NewSQL新型分布式数据库研究报告

Wen Zhenglin

2016-9-28

概要

1. 问题
2. NewSQL相关比较
3. CockroachDB
4. 相关测试结果

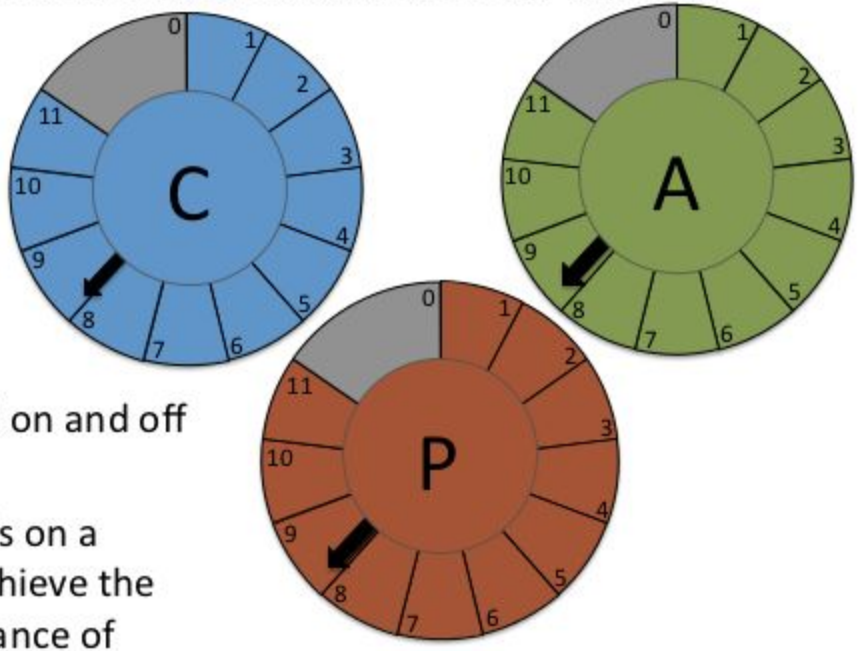
问题

传统数据库不支持水平扩展，带来性能的局限性，NoSQL数据库具备水平扩展，但却不具备ACID事务交易、数据的强一致性、SQL语句的支持等功能，进而给开发带来昂贵的代价。

NewSQL新型分布式数据库旨在解决以上问题，通过结合传统数据库的功能和NoSQL的性能优点，实现高可扩展性和强一致性数据库，支持ACID事务交易，无需在应用层做一致性的控制和事务的处理，简化开发人员的负担，使投入更多精力解决核心业务问题。

关于CAP [1]

- Either way, in the face of the evidence, what does CAP Theorem tell us?
- It's not pick two
- Or even pick one
- C, A, and P are not switches to be turned on and off
- Think off them as dials on a dashboard used to achieve the most appropriate balance of consistency, availability and partition tolerance for specific workloads



NewSQL的定义[2]

"A DBMS that delivers the scalability and flexibility promised by NoSQL while retaining the support for SQL queries and/or ACID, or to improve performance for appropriate workloads."

451 Group

NewSQL的定义[2]（续）

"SQL as the primary interface

ACID support for transactions

Non-locking concurrency control

High per-node performance

Scalable, shared nothing architecture"

Michael Stonebraker

NewSQL数据库类别

1. 新架构设计
2. 中间件技术
3. 数据库在线服务

NewSQL Systems[3]

		Year Released	Main Memory Storage	Partitioning	Concurrency Control	Replication	Summary
NEW ARCHITECTURES	**Clustrix** [6]	2006	No	Yes	MVCC+2PL	Strong+Passive	MySQL-compatible DBMS that supports shared-nothing, distributed execution.
	CockroachDB [7]	2014	No	Yes	MVCC	Strong+Passive	Built on top of distributed key/value store. Uses software hybrid clocks for WAN replication.
	Google Spanner [24]	2012	No	Yes	MVCC+2PL	Strong+Passive	WAN-replicated, shared-nothing DBMS that uses special hardware for timestamp generation.
	H-Store [8]	2007	Yes	Yes	TO	Strong+Active	Single-threaded execution engines per partition. Optimized for stored procedures.
	HyPer [9]	2010	Yes	Yes	MVCC	Strong+Passive	HTAP DBMS that uses query compilation and memory efficient indexes.
	MemSQL [11]	2012	Yes	Yes	MVCC	Strong+Passive	Distributed, shared-nothing DBMS using compiled queries. Supports MySQL wire protocol.
	NuoDB [14]	2013	Yes	Yes	MVCC	Strong+Passive	Split architecture with multiple in-memory executor nodes and a single shared storage node.
	SAP HANA [55]	2010	Yes	Yes	MVCC	Strong+Passive	Hybrid storage (rows + cols). Amalgamation of previous TREX, P*TIME, and MaxDB systems.
	VoltDB [17]	2008	Yes	Yes	TO	Strong+Active	Single-threaded execution engines per partition. Supports streaming operators.

NewSQL Systems[3]（续）

MIDDLEWARE	**AgilData** [1]	2007	No	Yes	MVCC+2PL	Strong+Passive	Shared-nothing database sharding over single-node MySQL instances.
	MariaDB MaxScale [10]	2015	No	Yes	MVCC+2PL	Strong+Passive	Query router that supports custom SQL rewriting. Relies on MySQL Cluster for coordination.
	ScaleArc [15]	2009	No	Yes	Mixed	Strong+Passive	Rule-based query router for MySQL, SQL Server, and Oracle.
DBAAS	**Amazon Aurora** [3]	2014	No	No	MVCC	Strong+Passive	Custom log-structured MySQL engine for RDS.
	ClearDB [5]	2010	No	No	MVCC+2PL	Strong+Active	Centralized router that mirrors a single-node MySQL instance in multiple data centers.

