HiveServer2 with Option to Write Serialized Thrift Object in Final Tasks Performance Test Results

Last Date Revised:
June 20, 2016

Simba Connects

Any Data. Anywhere. Anytime.

Revision History

Rev.	Date	Author	Description
0.1	2016-06-20	ZZ	Initial draft

Introduction

This document details the performance test results for HiveServer2 with the option to write serialized thrift objects in final tasks (JIRA: https://issues.apache.org/jira/browse/HIVE-12049). It covers the test environment used during testing and tests that were run to generate the benchmarks.

Note:

1. All time reported in seconds.
2. No compression algorithm is applied.
3. Iteration time for each test is 10.
4. Open-source JDBC driver is used for testing, the driver was built with HIVE-12049's patch.
5. The tables and graphs only show the execute time, fetch time and total time. Time for prepare statement, get metadata, get column count, get object, close result set and close connection are not included.
6. All test machines are in the same private network in Amazon Elastic Compute Cloud.

Test Environment

The Hive Server with thrift serde for the testing is a three nodes cluster EMR, which has one master node and two slave nodes, and has the following machine specifications:

CPU: 8 cores
Memory:32GB
Hard drive:74G
Bandwidth:1000Mbps

The Hive Server without thrift serde for the testing is a three nodes cluster EMR, which has one master node and two slave nodes, and has the following machine specifications:

CPU: 8 cores
Memory:32GB
Hard drive:74G
Bandwidth:1000Mbps

The client used to run the benchmarking tool has the following machine specifications:

CPU: 2 cores
Memory:8GB
Hard drive:25GB
Bandwidth:450Mbps

Abbreviations

Without Serde: hive server2 before apply thrift serde's patch
Enable Serde: hive server2 with thrift serde's patch, thrift serde enabled
t= #: Using # threads / concurrent connections

Test Sets

Filter1

Query:

```
select * from catalog_returns where cr_item_sk < 20
```

Data set:

catalog_returns
18 INT columns, 1 BIGINT column, 9 FLOAT columns
288491 rows
Result set:
18 INT columns, 1 BIGINT column, 9 FLOAT columns
213 rows
Number of threads/concurrent connections:
1, 2, 8, 16

Filter2

Query:
select * from catalog_returns where cr_item_sk >= 20
Data set:
catalog_returns
18 INT columns, 1 BIGINT column, 9 FLOAT columns
288491 rows
Result set:
18 INT columns, 1 BIGINT column, 9 FLOAT columns
288278 rows
Number of threads/concurrent connections:
1, 2, 8, 16

Groupby1

Query:
select cs_sold_date_sk, count(*) from catalog_sales cs group by cs_sold_date_sk
Data set:
catalog_sales
19 INT columns, 1 BIGINT column, 15 FLOAT columns
2880058 rows
Result set:
2 INT columns
1840 rows
Number of threads/concurrent connections:
1, 2, 8, 16

Groupby2

Query:
select ws_bill_customer_sk, count(*) from web_sales group by ws_bill_customer_sk
Data set:
web_sales
19 INT columns, 1 BIGINT column, 15 FLOAT columns
1438653 rows
Result set:
2 INT columns
81455 rows
Number of threads/concurrent connections:
1, 2, 8, 16

Join1

Query:

select * from catalog_sales cs join date_dim d on d.d_date_sk = cs.cs_sold_date_sk

Data set:

catalog_sales

19 INT columns, 1 BIGINT column, 15 FLOAT columns

2880058 rows

date_dim

16 INT columns, 12 STRING columns

73049 rows

Result set:

2865785 rows

Number of threads/concurrent connections:

1, 2, 8, 16

Join2

Query:

select * from web_sales JOIN customer ON web_sales.ws_bill_customer_sk = customer.c_customer_sk

Data set:

web_sales

19 INT columns, 1 BIGINT column, 15 FLOAT columns

1438653 rows

customer

9 INT columns, 9 STRING columns

144000 rows

Result set:

1438929 rows

Number of threads/concurrent connections:

