Evaluation of SQL benchmark for distributed in-memory Database Management Systems

Oleg Borisenko[†] **and David Badalyan**[††]

Ivannikov Institute for System Programming of the RAS, Moscow, Russia

Summary

Requirements for modern DBMS in terms of speed are growing every day. As an alternative to traditional relational DBMS distributed, in-memory DBMS are proposed. In this paper, we investigate capabilities of Apache Ignite and VoltDB from the point of view of relational operations and compare them to PostgreSQL using our implementation of TPC-H like workload.

Key words:

Apache Ignite, VoltDB, PostgreSQL, In-memory Computing, distributed systems.

1. Introduction

Today, most applications have to handle large amounts of data. Architects of complex systems face a problem of choosing a DBMS corresponding to reliability, scalability and usability requirements. Traditionally used RDBMS have several advantages, such as integrity control, data consistency, matureness. However, the centralized data placement and classic disk storage limit performance scalability to vertical scalability model (extending single-node performance) or some kind of sharding which may not be applicable for particular application.

As an alternative to relational DBMS, in-memory technology was introduced in the last decade. DBMS of this class store the entire database in RAM, thereby increasing bandwidth for data access and decreasing latency. Also, these systems are often have distributed nature, which allows to scale systems horizontally across multiple nodes.

This paper is focused on comparison of in-memory Apache Ignite and VoltDB to PostgreSQL in terms of SQL operations. Using the TPC-H benchmark approach, described in detail in [1] [2] we implement our own workload similar to TPC-H. The following problems are investigated:

- DBMS performance comparison;
- correctness of the benchmark query results in the distributed operation mode;
- performance impact of increasing the number of nodes in a cluster.

Results are compared to the same tests performed against PostgreSQL DBMS.

Manuscript received October 5, 2018
Manuscript revised October 20, 2018

2. Overview

This chapter discusses the features of Apache Ignite and VoltDB. Both DBMS support data replication in distributed mode. However, the paper will consider only the case of data distribution without replication.

2.1 Apache Ignite

Apache Ignite is a distributed in-memory DBMS [3] [4] written in the Java programming language. It is a caching and data processing platform designed for managing large amounts of data using large number of compute nodes. Despite original key-value nature of the system, the developers declare ACID compliance and full support for the SQL:1999 standard.

H2 Database is used as a subsystem to process SQL queries. After initial query processing it generates local data retrieval requests on the nodes containing the necessary data, and a global data collection request. To eliminate a possibility of results loss, Apache Ignite implements a special mode in which local query execution is accompanied by polling other cluster nodes for the presence of necessary data.

Apache Ignite has two cluster modes: atomic and transactional. In atomic mode, Apache Ignite is an AP system, in transactional mode - CP in terms of CAP theorem (P - Partition tolerance, C - Consistency, A - Availability [5]). In this paper, Apache Ignite is evaluated in transactional mode exclusively.

2.2 VoltDB

VoltDB is a distributed relational in-memory DBMS [6] [7] representing the newSQL DBMS class. In non-replicated mode of operation, VoltDB distributes the data using hashes of the selected table column values; the hashes are used to determine which node should get the data.

VoltDB is also declared to be ACID compliant and supports SQL queries. An interface is provided for stored procedures written in the Java programming language for SQL queries execution [8]. All declared procedures are stored on each cluster node and it's one of the reasons for system performance gain.

3. Benchmark and raw results

Testing was performed on cluster configurations of 1, 2, 4, and 8 computing nodes with 1, 10, 50, and 100 GB as data payload. Each node in the cluster is a virtual machine with 92GB of RAM, 3 processor cores at 2.1 GHz (Intel Xeon Gold 6152) running under Xen hypervisor and each virtual machine has dedicated physical server-class SSDs.
Benchmark consists of 22 queries comprising a TPC-H like workload. Legend:

- Each cell contains query execution time in seconds.
- Q1-Q22 are ids of the queries in terms of TPC-H description.
- "Err" means that an error occurred in DMBS engine during the test.
- T — test was running too long.
- TF — test failed with database creation error due to memory insufficiency.

You can see the results in tables 1, 2, 3, 4 (Appendix A).