现有解决方案

商业数据库：Voltdb,Clustrix,SAP hana

开源数据库：CockroachDB,Tidb (TiKV compile failed)

CockroachDB vs TiDB

	CockroachDB	TiDB
数据库类型	NewSQL	NewSQL
实现方式	新架构设计	新架构设计
基于	Spanner[4]+F1[5]	TiDB(F1)+TiKV(Spanner)
开发语言	Go	TiDB(Go)+TiKV(Rust)+PD(Go)
许可证	Apache2.0	Apache2.0
开发人员	Google前员工发起	国内人员发起
开发状态	活跃	活跃
已发行版本数	26个Beta	1个Beta
首次发行日期	2015年2月22日	2016年7月30日
发行间隔	一周（或两周）	未知
贡献者数	106	TiDB(53), TiKV(22), PD(13)
提交数	13697	TiDB(3627), TiKV(1685), PD(385)
星指数	7641	TiDB(4745), TiKV(1040), PD(22)
文档信息	比较完善	欠缺
一致性算法	Raft	Raft
底层数据库	RocksDB(基于LevelDB)	RocksDB(基于LevelDB)
官方网站	www.cockroachlabs.com	www.pingcap.com/

CockroachDB (NewSQL数据库)

CockroachDB is a distributed SQL database built on a transactional and strongly-consistent key-value store.

It scales horizontally; survives disk, machine, rack, and even datacenter failures with minimal latency disruption and no manual intervention.

supports strongly-consistent ACID transactions; and provides a familiar SQL API for structuring, manipulating, and querying data.

Source: https://github.com/cockroachdb/cockroach

CockroachDB与其它数据库的比较

	CockroachDB	MySQL	PostgreSQL	Oracle	SQL Server	Cassandra	HBase	MongoDB	DynamoDB
Automated Scaling	Yes	No	No	No	No	Yes	Yes	Yes	Yes
Automated Failover	Yes	Optional	Optional	Optional	Optional	Yes	Yes	Yes	Yes
Automated Repair	Yes	No	No	No	No	Yes	Yes	Yes	Yes
Strongly Consistent Replication	Yes	No	No	Optional	Optional	Optional	No	No	Yes
Consensus-Based Replication	Yes	No	No	No	No	Optional	No	No	Yes
Distributed Transactions	Yes	No	No	Yes	Yes	No	No	No	No*
ACID Semantics	Yes	Yes	Yes	Yes	Yes	No	Row-only	Document-only	Row-only*
Eventually Consistent Reads	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SQL	Yes	Yes	Yes	Yes	Yes	No	No	No	No
Open Source	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No
Commercial Version	No	Optional	No	Yes	Yes	Optional	Optional	Optional	Yes
Support	Limited	Full	Full	Full	Full	Full	Full	Full	Full

** In DynamoDB, distributed transactions and ACID semantics across all data in the database, not just per row, requires an additional transaction library.*

Source: https://www.cockroachlabs.com/docs/cockroachdb-in-comparison.html

CockroachDB SQL支持的范围

See next slide

Reference date: September 27, 2016

Official Source: https://www.cockroachlabs.com/docs/sql-feature-support.html

(for more detail explanation)

SQL-Feature	Support	Planned	Not-Support
ROW VALUES	**Alternative**: AUTO INCREMENT(SERIAL),Key-value pairs(2 column table)	Arrays,JSON	XML,UNSIGNED INT,SET, ENUM
CONSTRAINTS	Full-Support		
TRANSACTIONS	Full-Support		
INDEXES	Indexes,Multi-column indexes,Covering indexes **Potential**: Prefix/Expression Indexes,Geospatial indexes	Multiple indexes per query,Full-text indexes	Hash indexes,Partial indexes
SCHEMA CHANGES	Full-Support		
STATEMENTS	UPSERT,EXPLAIN **Functional**:JOIN (INNER, LEFT, RIGHT, FULL, CROSS),SELECT INTO **Alternative**:SELECT INTO		
CLAUSES	**Partial**: Subqueries,EXISTS	COLLATE	
FUNCTIONS	Full-Support		
CONDITIONAL EXPRESSIONS	Full-Support		
PERMISSIONS	Full-Support		
MISCELLANEOUS	Column families,Interleaved tables	Views,Common Table Expressions,Stored Procedures,Window functions	Cursors,Triggers,Sequences

CAP的解決办法

- Spanner[4] do the delay

TrueTime API directly exposes clock uncertainty, and the guarantees on Spanner's timestamps depend on the bounds that the implementation provides. If the uncertainty is large, Spanner slows down to wait out that uncertainty.