1, 2, 8, 16

Orderby1

Query:

select * from catalog_sales cs order by cs_sold_date_sk

Data set:

catalog_sales

19 INT columns, 1 BIGINT column, 15 FLOAT columns

2880058 rows

Number of threads/concurrent connections:

1, 2, 8, 16

Orderby2

Query:

select * from web_sales order by ws_bill_customer_sk

Data set:

web_sales

19 INT columns, 1 BIGINT column, 15 FLOAT columns

1438653 rows

Number of threads/concurrent connection:

1, 2, 8, 16

Select1

Query:

select * from catalog_sales

Data set:

catalog_sales

19 INT columns, 1 BIGINT column, 15 FLOAT columns

2880058 rows

Number of threads/concurrent connections:

1, 2, 8, 16

Select2

Query:

select * from web_sales

Data set:

web_sales

19 INT columns, 1 BIGINT column, 15 FLOAT columns

1438653 rows

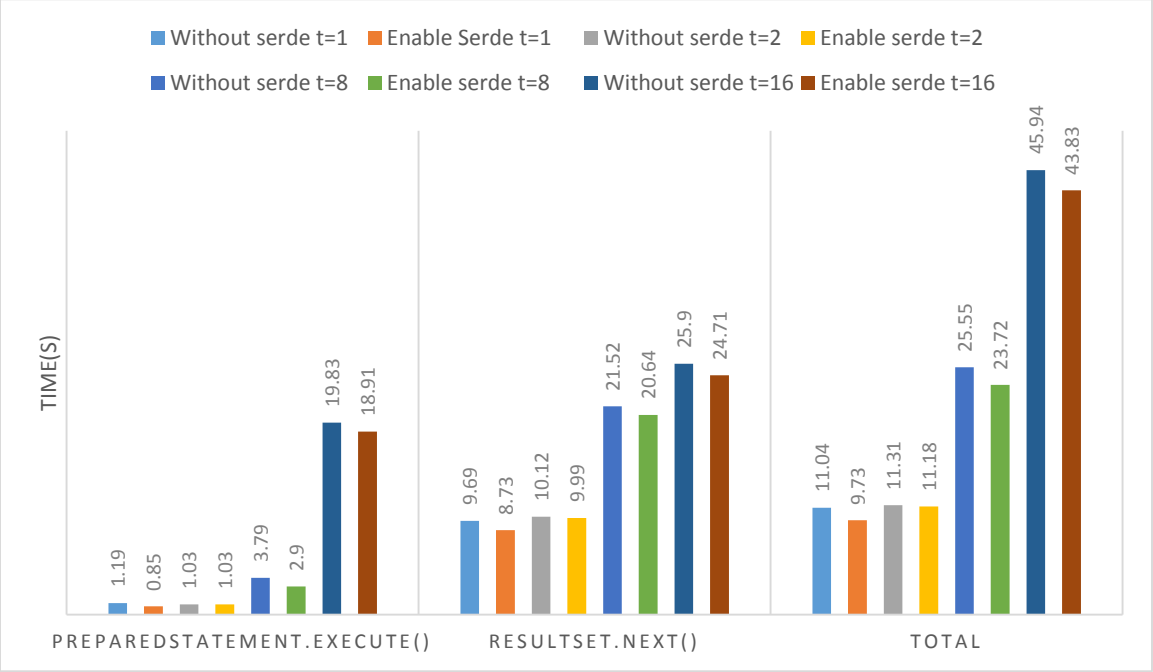
Number of threads/concurrent connections:

1, 2, 8, 16

Test Results

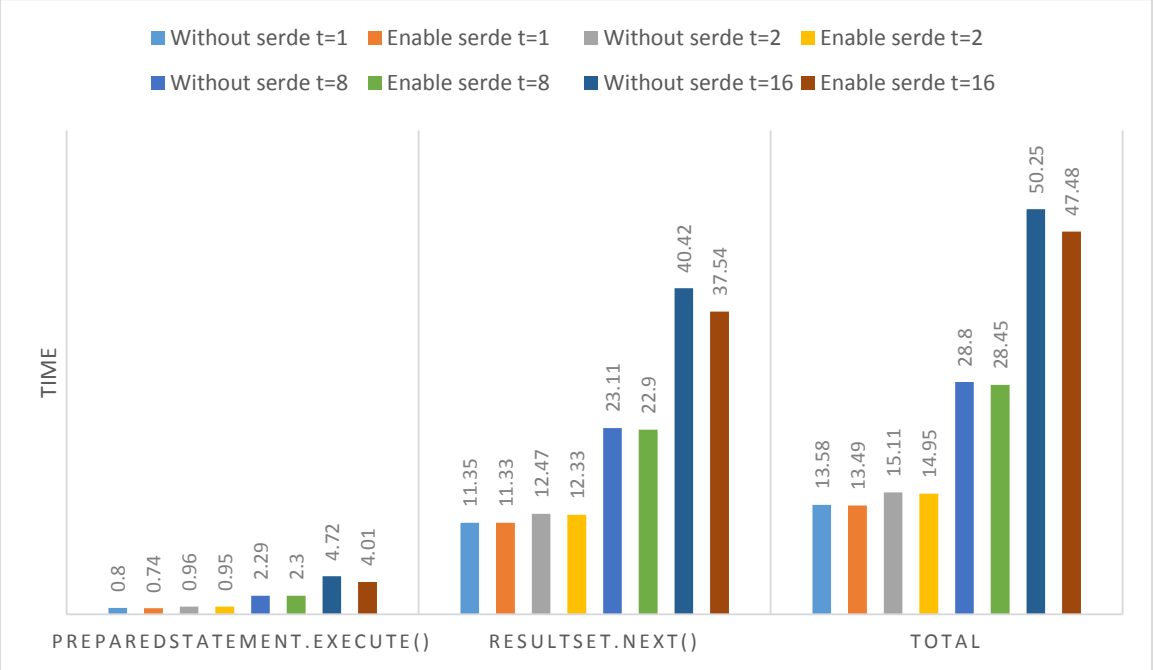
Filter1

	# of Threads	PreparedStatement.execute()	ResultSet.next()	TOTAL
Without serde	1	1.19	9.69	11.04
Enable serde	1	0.85	8.73	9.73
Without serde	2	1.03	10.12	11.31
Enable serde	2	1.03	9.99	11.18
Without serde	8	3.79	21.52	25.55
Enable serde	8	2.9	20.64	23.72
Without serde	16	19.83	25.9	45.94
Enable serde	16	18.91	24.71	43.83