4. Errors and restrictions discovered

4.1 Apache Ignite

- Requests Q8 and Q19, when the polling mode of other nodes of the cluster is turned on for the presence of data necessary for the correct execution of the request, end with errors
  - Q8: `java.sql.SQLException: General error: "java.lang.ArrayIndexOutOfBoundsException" [50000-195]`
  - Q19: `java.sql.SQLException: javax.cache.CacheException: Failed distributed join query: join condition does not [joinedCache = SQL_PUBLIC_PART, plan = ... (query code)`
- Results of queries Q2, 4, 16, 17, 18, 20, 21, 22 are incorrect in distributed mode with replication disabled. These requests process only the local data of each node, aggregating the results, despite the activation of a mode that prevents such behavior. This happens due to the presence of subqueries in the WHERE query section. This problem is mentioned in the official Apache Ignite documentation [9].
- Q11, 13, 15 requests contain operations not supported by Apache Ignite
- The interval data type, as well as SQL operations CREATE TYPE and CREATE VIEW are not supported in Apache Ignite

4.2 VoltDB

- During the execution of the queries Q2, 3, 4, 5, 7, 8, 9, 10, 12, 14, 16, 17, 18, 19, 20, 21, 22 in distributed mode with replication disabled VoltDB scheduler returns the following errors:
  - Q2, 4, 7, 8, 9, 16, 17, 18, 20, 21, 22: java.sql.SQLException: General Provider Error (GRACEFUL_FAILURE): Unexpected Ad Hoc Planning Error: java.lang.RuntimeException: Error Compiling query: org.voltdb.planner.PlanningErrorException: Subquery expressions referencing only replicated tables.
  - Q3, 5, 10, 12, 14, 19: java.sql.SQLException: General Provider Error (GRACEFUL_FAILURE): "Unexpected Ad Hoc Planning Error: java.lang.RuntimeException: Error compiling query: org.voltdb.planner.PlanningErrorException: This query is not plannable. Planner cannot guarantee that it's a single partition.
- Q11, Q15 queries contain operations that are not supported by VoltDB.
- The char data type as well as the SQL operations CREATE TYPE and CREATE VIEW are not supported by VoltDB.

5. Results analysis

The test results show that Apache Ignite's performance increases with the number of nodes in the cluster. Also, analyzing the results obtained, it is possible to identify two patterns in temporal results changes effected by the scaling up of the system:

- The query execution time always reduces as the number of computing nodes increases. Examples of such queries are: Q1, 4, 5, 6, 12, 14, 21
- The query execution time increases after scaling up from one computing node to two and decreases with further scaling of the system. Examples of such queries: Q2, 3, 7, 9, 10, 16, 17

Based on the results, it can be seen that VoltDB performance stops growing after a certain number of nodes in the cluster, despite further scalability.
The results show that when you run queries on a single node, VoltDB is ahead of Apache Ignite in almost all queries except Q7, 8, 9, 17. At the same time, VoltDB succeeds to complete only Q1 and Q6 queries in distributed mode.

These requests in VoltDB also work faster than Apache Ignite at the same cluster and data configuration. It is also worth noting that in VoltDB there was no more performance gain after the number of cluster nodes increased from four to eight. In the case of Apache Ignite, there are noticeably fewer errors when scaling a cluster, and an increase in performance is linear to the cluster node count.

Comparing the Apache Ignite to PostgreSQL, we can say that performance gain becomes visible only for large amounts of data in several requests. Query time Q2, Q4, Q17 and Q21 turned out to be 1.78, 6.35, 1.41, 2.68 times less than similar results in PostgreSQL, respectively, during launches per 100 GB of data, with the number of Apache Ignite cluster nodes equal to eight. These requests also showed the best time relative to PostgreSQL, with a similar Apache Ignite cluster configuration, during launches of 50 GB of data, but this gain was not as significant as in the case of 100 GB of data. The remaining requests were either significantly slower than PostgreSQL, or there was a slight increase in various variations of the Apache Ignite cluster configuration.

Even though VoltDB was faster than Apache Ignite in most cases, it still has results inferior to PostgreSQL on data volumes of 1 and 10 GB with a single VoltDB node. An exception is Q18 query, the execution time of which turned out to be 3.74 times less than the same time in PostgreSQL with a data volume of 10 GB. Considering large amounts of data and distributed mode of VoltDB cluster, we come to the conclusion that comparison is possible only in queries Q1 and Q6. These requests are faster than PostgreSQL in all cases.

6. Conclusion

In this paper, we investigated Apache Ignite and VoltDB in-memory DBMS using TPC-H like benchmark guidelines. The sources for benchmark process are available at https://github.com/ispras/Apache-Ignite-and-Voltdb-TPCH-Implementation.