- Cockroachdb do using MVCC, Optimistic approach

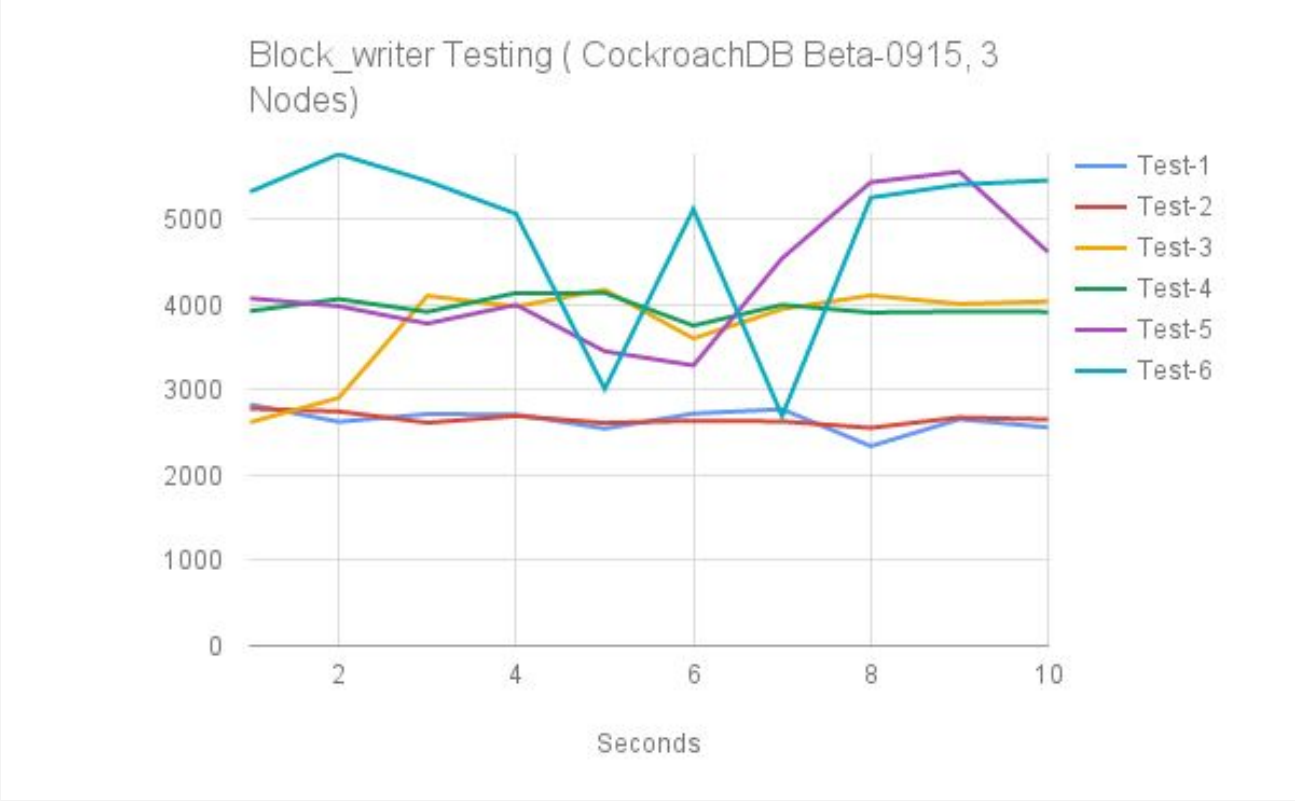
Unlike most relational databases, CockroachDB uses optimistic concurrency control instead of locking. This means that when there is a conflict between two transactions one of them is forced to restart, instead of waiting for the other to complete.

Block_Writer testing

The block writer example program is a write-only workload intended to insert a large amount of data into cockroach quickly. This example is intended to trigger range splits and rebalances.

Source Repo:
https://github.com/cockroachdb/examples-go/tree/master/block_writer

Block_Writer testing results



Jepsen testing

Jepsen testing as a high quality review of the correctness and consistency claims of modern database systems.

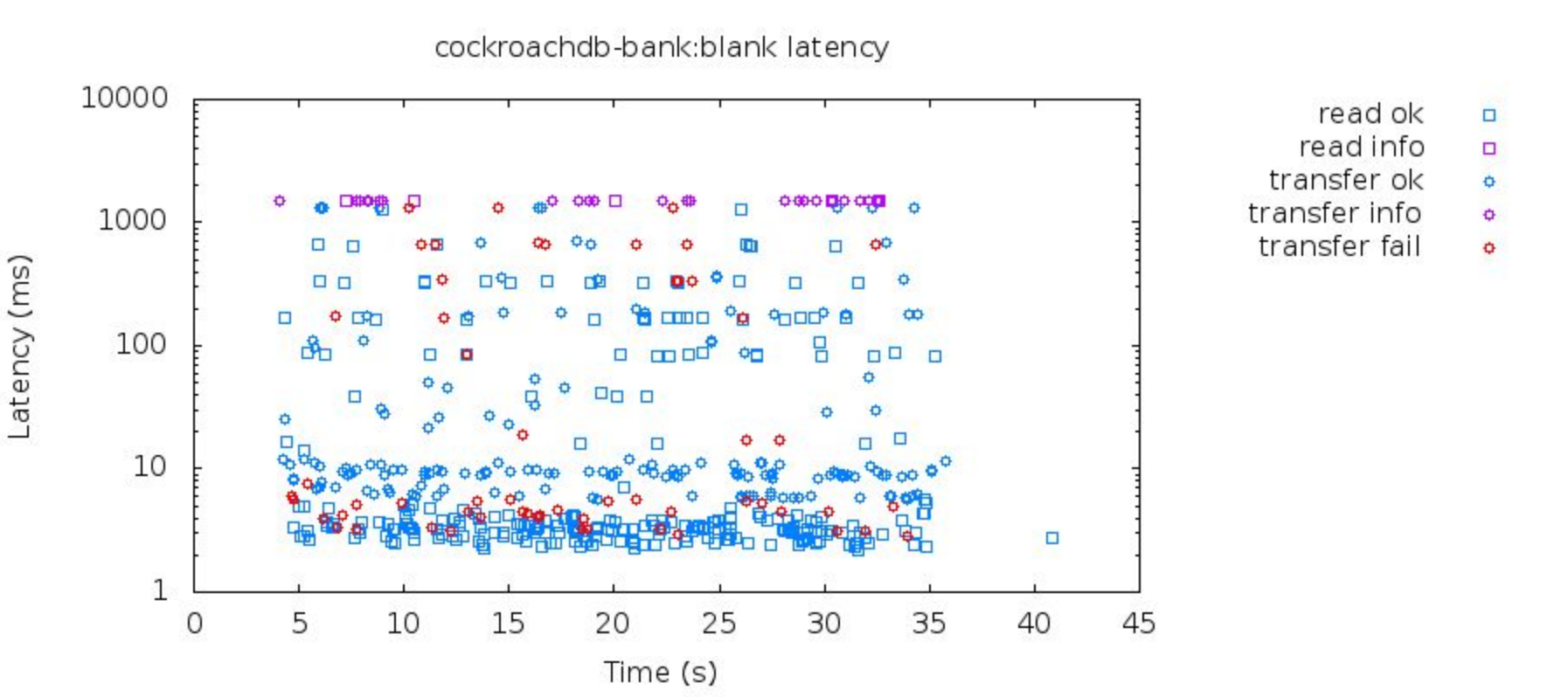
Written in Clojure

Site: https://aphyr.com/tags/jepsen

Source Repo: https://github.com/aphyr/jepsen

Cockroachlabs' try: https://www.cockroachlabs.com/blog/diy-jepsen-testing-cockroachdb/（测试结果：未发现严重的数据库一致性问题）