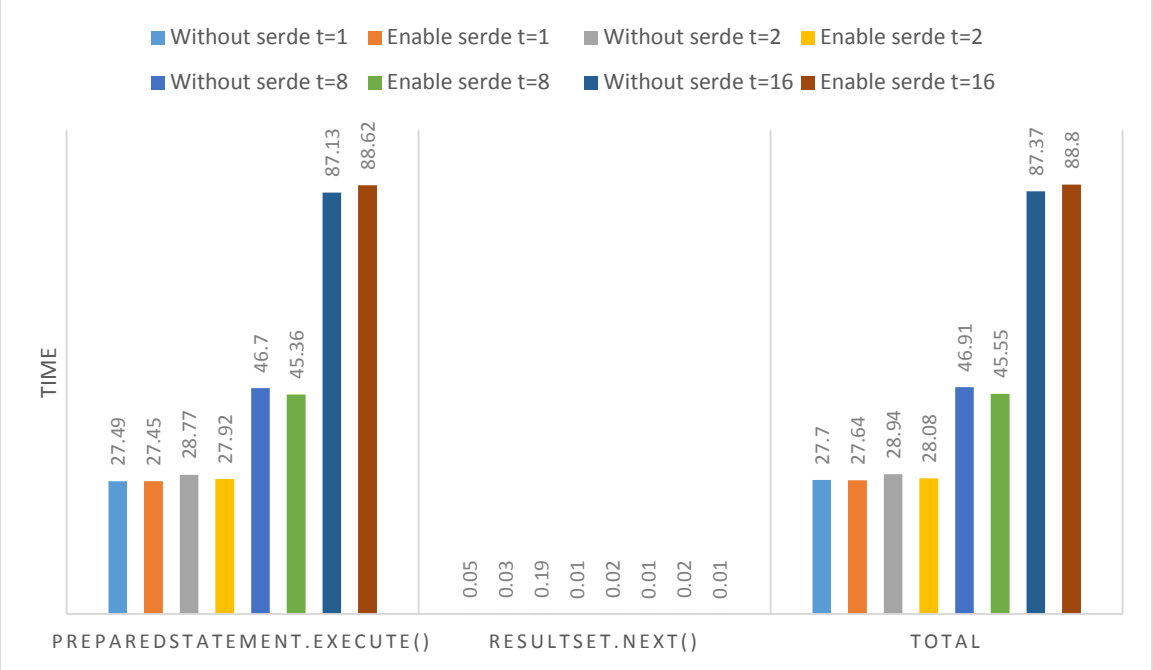
Filter2

	# of Threads	PreparedStatement.execute()	ResultSet.next()	TOTAL
Without serde	1	0.8	11.35	13.58
Enable serde	1	0.74	11.33	13.49
Without serde	2	0.96	12.47	15.11
Enable serde	2	0.95	12.33	14.95
Without serde	8	2.29	23.11	28.8
Enable serde	8	2.3	22.9	28.45
Without serde	16	4.72	40.42	50.25
Enable serde	16	4.01	37.54	47.48



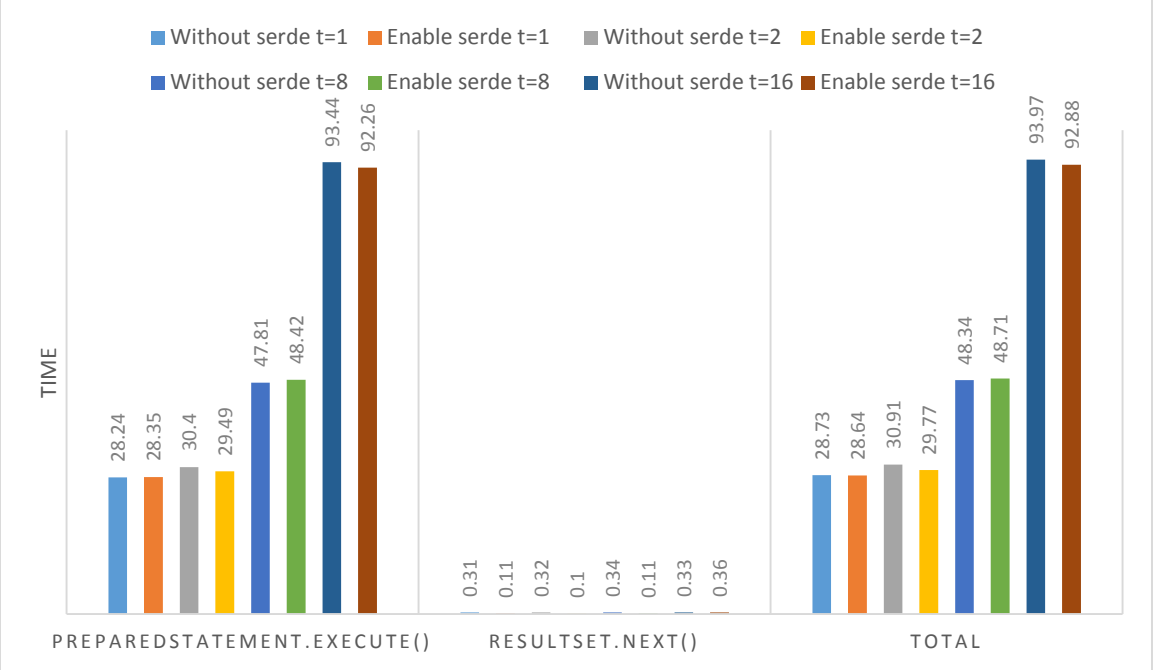
Groupby1

	# of Threads	PreparedStatement.execute()	ResultSet.next()	TOTAL
Without serde	1	27.49	0.05	27.7
Enable serde	1	27.45	0.03	27.64
Without serde	2	28.77	0.19	28.94
Enable serde	2	27.92	0.01	28.08
Without serde	8	46.7	0.02	46.91
Enable serde	8	45.36	0.01	45.55
Without serde	16	87.13	0.02	87.37
Enable serde	16	88.62	0.01	88.8