Despite better results of VoltDB relative to Apache Ignite in most requests and faster execution of some requests on large amounts of data compared to PostgreSQL, VoltDB does not support the execution of most types of queries in distributed mode. Performance of VoltDB does not scale linearly with cluster growth.

Compared to this, Apache Ignite showed a linear increase in performance with an increase in the number of computing nodes, a noticeably smaller number of errors, and better performance relative to PostgreSQL in some queries on large amounts of data.

Both systems do not meet SQL:1999 standard.

Acknowledgments

This work is funded by the Minobrnauki Russia (grant id RFMEFI60417X0199, grant number 14.604.21.0199).

References

[1] Meikel Poess and Chris Floyd. New TPC Benchmarks for Decision Support and Web Commerce. SIGMOD Rec. 29, 4, pp. 64-71, DOI=http://dx.doi.org/10.1145/369275.369291
[2] TPC-H, an ad-hoc, decision support benchmark. URL: http://www.tpc.org/tpch/ (date of the application: 12.08.2018)
[3] Official Apache Ignite documentation. URL: https://apacheignite-sql.readme.io/docs (date of the application: 12.08.2018)
[4] Zheludkov, Michael, and Timur Isachenko. High Performance in-memory computing with Apache Ignite. Lulu. com, 2017.
[5] Simon, Salomé. "Brewer’s CAP theorem." CS341 Distributed Information Systems, University of Basel (HS2012) (2000).
[6] VoltDB official website. URL: https://www.voltdb.com/ (date of the application: 12.08.2018)
[7] Stonebraker and Weisberg. "The VoltDB Main Memory DBMS." IEEE Data Eng. Bull. 36.2 (2013): 21-27.
[8] Buckle S. Introduction to VoltDB. URL: https://www.ibm.com/developerworks/library/os-voltdb/ (date of the application: 12.08.2018)
[9] Official Apache Ignite documentation. URL: https://apacheignite-sql.readme.io/docs/how-ignite-sql-works#section-known-limitations (date of the application: 12.08.2018)

Oleg Borisenko was born in Russia in 1988. Received the specialist degree at Moscow State University, Faculty of Computational Mathematics and Cybernetics in 2011. Works at Ivannikov Institute for System Programming of the Russian Academy of Sciences. Zones of interest: cloud infrastructure problems, distributed systems.

David Badalyan was born in Yerevan, Armenia in 1998. He is a student at the Bauman Moscow State University (BMSTU), Computer Science and Control Systems department. Works at Ivannikov Institute for System Programming of the Russian Academy of Sciences. Zones of interest: parallel computations, distributed systems

Appendix A.

Table 1. 1GB dataset

1GB	**PostgreSQL**		**Ignite/nodes**				**VoltDB/nodes**			
	no index	**indexed**	**1**	**2**	**4**	**8**	**1**	**2**	**4**	**8**
Q1	3.63	3.33	20.33	10.24	5.37	3.012	7	3.2	1.9	1.95
Q2	1.4	0.78	2.58	4.36	2.18	0.82	0.4	Err	Err	Err
Q3	1.1	0.9	4.85	19.24	7.84	4	2.19	Err	Err	Err
Q4	0.37	0.38	2.64	1.29	0.6	0.37	1.69	Err	Err	Err
Q5	0.79	0.68	1923.6	37.94	14.7	7.79	2.27	Err	Err	Err
Q6	0.54	0.51	9.44	4.28	2	1.21	0.84	0.39	0.24	0.23
Q7	1.32	0.8	28.4	39.36	16.14	8.6	738.16	Err	Err	Err
Q8	1.27	0.32	12	Err	Err	Err	39.9	Err	Err	Err
Q9	4.73	1.52	32.94	63.7	26.33	15.22	52.76	Err	Err	Err
Q10	0.85	0.83	3.69	11.29	4.88	2.7	1.75	Err	Err	Err
Q12	0.77	0.79	9.18	5.12	2.62	1.42	0.68	Err	Err	Err
Q14	0.86	2.6	10.43	6	2.85	1.63	0.51	Err	Err	Err
Q16	1.8	1.7	649.55	1347.77	347.3	142.83	1.14	Err	Err	Err
Q17	T	6	0.3	26.9	10.37	5.16	5.66	Err	Err	Err
Q18	14.5	14.49	T	T	T	T	2.77	Err	Err	Err
Q19	0.76	0.07	T	Err	Err	Err	T	Err	Err	Err
Q20	T	0.31	T	T	T	3355.51	1.19	Err	Err	Err
Q21	2.46	2.17	105	50.4	20.42	8.78	7.2	Err	Err	Err
Q22	1.72	0.14	T	123.33	32.03	12.66	T	Err	Err	Err