Snapshot isolation: pseudo-banking



source: https://www.cockroachlabs.com/blog/diy-jepsen-testing-cockroachdb/

Snapshot isolation: pseudo-banking (续)

This reveals there are still “transfer failures,” but looking at the log, we see they are simply cases where a transfer would otherwise result in a negative balance, and are thus disallowed by the model.

Jepsen的其它测试

根据cockroachlabs的尝试，以下测试均通过

- 一致性测试(唯一附加)
- 快照隔离(银行模拟)
- 序列化隔离（默认支持）
- 线性隔离（特殊场景支持）
- 多单元快照隔离
- 网络分割测试

YCSB testing

The goal of the Yahoo Cloud Serving Benchmark (YCSB) project is to develop a framework and common set of workloads for evaluating the performance of different "key-value" and "cloud" serving stores.

Written in Java

Site: https://research.yahoo.com/news/yahoo-cloud-serving-benchmark

Source Repo: https://github.com/brianfrankcooper/YCSB

YCSB测试环境

三个物理节点（超过3个节点时，为共用物理机器创建的额外节点）

- Intel(R) Xeon(R) CPU X5660 @ 2.80GHz 24核
- 128G内存
- Network 1000 Mbps

通过ycsb_starter 进行自动测试

通过ycsb2graph 生成图形结果

YCSB测试信息

All Server OS版本: CentOS-6.4

YCSB版本：ycsb-jdbc-binding-0.10.0

CockroachDB版本：beta-0829, beta-0915

Mysql版本：mariadb-10.0.13-linux-x86_64 (single node)

测试日期：2016-9-20

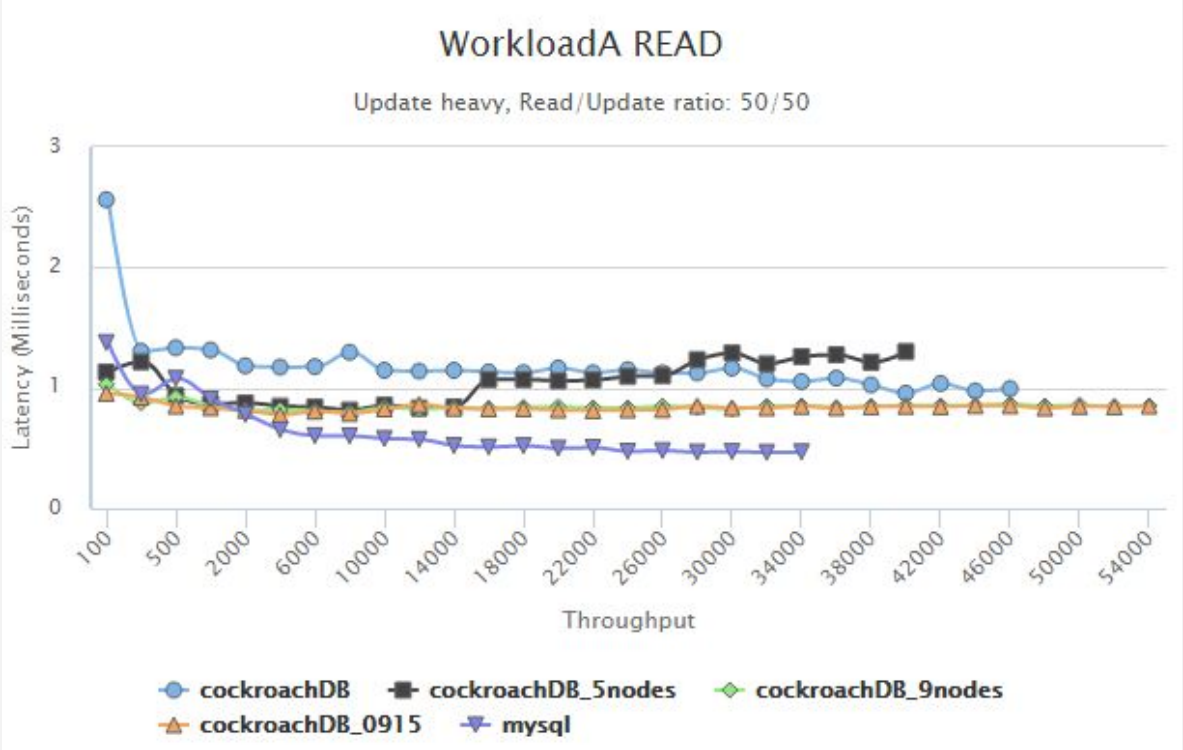
在线结果：http://g.clwen.com:9000/YCSB_cockroach_0829_vs_0915/

WorkloadA Read

CockroachDB新Beta版本有少量提升

CockroachDB与mysql的读性能比较接近(相差在一毫秒左右)