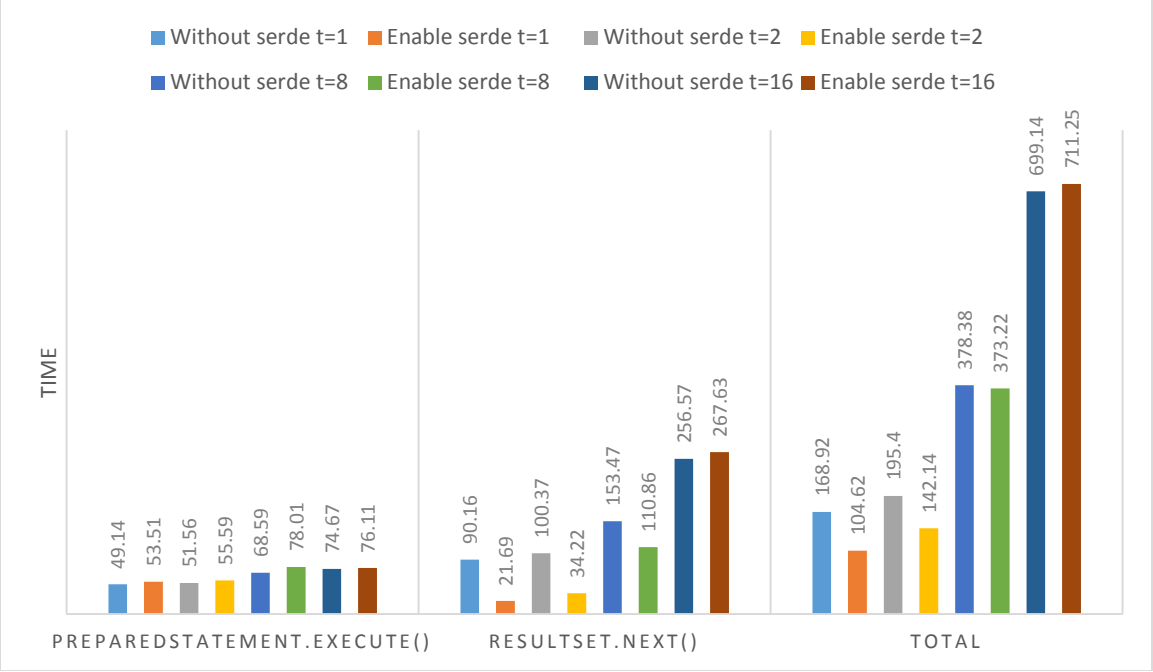
Groupby2

	# of Threads	PreparedStatement.execute()	ResultSet.next()	TOTAL
Without serde	1	28.24	0.31	28.73
Enable serde	1	28.35	0.11	28.64
Without serde	2	30.4	0.32	30.91
Enable serde	2	29.49	0.1	29.77
Without serde	8	47.81	0.34	48.34
Enable serde	8	48.42	0.11	48.71
Without serde	16	93.44	0.33	93.97
Enable serde	16	92.26	0.36	92.88



