Observations & Explanations</u>

- Query 12:
    - Applying the best performing settings for query 21 to query 12 doesn't result in the best performance. Using query 21 settings, we see a running time of 202 seconds, while the best performance seen using tuned values is 182.11 seconds.

    - The best performing settings for query 12 are not too different from those for query 21. The only difference is in the fraction of completed map tasks - 50% works best for query 12, while 5% works best for query 21

    - Query 21 has two stages with both Map and Reduce tasks - one stage has 4 map tasks and another stage has 1 map task. On the other hand, query 12 has three such stages - with 2, 4 and 4 map tasks respectively. Trying to run query 21 with a configuration of 5% for `completedmaps` means that in each stage, we will try to schedule a reduce task as soon as 1 map task completes. Whereas using 50% means that we will schedule a reduce task when 1, 2, 2 map tasks finish in each of the stages. 50% gives us better performance than 5% probably because it has the optimal number of reduce tasks running and fetching input from map tasks.

- Query 50:
  - Applying the best performing settings for query 21 to query 50 doesn't result in the best performance. Using query 21 settings, we see a running time of 442.9 seconds, while the best performance seen using tuned values is 403.1 seconds.

  - The best performing settings for query 50 are not too different from those for query 21. The only difference is in the fraction of completed map tasks - 75% works best for query 50, while 5% works best for query 21.

  - Query 50 has three stages with both Map and Reduce tasks with 82, 1 and 4 map tasks respectively. Trying to run query 50 with a configuration of 5% for `completedmaps` means that we would schedule a reduce task when 4, 1 and 1 map tasks have finished in each of the stages. 50% gives us better performance than 5% probably it results in a more balanced Stage-1 where reducers start when 42 map tasks have finished.

<u>Tez Performance Comparison</u>

	Tez Query 12 (using query 21 settings)	Tez Query 12 (best performing settings)	Tez Query 50 (using query 21 settings)	Tez Query 50 (best performing settings)
Number of parallel transfers	5	20	5	20
Run time	110.2787459	103.426755	413.3869631	369.6127379

<u>Observations & Explanations</u>

- Query 12:
  - Applying the best performing settings for query 21 to query 12 doesn't result in the best performance. Using query 21 settings, we see a running time of 110.2 seconds, while the best tuned performance seen is 103.4 seconds.

  - The difference in performance when using query 21 settings and best tuned settings is noticeable - roughly 11% improvement. The best performance is seen when we use 20 parallel transfers.

  - The variation in performance probably stems from the differences in the DAG structures and the number of shuffled bytes. Query 21 probably has a lot more intermediate data being transferred across vertices. Therefore increasing the number of parallel transfers ends up degrading the performance because of increased competition between TCP flows. On the other hand, query 12 probably has less bytes being shuffled. Thus we can increase the number of parallel transfers without degrading the performance of various TCP flows

- Query 50:
    - Applying the best performing settings for query 21 to query 50 doesn't result in the best performance. Using query 21 settings, we see a running time of 413.4 seconds, while the best tuned performance seen is 369.6 seconds.

    - The difference in performance when using query 21 settings and best tuned settings is somewhat small - 7%. The best performance is seen when we use 20 parallel transfers.

    - The variation in performance again probably stems from the differences in the DAG structures and the number of shuffled bytes. Query 50 probably has a lot less intermediate data being generated than Query 21. Therefore increasing the number of parallel transfers for query 21 ends up degrading the performance because of increased competition between TCP flows, and improving performance for query 50 because of higher network utilization.

