


KYLIN-5530 Build Performance Optimization

Optimization effect

01 Index Build - Remove secondary reads and writes of indexes

默认开关: `kylin.engine.aggIndex-adaptive-build-enabled=false`

Test Scenario 1: TPCCH-100

Dataset	Model	Index Num	Build Type	Metadata
TPCH-100	AUTO_MODEL_LINEITEM_8	66	Full	 t100_model_metadata_20528FBF27C088FBCAD04D4

Build result comparison

	Build time/min	Input	Task Num	Shuffle Read	Shuffle Write
Without optimize index build	57	1T	36663	1.7T	1.7T
With optimizing index build	37	560G	30672	1.7T	1.7T

02 Snapshot

Test Scenario 1: TPCCH-100

- Create View V_TPCCH_Q18

```
1 CREATE OR REPLACE view V_TPCCH_Q18 as
2 select
```

```

3      c_name,
4      c_custkey as o_custkey,
5      o_orderkey,
6      o_orderdate,
7      o_totalprice,
8      sum(l_quantity) as sumq
9  from
10     customer,
11     orders,
12     (select
13      l_orderkey,
14      sum(l_quantity) as t_sum_quantity
15  from
16     lineitem
17  where
18     l_orderkey is not null
19  group by
20     l_orderkey) t,
21     lineitem l
22  where
23     c_custkey = o_custkey
24     and o_orderkey = t.l_orderkey
25     and o_orderkey = l.l_orderkey
26  group by
27     c_name,
28     c_custkey,
29     o_orderkey,
30     o_orderdate,
31     o_totalprice
32  order by
33     o_totalprice desc,
34     o_orderdate;

```

- Using SQL modeling

```


1  select
2     sn.n_name,
3     sum(l_extendedprice * (1 - l_discount)) as revenue
4  from
5     lineitem
6     inner join V_TPCH_Q18 on l_orderkey = o_orderkey
7     inner join customer on o_custkey = c_custkey
8     inner join nation cn on c_nationkey = cn.n_nationkey
9     inner join supplier on l_suppkey = s_suppkey
10    inner join nation sn on s_nationkey = sn.n_nationkey

```

```

11 inner join region on sn.n_regionkey = r_regionkey
12 group by
13     sn.n_name
14 order by
15     revenue desc;

```

Dataset	Model	Index Num	Build Type	Metadata
TPCH-100	<u>AUTO_MODEL_LINEITEM_1</u>	2	Full	 bigview_model_metadata_B86E0ED7157A3978552F61

Build result comparison

	Build time/min	Input	Task Num	Shuffle Read	Shuffle Write
Without optimize snapshot build	17.3	87.3G	11686	174.2G	137.2G
With optimizing snapshot build	8.96	49.1G	4668	93.6G	79.8G