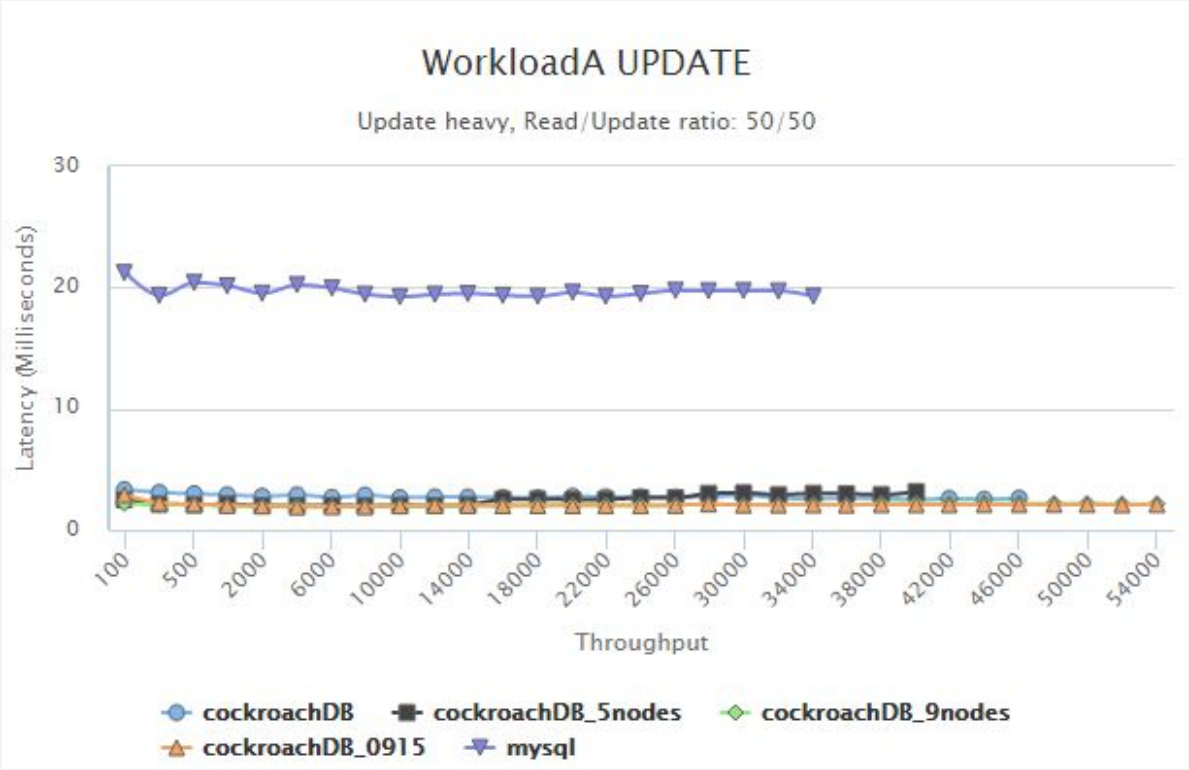
CockroachDB的扩展节点未发现明显的性能提升(分布式查询暂未实现？)



WorkloadA Update

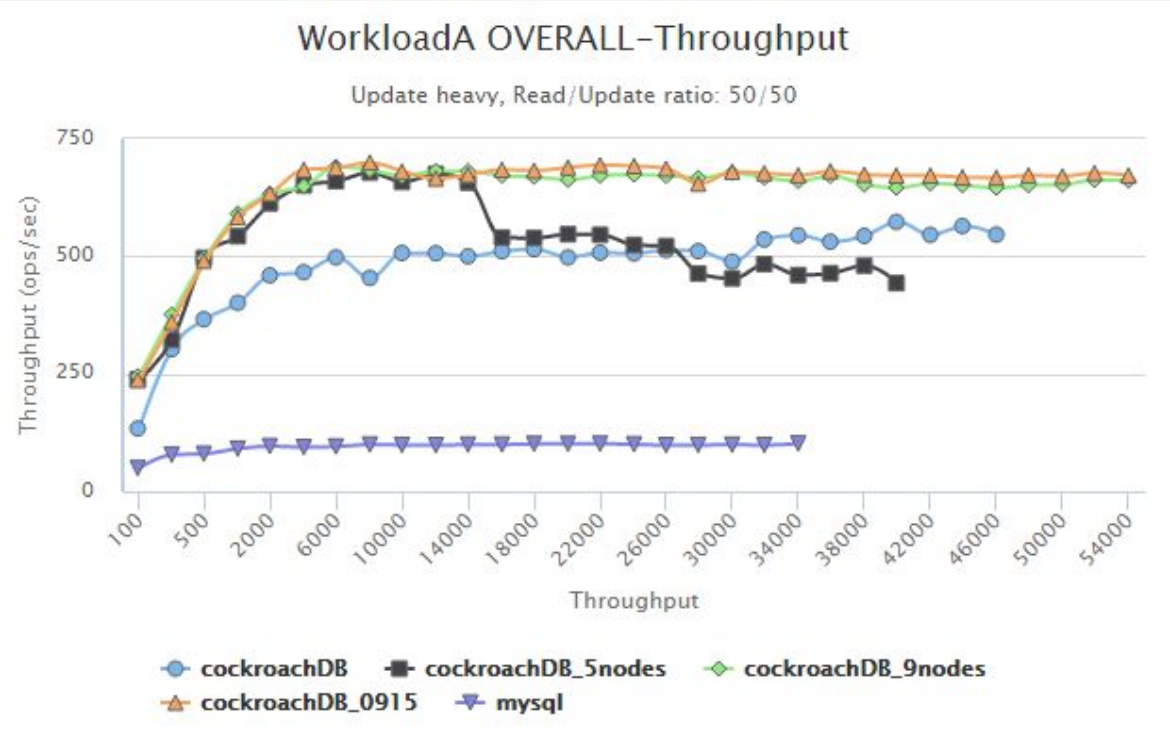
CockroachDB的写性能相比Mysql有较大的写性能提升，是Mysql的十倍写性能

