Cynomys 的 JSON 定义格式

```
{
  “namespace”: “com.cynomys.code.serialization”,
  “type”: “record”,
  “name”: “Message”,
  “fields”: [
    {
      “name”: “Ti”,
      “type”: “string”
    },
    {
      “name”: “Database_name”,
      “type”: “int”
    },
    {
      “name”: “Table_name”,
      “type”: “int”
    },
    {
      “name”: “SQL”,
      “type”: “string”
    }
  ]
}
```

在 JSON 定义的模式中,分别定义了一条消息的几个域,为简洁明了表述消息内容,选择事务 ID(tid)、涉及的数据库名(database_name)、涉及的表名(table_name)以及还原出来的 SQL 语句(SQL).经过序列化模块产生的序列化文件会被传送到下一个处理模块进行进一步处理,序列化具体的流程如图 9 所示.

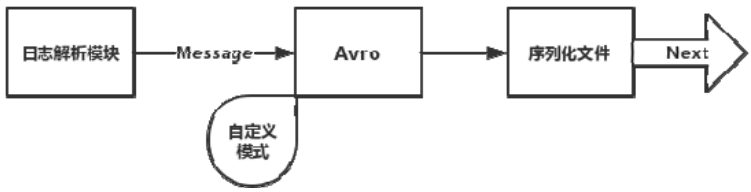

Fig.9 Flow chart of serialization based on Avro

图 9 基于 Avro 的序列化操作流程图

3.4 其他模块

除前面提到的日志解析模块和序列化模块外,Cynomys 还包含数据压缩加密、消息传递、SQL 执行等模块.在数据压缩加密模块中,本文所采用的 GZip 数据压缩算法[18],用 AES 算法对压缩包进行加密,RSA 算法对 AES 算法的密钥进行加密,与被加密的压缩包一起发送到消息传递模块.数据压缩加密模块的流程如图 10 所示.在消息传递模块中,本文选择 Redis 作为消息队列,基于 Redis 实现消息的发布/订阅(简称 Pub/Sub)模式[19].利用发布/订阅模式,解决了被压缩加密的消息包可能在网络传输过程中丢失或者消息发送方和接收方处理效率不一致带来的延迟等问题.

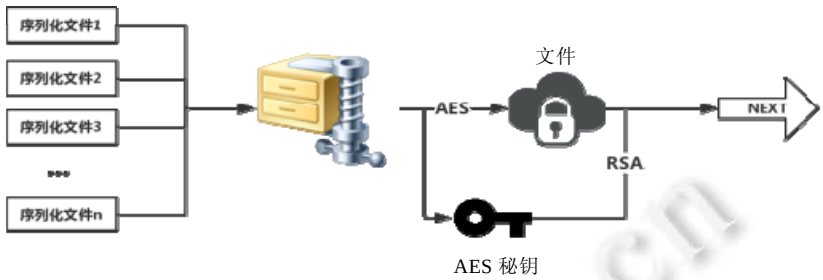

Fig.10 Flow chart of compression and encryption

图 10 数据压缩加密模块流程图

来自消息队列中的消息,需由消费模块对消息进行解密、解压缩并反序列化后使用消息.Cynomys 不会直接执行消息中的 SQL 语句,而是根据情况决定是否启用延迟提交机制.

4 实验评估与分析

本节内容旨在检验 Cynomys 的可用性、高效性、兼容性和正确性.在可用性实验中,主要验证 Cynomys 对数据库基本操作的支持度以及 Cynomys 本身的稳定性;在高效性实验中,主要验证了延迟提交机制下 Cynomys 的 INSERT 性能;在兼容性实验中,主要验证了 Cynomys 的数据库兼容性和运行平台的兼容性;在正确性实验中,主要验证了多个事务并发情况下 Cynomys 同步数据的正确性.

4.1 环境配置

Cynomys 的功能实验、性能实验在两台服务器上运行,其中,主服务器运行 MySQL 等行存储数据库,备份服务器运行 MariaDB ColumnStore 等列存储数据库.所有实验的测试数据均采用 TPC-C 或 TPC-H 基准测试数据集.具体的实验环境参数见表 3.

