Qflock

September 2022

September Update

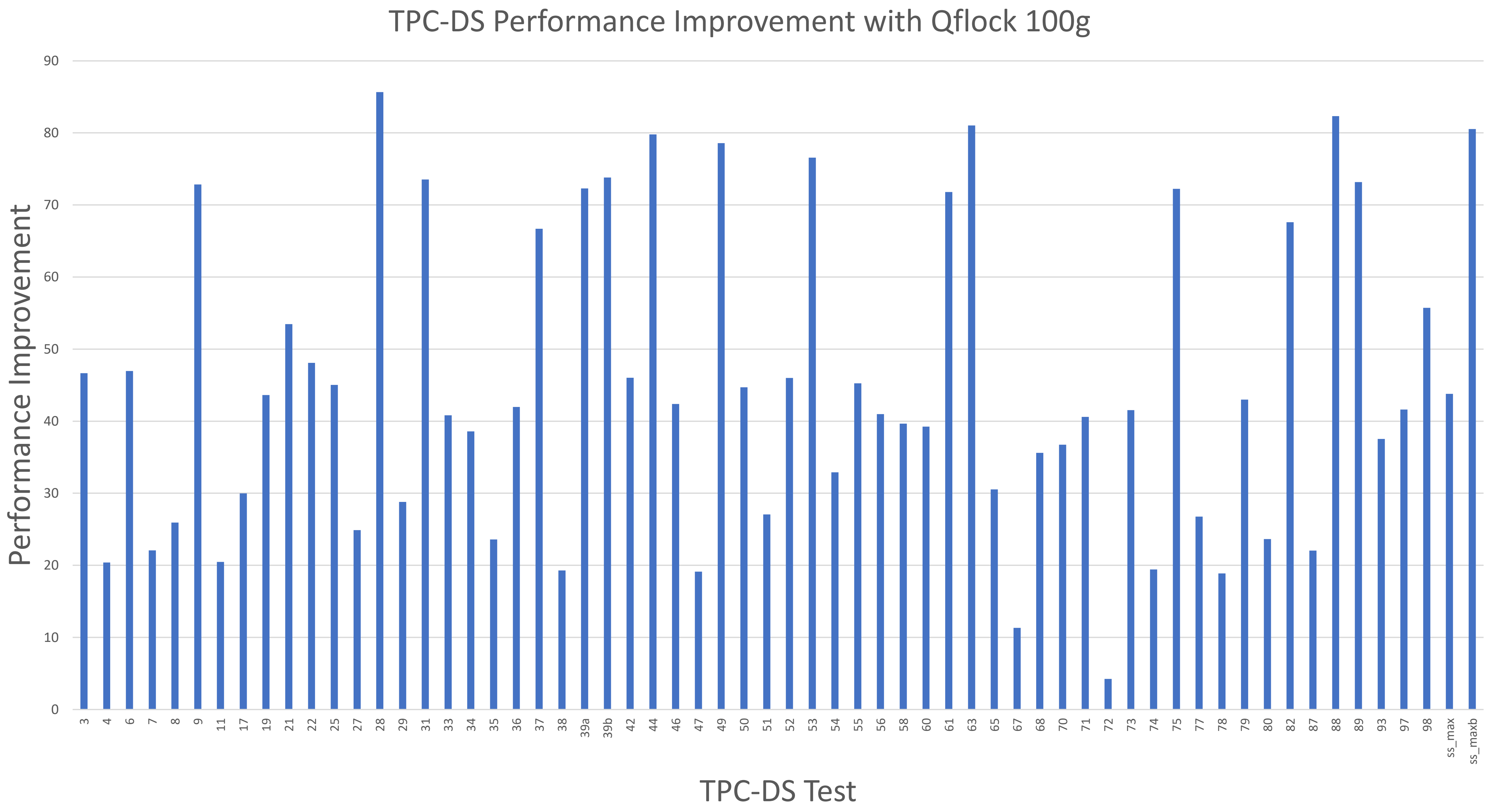
- Join Updates
  - Goal: Decide when to pushdown join operations.
  - Background:
    - Our prior month's work on join pushdown discovered that only some Join operations will benefit from pushdown.
    - Pushing down join when it is not necessary often results in worse performance, since some joins generate more data as output than needs to be read in as part of the join.
    - The purpose of this work was to implement a join pushdown selection criteria in order to enable our Join Pushdown rule to be in effect during the full TPC-DS test run of ~100 complex queries.
  - Result:
    - Our join pushdown selection criteria is implemented and functional under TPC-DS.
    - TPC-DS performance results show that join is not being pushed down when it is not needed.