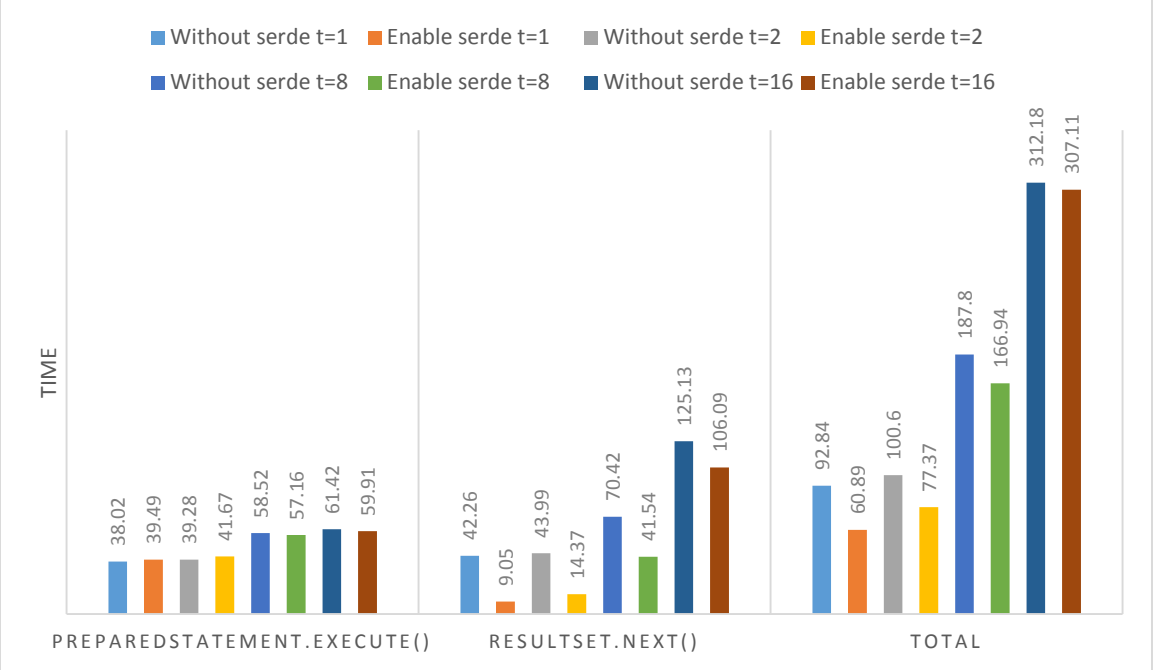
Join1

	# of Threads	PreparedStatement.execute()	ResultSet.next()	TOTAL
Without serde	1	49.14	90.16	168.92
Enable serde	1	53.51	21.69	104.62
Without serde	2	51.56	100.37	195.4
Enable serde	2	55.59	34.22	142.14
Without serde	8	68.59	153.47	378.38
Enable serde	8	78.01	110.86	373.22
Without serde	16	74.67	256.57	699.14
Enable serde	16	76.11	267.63	711.25