Table 2. 10GB dataset

10GB	**PostgreSQL**		**Ignite/nodes**				**VoltDB/nodes**			
	no index	**indexed**	**1**	**2**	**4**	**8**	**1**	**2**	**4**	**8**
Q1	32.4	32.6	208.2	103.18	51.66	30.09	69.55	31.51	18.3	18.59
Q2	14	8.13	28.08	40	16.2	9.48	4.3	Err	Err	Err
Q3	30.7	31.14	51.37	200.3	80.06	41.57	21.75	Err	Err	Err
Q4	5	4.8	24.53	11.84	5.85	3.15	16.5	Err	Err	Err
Q5	8.5	7.3	T	T	T	75	26.46	Err	Err	Err
Q6	4.6	4.5	91.33	44.6	22.22	12.75	8.42	3.84	2.18	2.24
Q7	18.5	17.2	287.31	408.34	164.42	83.97	T	Err	Err	Err
Q8	28.9	3.36	132.45	Err	Err	Err	659.67	Err	Err	Err
Q9	51.2	19.3	362.53	642.15	268.59	146.53	T	Err	Err	Err
Q10	10.2	10.2	32.5	121.22	55.93	33.22	18.33	Err	Err	Err
Q12	6.5	6.75	100.85	51.34	26.35	14.96	6.84	Err	Err	Err
Q14	7.75	52.9	110.43	57.49	29.57	16.52	5.41	Err	Err	Err
Q16	16.6	17.6	T	T	T	T	18.4	Err	Err	Err
Q17	T	3.2	2.45	298.02	111.16	53.31	96.52	Err	Err	Err
Q18	102	103.6	T	T	T	T	27.21	Err	Err	Err
Q19	6.5	0.69	T	Err	Err	Err	T	Err	Err	Err
Q20	T	13	T	T	T	T	12.24	Err	Err	Err
Q21	123.7	26.7	1070.4	501.85	208.8	97.14	70.28	Err	Err	Err
Q22	19	1.6	T	T	T	T	T	Err	Err	Err

Table 3. 50GB dataset

50GB	PostgreSQL		Ignite/nodes		VoltDb/nodes	
	no index	indexed	4	8	4	8
Q1	147.09	161.88	266.9	153.81	94	95.33
Q2	75.25	62.08	80.76	38.8	Err	Err
Q3	165.76	169.23	421.65	209.73	Err	Err
Q4	20.34	23.43	28.6	17.4	Err	Err
Q5	186.49	189.07	T	T	Err	Err
Q6	20.81	20.94	117.08	66.45	11.02	11.2
Q7	183.82	177.61	848.52	410.18	Err	Err
Q8	152.56	40.69	Err	Err	Err	Err
Q9	377.57	232.37	1445.82	743.32	Err	Err
Q10	102.79	120.41	289.39	277.55	Err	Err
Q12	34.57	36.26	135.54	78.33	Err	Err
Q14	35.91	35.47	150.81	88.05	Err	Err
Q16	85.78	99.67	T	T	Err	Err
Q17	T	305.62	578.7	266.64	Err	Err
Q18	652.59	498.06	T	T	Err	Err
Q19	31.22	11.43	Err	Err	Err	Err
Q20	T	255.11	T	T	Err	Err
Q21	875.19	839.08	1062.8	477.02	Err	Err
Q22	93.37	10.68	T	T	Err	Err

Table 4. 100GB dataset

100GB	PostgreSQL		Ignite/nodes	VoltDB/nodes
	no index	indexed	8	8
Q1	307.8	364.05	304.4	TF
Q2	T	155.05	86.84	TF
Q3	376.5	398.78	452.8	TF
Q4	243.43	224.3	35.3	TF
Q5	441.6	482.4	T	TF
Q6	148.3	146.18	134.23	TF
Q7	434	439.46	822.75	TF
Q8	318.8	340.89	Error	TF
Q9	763.15	1652.3	1527.94	TF
Q10	250.3	289.01	723.08	TF
Q12	154.55	187.84	155.55	TF
Q14	170.2	184.95	176.24	TF
Q16	174.7	177.66	T	TF
Q17	T	751.65	537.2	TF
Q18	1422.7	1420.15	T	TF
Q19	148.9	59.11	Error	TF
Q20	T	2560.5	T	TF
Q21	2700.21	2608.82	971.49	TF
Q22	194.9	62.23	T	TF