September Update

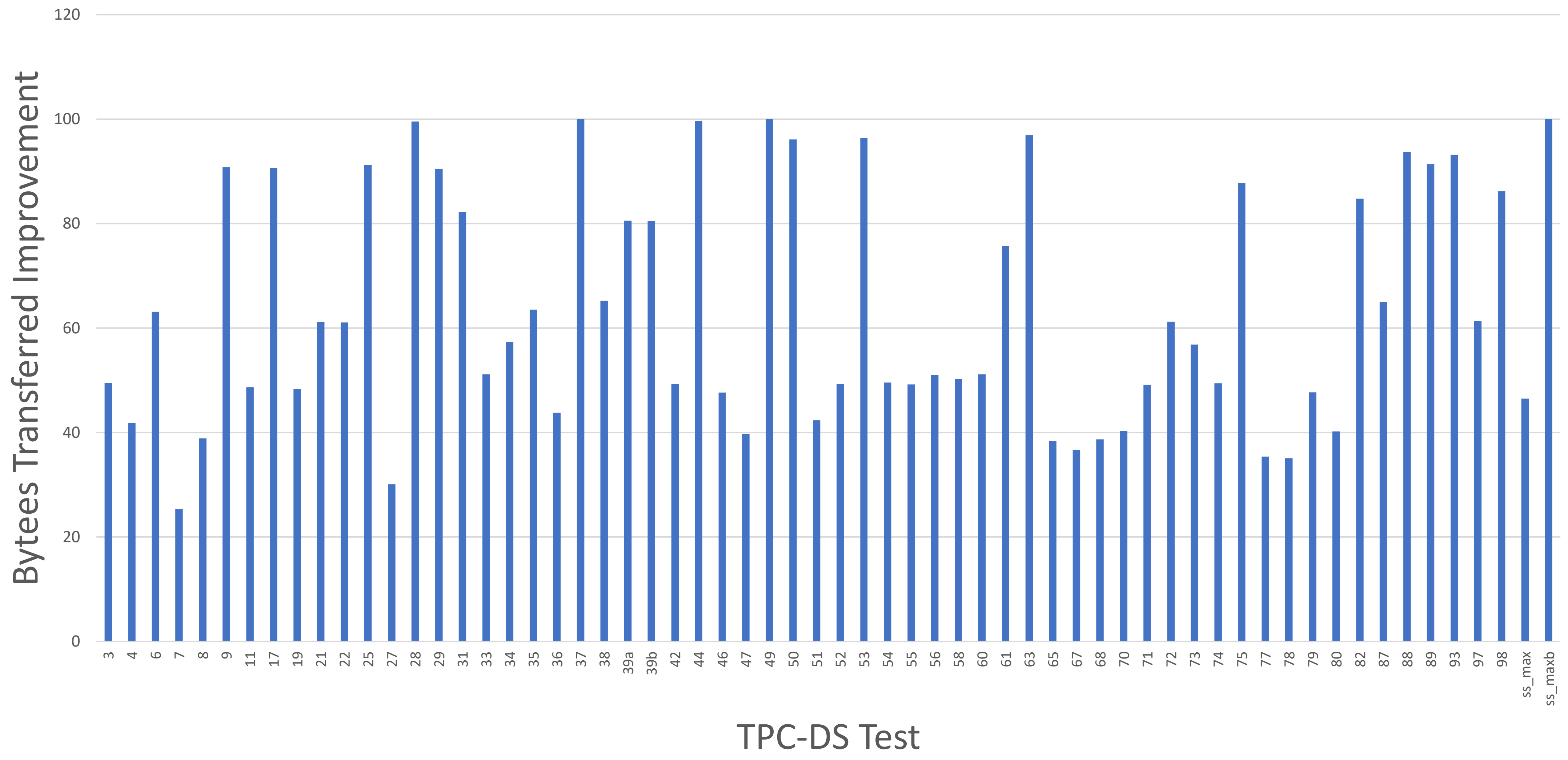
- Pushdown Protocol Updates
  - Goal: Optimize transfer of data from remote server.
    - This was an area of interest from previous research.
    - We are looking to gain maximum data efficiency in the data transfer and provide more optimal use of CPU and pipelining techniques during data transfer.
    - At the same time, our goal was to maintain all the same great improvements that our research showed for filter, aggregate and join.
  - Results
    - New implementation introduces a data transfer mechanism, which allows pipelining of batches of data transfers from the remote server.
    - This mechanism allows Spark to begin processing while data is still being received.
      - Previous implementation did not allow for pipelining on either Server or client side.
      - Previous python implementation also had data copy performance issue especially for large results.
  - Most importantly, our research results still maintains the same great performance improvements we introduced previously when compared to Spark with TPC-DS.
    - Above 20% improvement average across all TPC-DS tests.
    - With many tests in the TPC-DS suites above 50% and even >80% improved.
    - Bytes transferred from the remote datacenter is even better with average improvement above 50%.

September Update

- Query Caching
  - Also employed query caching for this new protocol.
  - We improved on our prior caching implementation.
  - This query caching is designed to provide more optimal memory usage for caching large queries.
  - Queries are cached to disk and read in only when needed.
- Other advantages
  - This approach of saving data to disk instead of memory plays much better with Spark, which performs poorly with higher memory utilization.
  - This also allows for caching of more queries since we are not constrained by memory capacity.
  - This approach will be reused in our future research as we explore more ideas for query caching.


TPC-DS Performance Improvement with Qflock 100g
Performance Improvement
0
10
20
30
40
50
60
70
80
90
3
4
6
7
8
9
11
17
19
21
22
25
27
28
29
31
33
34
35
36
37
38
39a
39b
42
44
46
47
49
50
51
52
53
54
55
56
58
60
61
63
65
67
68
70
71
72
73
74
75
77
78
79
80
82
87
88
89
93
97
98
ss_max
ss_maxb
TPC-DS Test

Remote Data Center Bytes Transferred Improvement with QFlock 100g



References

- Our design specification:
    - https://github.com/open-infrastructure-labs/QFlock/blob/main/doc/qflock_design.pdf

- If you have any questions, problems or suggestions
- Please do not hesitate to create an issue at:
- https://github.com/open-infrastructure-labs/QFlock/issues