Duration of different operations in invalidateDb (ms)

Database	#Tables	getAllTables	getTableObjectsByName	all other operations	invalidateDb
db40k	40,000	408	170,889	373	171,670
functional	89	2	317	4	323
functional_avro	35	1	153	4	158
functional_avro_def	35	1	134	4	139
functional_avro_snap	54	1	205	5	211
functional_hbase	34	2	149	2	153
functional_kudu	23	1	85	3	89
functional_orc_def	53	2	222	3	227
functional_parquet	70	2	245	5	252
functional_rc	45	2	176	3	181
functional_rc_bzip	42	1	160	4	165
functional_rc_def	42	1	155	5	161
functional_rc_gzip	42	2	165	3	170
functional_rc_snap	43	1	172	4	177
functional_seq	42	2	163	5	170
functional_seq_bzip	42	2	166	3	171
functional_seq_def	42	2	176	3	181
functional_seq_gzip	42	1	155	3	159
functional_seq_record_bzip	42	2	155	4	161
functional_seq_record_def	42	2	211	5	218
functional_seq_record_gzip	42	2	165	3	170
functional_seq_record_snap	42	1	177	3	181
functional_seq_snap	49	2	186	3	191
functional_text_bzip	48	1	175	4	180
functional_text_def	44	2	164	3	169
functional_text_gzip	48	2	189	4	195
functional_text_lzo	44	2	188	3	193
functional_text_snap	44	2	185	4	191
scale_db	1	1	7	4	12
tpcds	24	1	94	3	98
tpcds_orc_def	23	1	88	4	93
tpcds_parquet	23	2	79	4	85
tpcds_seq_snap	23	1	94	4	99
tpch	8	2	34	3	39
tpch_avro	8	1	32	3	36
tpch_avro_snap	8	1	27	4	32
tpch_kudu	8	2	28	3	33
tpch_nested_orc_def	4	1	15	4	20
tpch_nested_parquet	4	1	16	2	19
tpch_orc_def	8	2	28	2	32
tpch_parquet	8	1	29	3	33
tpch_rc	8	1	28	3	32
tpch_seq_gzip	8	1	27	2	30
tpch_seq_snap	8	1	29	2	32
tpch_text_gzip	8	1	30	2	33

Percentage of duration of different operations in invalidateDb

getAllTables	getTableObjectsByName	all other operations	invalidateDb
0.24%	99.55%	0.22%	100.00%
0.62%	98.14%	1.24%	100.00%
0.63%	96.84%	2.53%	100.00%
0.72%	96.40%	2.88%	100.00%
0.47%	97.16%	2.37%	100.00%
1.31%	97.39%	1.31%	100.00%
1.12%	95.51%	3.37%	100.00%
0.88%	97.80%	1.32%	100.00%
0.79%	97.22%	1.98%	100.00%
1.10%	97.24%	1.66%	100.00%
0.61%	96.97%	2.42%	100.00%
0.62%	96.27%	3.11%	100.00%
1.18%	97.06%	1.76%	100.00%
0.56%	97.18%	2.26%	100.00%
1.18%	95.88%	2.94%	100.00%
1.17%	97.08%	1.75%	100.00%
1.10%	97.24%	1.66%	100.00%
0.63%	97.48%	1.89%	100.00%
1.24%	96.27%	2.48%	100.00%
0.92%	96.79%	2.29%	100.00%
1.18%	97.06%	1.76%	100.00%
0.55%	97.79%	1.66%	100.00%
1.05%	97.38%	1.57%	100.00%
0.56%	97.22%	2.22%	100.00%
1.18%	97.04%	1.78%	100.00%
1.03%	96.92%	2.05%	100.00%
1.04%	97.41%	1.55%	100.00%
1.05%	96.86%	2.09%	100.00%
8.33%	58.33%	33.33%	100.00%
1.02%	95.92%	3.06%	100.00%
1.08%	94.62%	4.30%	100.00%
2.35%	92.94%	4.71%	100.00%
1.01%	94.95%	4.04%	100.00%
5.13%	87.18%	7.69%	100.00%
2.78%	88.89%	8.33%	100.00%
3.13%	84.38%	12.50%	100.00%
6.06%	84.85%	9.09%	100.00%
5.00%	75.00%	20.00%	100.00%
5.26%	84.21%	10.53%	100.00%
6.25%	87.50%	6.25%	100.00%
3.03%	87.88%	9.09%	100.00%
3.13%	87.50%	9.38%	100.00%
3.33%	90.00%	6.67%	100.00%
3.13%	90.63%	6.25%	100.00%
3.03%	90.91%	6.06%	100.00%

1. In this experiment, we benchmark the patch for IMPALA-9363 that allows a user to specify the TTableType's to skip when Impala is loading the table metadata from the Hive metastore.
In order to achieve this, instead of calling only getAllTables() in invalidateDb(), we have to additionally call getTableObjectsByName() to retrieve the necessary information to determine the TTableType.

2. To get a more concrete idea about how this patch affects the performance of invalidating metadata, we additionally created 40,000 tables under the database db40k.
For idx from 0 to 39999, the table 'db40k.my_alltypes_idx' was created using
"create table db40k.my_alltypes_idx (id int, bool_col boolean, tinyint_col tinyint, smallint_col smallint, int_col int, bigint_col bigint, float_col float, double_col double,
data_string_col string, string_col string, timestamp_col timestamp, year int, month int) stored as parquet".

3. According to the experimental results, the impact on the performance is huge.
It took the RPC of getTableObjectsByName() 170 seconds to complete, whereas only less than 0.5 second was needed by getAllTables().
Therefore, it takes at least 170 seconds for the Impala minicluster to start.

4. According to the collected results, we found that for most databases, within invalidateDb(), most of the time, only less than 5% of the time was spent on the RPC of getAllTables(), whereas more than 90% of the time was due to getTableObjectsByName().
All other operations include the added logic used to determine the TTableType of a given org.apache.hadoop.hive.metastore.api.Table from the Hive metastore, which are very lightweight compared to getTableObjectsByName().