扩展节点未发现明显的更新性能提升



WorkloadA Throughput

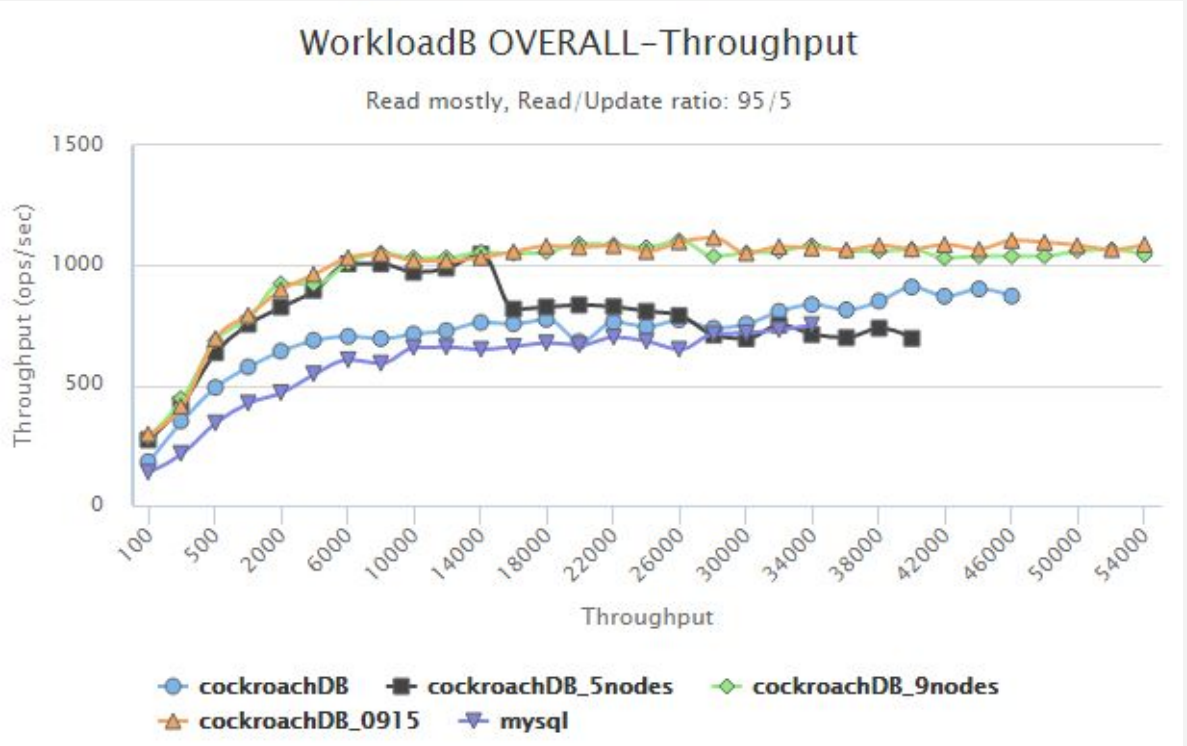
CockroachDB相较于Mysql读写性能有较大优势，保持在700 tps左右，Mysql则保持在100 tps左右