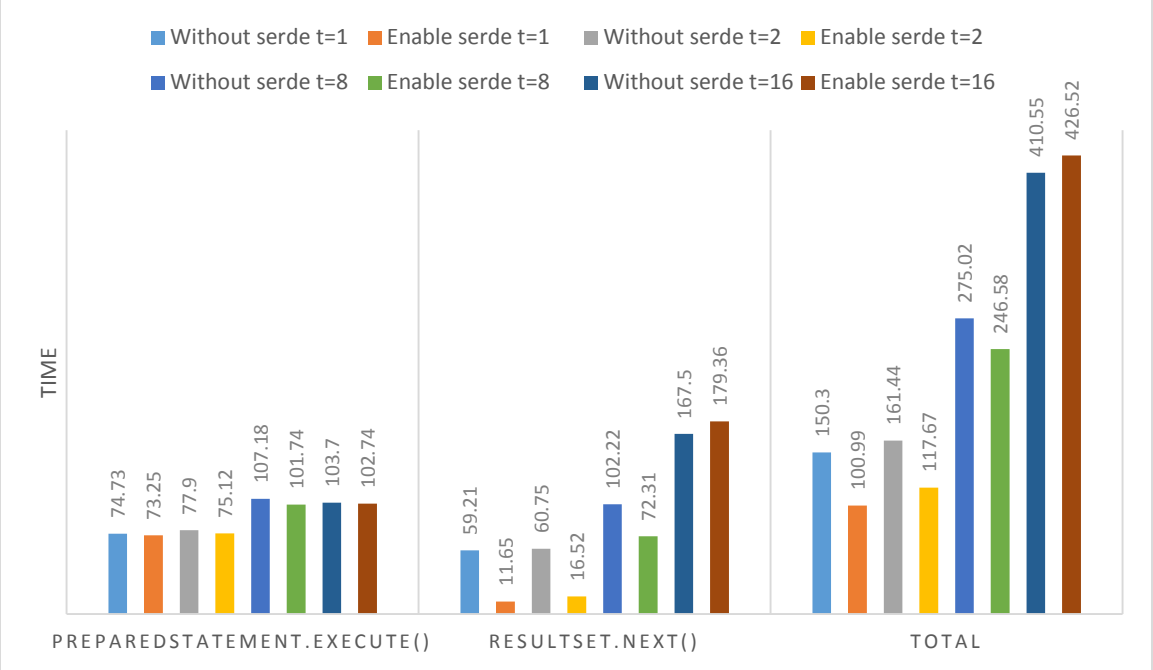
Join2

	# of Threads	PreparedStatement.execute()	ResultSet.next()	TOTAL
Without serde	1	38.02	42.26	92.84
Enable serde	1	39.49	9.05	60.89
Without serde	2	39.28	43.99	100.6
Enable serde	2	41.67	14.37	77.37
Without serde	8	58.52	70.42	187.8
Enable serde	8	57.16	41.54	166.94
Without serde	16	61.42	125.13	312.18
Enable serde	16	59.91	106.09	307.11



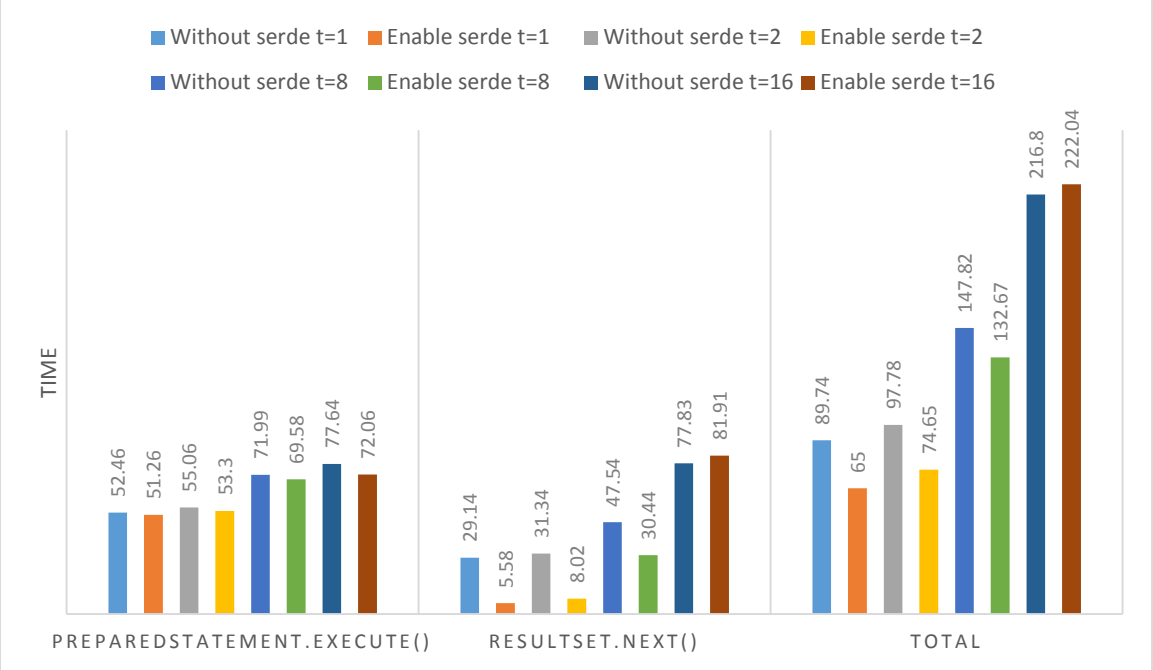
Orderby1

	# of Threads	PreparedStatement.execute()	ResultSet.next()	TOTAL
Without serde	1	74.73	59.21	150.3
Enable serde	1	73.25	11.65	100.99
Without serde	2	77.9	60.75	161.44
Enable serde	2	75.12	16.52	117.67
Without serde	8	107.18	102.22	275.02
Enable serde	8	101.74	72.31	246.58
Without serde	16	103.7	167.5	410.55
Enable serde	16	102.74	179.36	426.52