- Remove the isEmpty judgment of Snapshot

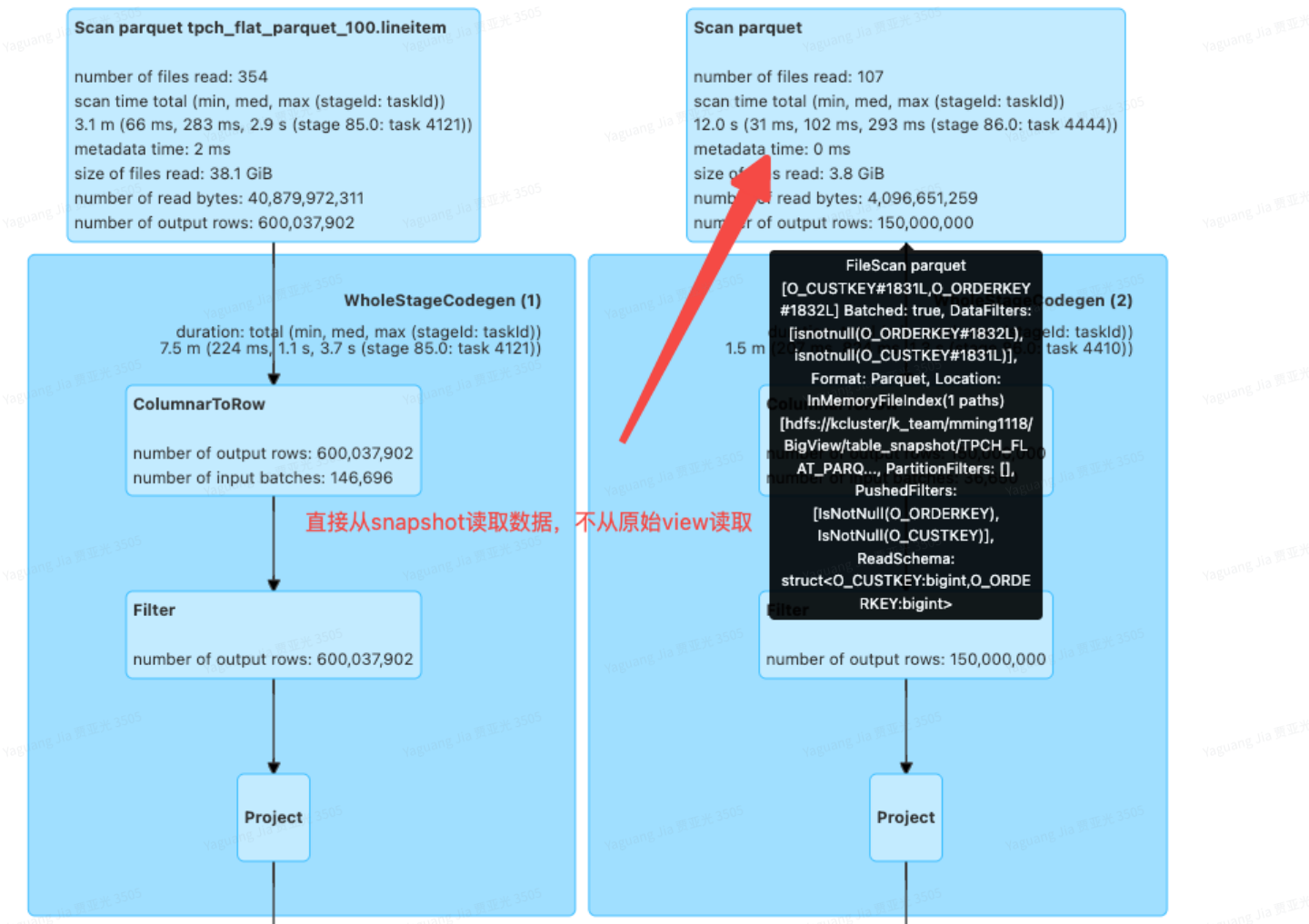
<p>Build table snapshot TPCH_FLAT_PARQUET_100.V_TPCH_Q18.</p> <pre> Build table snapshot TPCH_FLAT_PARQUET_100.V_TPCH_Q18. org.apache.spark.sql.Dataset.isEmpty(Dataset.scala:603) org.apache.kylin.engine.spark.builder.SnapshotBuilder.createSnapshotSize(SnapshotBuilder.scala:438) org.apache.kylin.engine.spark.builder.SnapshotBuilder.buildSnapshotWithoutMd5(SnapshotBuilder.scala:413) org.apache.kylin.engine.spark.builder.SnapshotBuilder.\$anonfun\$executeBuildSnapshot2(SnapshotBuilder.scala:227) scala.concurrent.Future\$.anonfun\$apply\$1(Future.scala:659) scala.util.Success.\$anonfun\$map\$1(Try.scala:255) scala.util.Success.map(Try.scala:213) scala.concurrent.Future.\$anonfun\$map\$1(Future.scala:292) scala.concurrent.impl.Promise.liftedTree\$1(Promise.scala:33) scala.concurrent.impl.Promise.\$anonfun\$transform\$1(Promise.scala:33) scala.concurrent.impl.CallbackRunnable.run(Promise.scala:64) java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149) java.util.concurrent.ThreadPoolExecutor\$Worker.run(ThreadPoolExecutor.java:624) java.lang.Thread.run(Thread.java:748) </pre> <p>去除</p>	2022/11/18 11:14:02	2.0 min	[25] [26] [27] [28] [29] [30] [31] [32]
--	---------------------	---------	---