WorkloadB Throughput

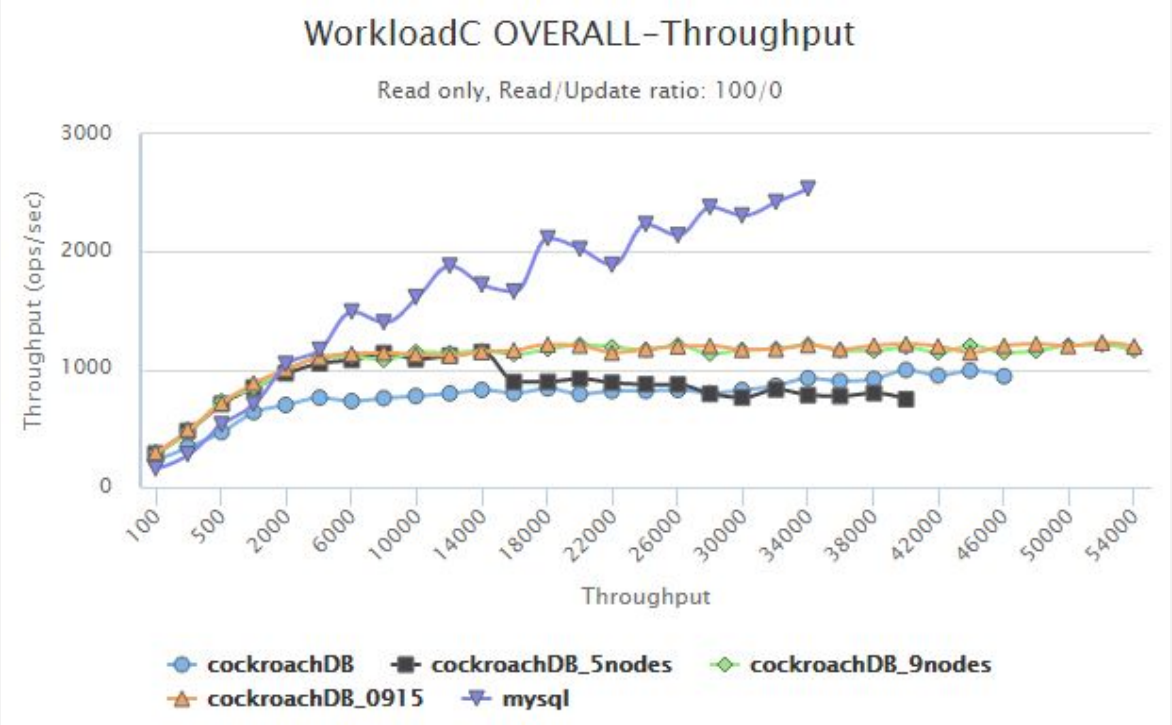
相比WorkloadA（读写分别各占50%）

WorkloadB（读占95%，写占5%），CockroachDB的读写性能在1000 tps左右，Mysql则在750 tps左右



WorkloadC Throughput

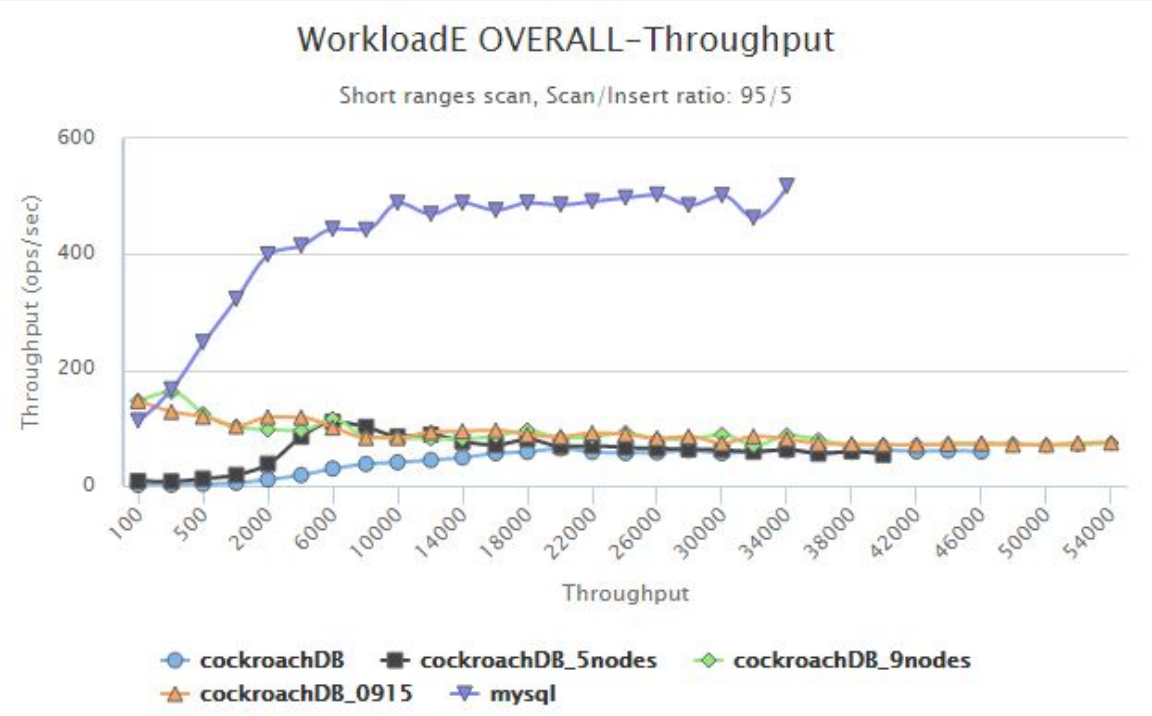
纯读的性能，Mysql较有优势，可达到2500 tps，CockroachDB则保持在1100 tps左右



WorkloadE Throughput

范围扫描的性能，Mysql较有优势，可达到500 tps，CockroachDB则在 140 tps左右，随着量的增加，反而降低到70 tps左右

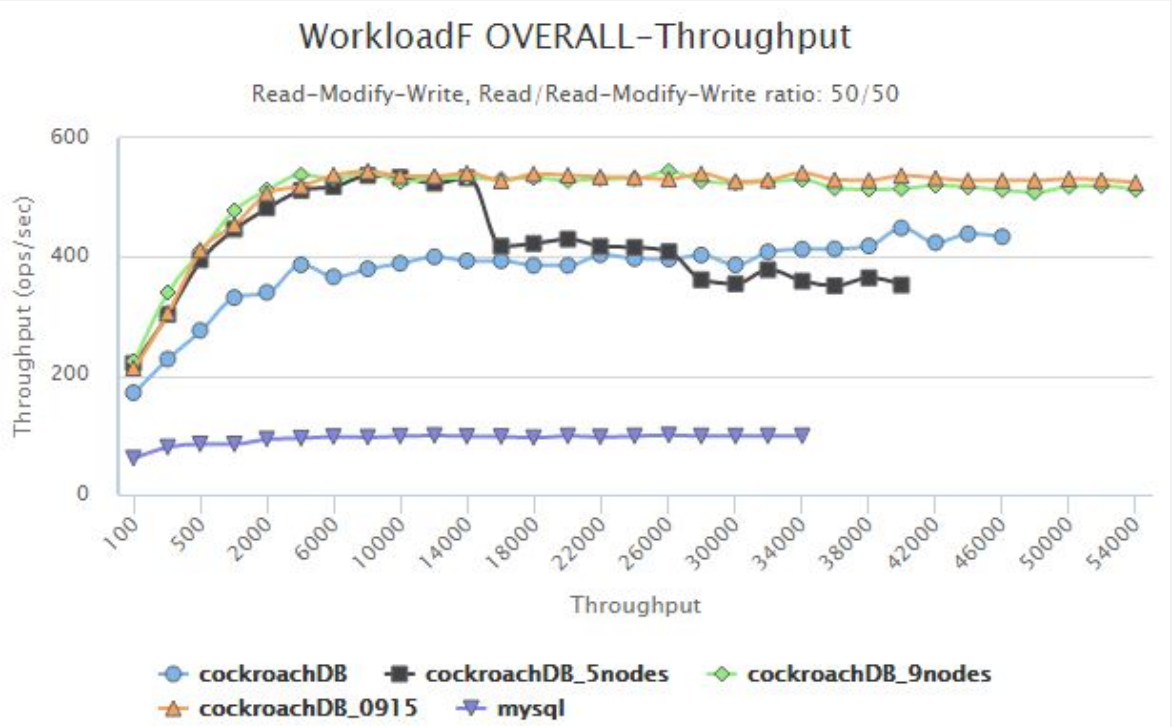
CockroachDB新Beta版本在初始阶段有明显的提升



WorkloadF Throughput

先读，修改，再写的性能，CockroachDB较有优势，保持在540 tps，Mysql保持在100 tps左右

CockroachDB新Beta版本在先读，修改，再写的性能上有明显的提升



YCSB测试结果总结

CockroachDB在读上与Mysql相近，在写上有较大的提升（10倍左右）。

Mysql相对来讲是一个趋于成熟稳定的产品，CockroachDB则保持高的版本迭代和性能的改进，从测试结果可以看出，相隔半个月的Beta版本就有明显的改进，再加上正在进行的Ddistributed SQL RFC(2015)，相信会有能充分体现分布式的优势的版本出现。