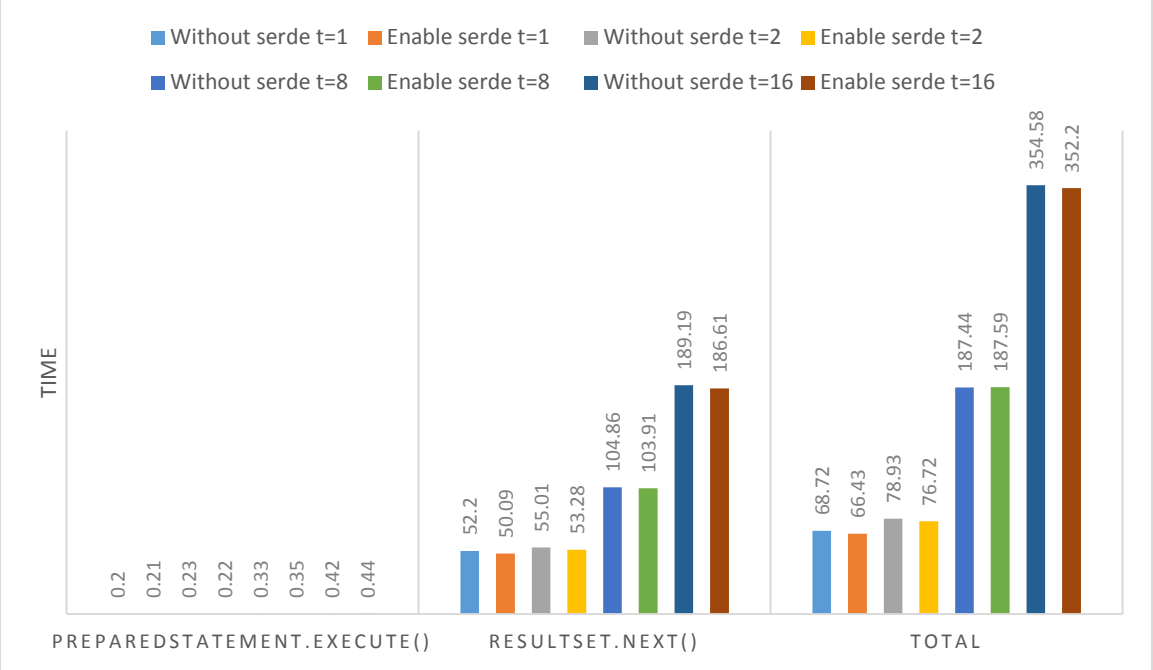
Orderby2

	# of Threads	PreparedStatement.execute()	ResultSet.next()	TOTAL
Without serde	1	52.46	29.14	89.74
Enable serde	1	51.26	5.58	65
Without serde	2	55.06	31.34	97.78
Enable serde	2	53.3	8.02	74.65
Without serde	8	71.99	47.54	147.82
Enable serde	8	69.58	30.44	132.67
Without serde	16	77.64	77.83	216.8
Enable serde	16	72.06	81.91	222.04



Select1

	# of Threads	PreparedStatement.execute()	ResultSet.next()	TOTAL
Without serde	1	0.2	52.2	68.72
Enable serde	1	0.21	50.09	66.43
Without serde	2	0.23	55.01	78.93
Enable serde	2	0.22	53.28	76.72
Without serde	8	0.33	104.86	187.44
Enable serde	8	0.35	103.91	187.59
Without serde	16	0.42	189.19	354.58
Enable serde	16	0.44	186.61	352.2



Select2

	# of Threads	PreparedStatement.execute()	ResultSet.next()	TOTAL
Without serde	1	0.19	29.78	38.09
Enable serde	1	0.21	29.24	37.57
Without serde	2	0.26	32.02	44.13
Enable serde	2	0.24	31.39	43.45
Without serde	8	0.34	57.39	97.04
Enable serde	8	0.31	56.35	96.52
Without serde	16	0.87	96.35	178.57
Enable serde	16	0.42	96.13	178.1