- Snapshot Capacity Billing Optimization

Mainly when calculating the capacity of snapshot, it is not calculated from the original table, but from the constructed snapshot, otherwise it will lead to repeated calculation when the original table is view

- Join dimension table when flat table builds, read directly from snapshot instead of reading from original table

v the Stage ID and Task ID that corresponds to the max metric



Build flat table + dimension table (snapshot) = 2.5min

Segment 6b59678f-d60f-d8c0-3a41-3a786d3386a9 persist flat table.	+details	2022/11/18 12:57:03	2.5 min	[35] [36] [37] [38] [39] [40] [41] [42] [43] [44]
--	--------------------------	---------------------	---------	---

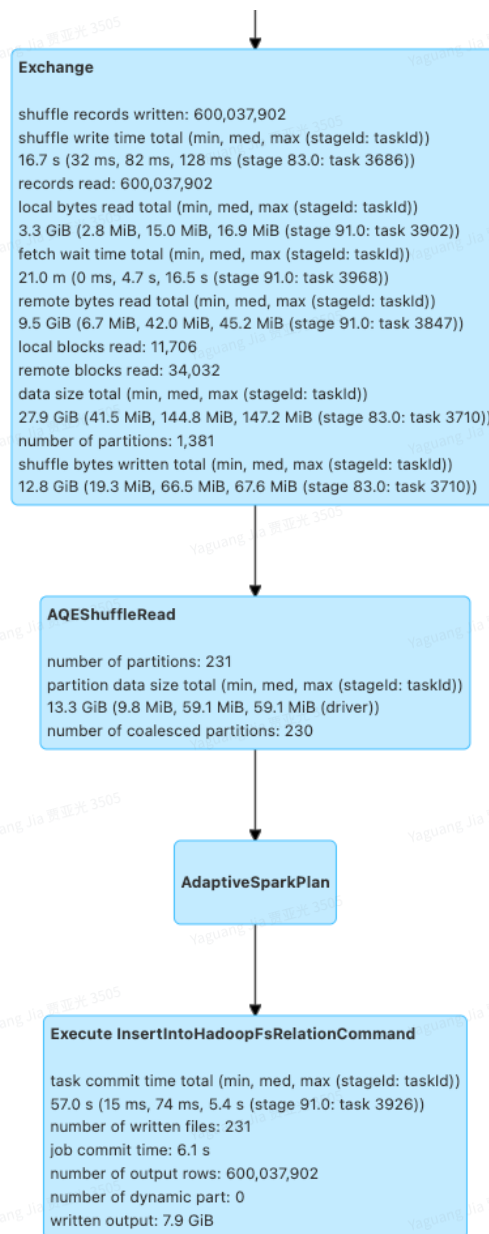
Build flat table + dimension table (original table) = 3.7min

Segment 21dc9dbe-a3c4-b57d-4c67-e558dd54c382 persist flat table.	+details	2022/11/18 11:18:21	3.7 min	[41] [42] [43] [44] [45] [46] [47] [48] [49] [50] [51] [52] [53] [54]
--	--------------------------	---------------------	---------	---

03 Flat Table Redistribution

Switch: `kylin.engine.redistribution-flattable-enabled` = `Flase`

Partition size control: `spark.sql.adaptation.dataWritePartitionSizeInBytes` = 64MB



Comparison of different spark.sql.adaptation.dataWritePartitionSizeInBytes configurations:

	spark.sql.adaptation.dataWritePartitionSizeInBytes	FlatTable Build Time	File Num	File Size
Without Flat-table redistribution	/	6.1min	465	11.3G
Flat-table redistribution	64MB	8.2min	1860	12.7G
Flat-table redistribution	96MB	8.3min	940	12.7G
Flat-table redistribution	128MB	9.3min	627	12.6G
Flat-table redistribution	192MB	8.8min	376	12.6G