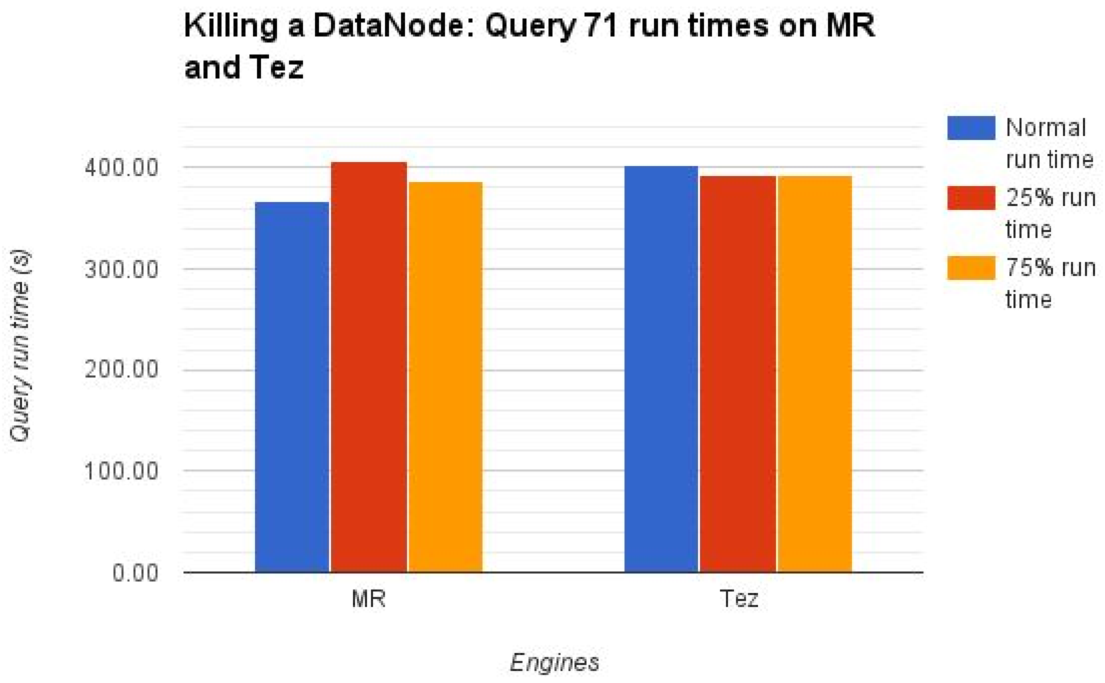
Question 3: Fault Tolerance

The following table and chart show the comparison of query completion times when query 71 is run on Hive/MR and Hive/Tez, when a DataNode process is killed during the middle of query in the following stages:

- DataNode is killed when the query is 25% complete
- DataNode is killed when the query is 75% complete.

We also show the baseline or normal query run time for comparison.

Scenario	MR run time	Tez run time
Normal run time	367.19	402.20
DataNode killed when query was 25% complete	407.22	392.84
DataNode killed when query was 75% complete	387.52	392.28

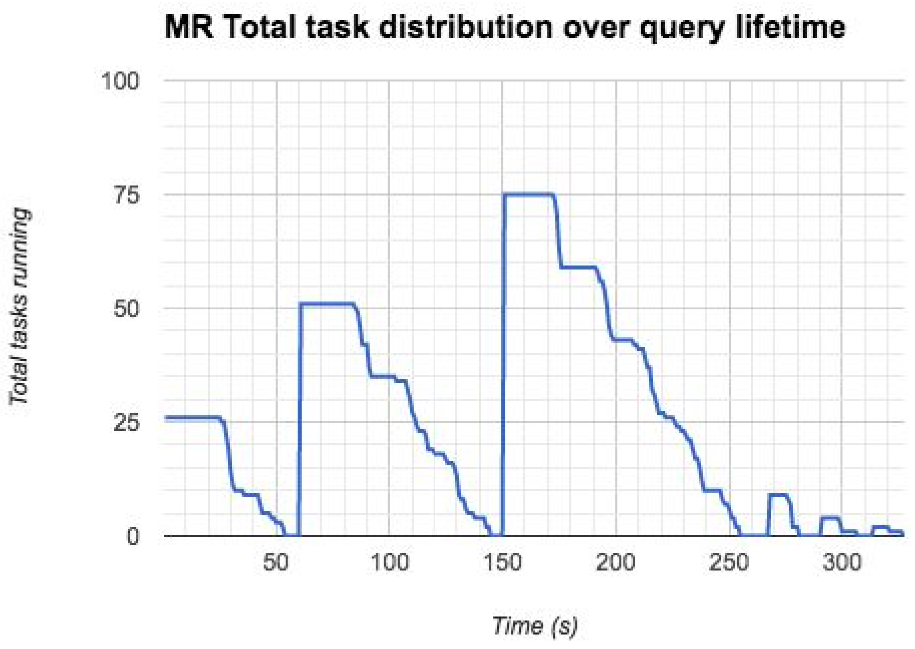


MR Observations

- We see that failing an HDFS datanode while the Map-Reduce query is running increases the completion time of the query. This remains true in both the cases i.e. whether the query is failed after 25% or at 75% completion. This happens primarily because when a datanode goes down, the tasks reading or writing to that node have to re-route their requests to other datanodes. This increases the latency of all the read and write operations which were interrupted by the datanode failure. As a result the query completion time goes up.

- We also see that failing a datanode after 25% completion affects the performance more than failing it at 75%. This can be explained by the fact that for query 71, the number of map tasks running at around 25% completion are much higher than the that at 75% - 51 tasks vs 10 tasks. (See the MR tasks line at t=82 and t=246 in the figure below).

  Since there are more Map tasks running, more HDFS operations are impacted by the datanode failure. This explains the higher completion time for the 25% case over the 75% case. Note that the completion time for both cases is still higher than the baseline.



Tez Observations

- In Tez, we observe that there is hardly any impact on query completion time when a datanode fails.

  This is primarily because a lot of intermediate tasks in Tez fetch most of their inputs from intermediate files which are not in HDFS. Therefore, failing an HDFS datanode barely affects the query completion time.