目前CockroachDB更侧重于先保证功能的正确性，然后再来做性能的优化或改进。

期间遇到的问题

Glibc版本过低

NTP时间同步问题

YCSB自动化测试和图形生成的问题

Glibc版本过低

CentOS6.4自带的Glibc-2.12版本，CockroachDB须Glibc至少2.14版本以上

编译Glibc新版本，并通过以下方式运行解决

```
url="$( which cockroach )"
g="/opt/glibc/build"
p="$g:$g/math:$g/elf:$g/dlfcn:$g/nss:$g/nis:$g/rt:$g/resolv:$g/crypt:$g/nptl:$g/dfp"
GCONV_PATH=$g/iconvdata LC_ALL=C
cockroachcmd="$g/elf/ld.so --library-path $p $url"
```

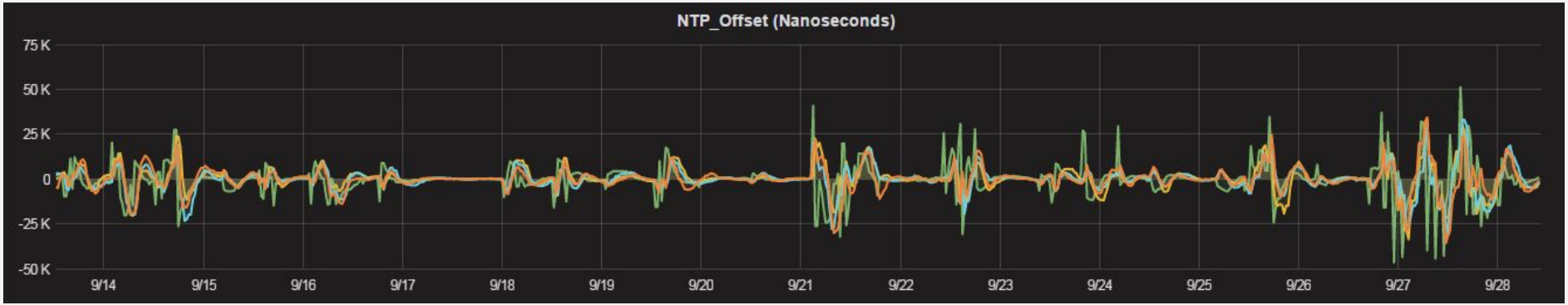
注：此依赖来自于RocksDB(C++)

NTP时间同步的问题

发现NTP恰当配置可以保证时间抖动控制在250ms以内，符合CockroachDB的要求

NTP的配置需要保证节点与时间服务器在同一局域网内

下图所示（平均控制在不超过70us的范围内）



YCSB自动化测试和图形生成的问题

YCSB官方的测试方法大量靠手工，没有自动化整个测试流程，只提供非常基础的单条测试，不易进行重复测试，且不支持生成图形结果

通过编写以下两个程序解决

ycsb_starter 进行自动测试（由Shell实现）

ycsb2graph 生成图形结果（通过Highcharts Javascript 库实现图形展现）

Reference

1. Matthew Aslett. "*CAP theorem - two out of three ain't right*", 451 research group, April 2013, pp 72.
2. Ivan Glushkov. "*NewSQL Overview*", Feb 2015, pp 13-14.
3. Andrew Pavlo and Matthew Aslett. "*What's Really New with NewSQL?*", Aug.2016, pp 9.
4. J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, W. Hsieh, S. Kanthak, E. Kogan, H. Li, A. Lloyd, S. Melnik, D. Mwaura, D. Nagle, S. Quinlan, R. Rao, L. Rolig, Y. Saito, M. Szymaniak, C. Taylor,R. Wang, and D. Woodford. "*Spanner: Google's Globally-Distributed Database*". In OSDI, 2012.

Reference（续）

5. J.Shute, C.Whipkey, D.Menestrina, R.Vingralek, B.Samwel, B.Handy, E.Rollins, M.Oancea, K.Littlefield,S.Ellner, J.Cieslewicz, I.Rae, T.Stancescu, H.Apte. "*F1: A Distributed SQL Database That Scales*", Google, Inc. Aug.2013

总结

NewSQL是目前传统数据库与NoSQL数据库两难局面的解决方案。

CockroachDB是一个实现的较好和功能支持比较完整的一个开源NewSQL数据库，待软件逐渐稳定，并形成产品发布版时，可以考虑正式用于生产环境。

附录1—Spanner介绍

- Spanner is Google's scalable, multi-version, globally-distributed, and synchronously-replicated database.
- It is the first system to distribute data at global scale and support externally-consistent distributed transactions.
- Spanner use a novel time API that exposes clock uncertainty.
- This API and its implementation are critical to supporting external consistency and a variety of powerful features: non-blocking reads in the past, lock-free read-only transactions, and atomic schema changes, across all of Spanner.

Source: https://research.google.com/archive/spanner.html

附录1—F1介绍（F1 is built on Spanner）

- F1 is a distributed relational database system built at Google to support the AdWords business.
- F1 is a hybrid database that combines high availability, the scalability of NoSQL systems like Bigtable, and the consistency and usability of traditional SQL databases.
- Synchronous replication implies higher commit latency, but we mitigate that latency by using a hierarchical schema model with structured data types and through smart application design.
- F1 also includes a fully functional distributed SQL query engine and automatic change tracking and publishing.

Source: https://research.google.com/pubs/pub41344.html