Table 3 Configurition of experiment

表 3 实验环境配置

参数设置	具体信息
主机 CPU	Intel(R) Xeon(R) CPU E5-4607
主机内存	8GB
操作系统	Linux CentOS 7.4
网络	千兆以太网
MySQL	V5.7
ColumnStore	V1.1.2
JDK	1.8.0_101

4.2 功能实验

4.2.1 数据类型支持度

数据库基本数据类型是支撑表元素的基础,Cynomys 对数据类型的解析是根据 MySQL 写日志的规则,不同的数据类型对应不同的 ID,对字符串类型和非字符串类型分别解析.本文对 MySQL 的各个基本数据类型都进行了相应的测试.实验结果表明:Cynomys 能够完全支持 MySQL 的基本数据类型(MySQL 的基本数据类型:https://dev.mysql.com/doc/refman/8.0/en/data-type-overview.html),MariaDB,MonetDB 的基本数据类型和 MySQL 一致.

4.2.2 操作类型支持度

本文用例中涉及的 SQL 语句大都与具体的数据直接相关,所以在实验论证中只对 DDL 语句和 DML 语句进行实验验证证,而对 DCL 语句不做支持.实验结果见表 4,可以看出,Cynomys 能够支持所有 DDL 语句和 DML 语句.

Table 4 SQL statement support in Cynomys

表 4 Cynomys 对 SQL 语句类型支持情况

语句类型		支持情况
DDL	CREATE	支持
	ALTER	支持
	DROP	支持
	TRUNCATE	支持
	COMMENT	支持
DML	SELECT	支持
	INSERT	支持
	UPDATE	支持
	DELETE	支持

4.2.3 系统稳定性

作为一个实时同步工具,对其稳定性要求一般较高,必须保证 7×24 小时无间断的稳定运行和快速响应.本实验是将 Cynomys 部署在服务器上连续运行一周,在每天的 4 个时间点各发起一次同步请求,观察 Cynomys 的表现并进行采样,得出结论,Cynomys 的稳定性达到了 7×24 小时及时响应,符合 MySQL 的运行状态和同步要求,实验结果见表 5.

Table 5 Test table of operational stability

表 5 Cynomys 运行稳定性测试表

	00:00	6:00	12:00	18:00
第 1 天	正常	正常	正常	正常
第 2 天	正常	正常	正常	正常
第 3 天	正常	正常	正常	正常
第 4 天	正常	正常	正常	正常
第 5 天	正常	正常	正常	正常
第 6 天	正常	正常	正常	正常
第 7 天	正常	正常	正常	正常

4.3 性能实验

性能实验中,主要测试从行存储到列存储的同步效率,其中,列存储数据库选择前面提到的 MariaDB ColumnStore.测试集选用的是 TPC-C 产生的模拟数据.

MariaDB ColumnStore 主要用于分析型数据库使用,所以 Cynomys 的同步只涉及 INSERT,对 UPDATE 和 DELETE 性能不作要求,在性能实验中,主要是要检验 Cynomys 执行 INSERT 的性能.

在测试 Cynomys 的 INSERT 性能时将启用延迟提交功能,通过调整延迟提交的缓冲区大小,得到实验数据如图 11 所示.

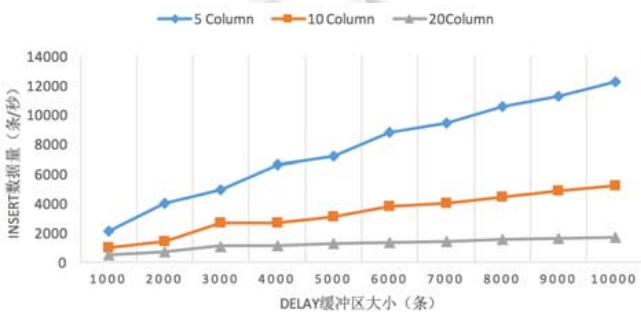

Fig.11 Figure of DELAY buffer and INSERT effiency

图 11 DELAY 缓冲区大小与 INSERT 效率关系表

依据上述实验结论,延迟提交确实可以提高 INSERT 操作的同步性能.在内存容量充足的情况下,缓存的待插入数据越大,相应的同步效率越高.

根据 MySQL 的现实场景,一般峰值的 INSERT 量在 1 000 条/s 左右.由于表的列数对插入性能影响较大,所以对于延迟提交缓冲区大小的需要可依据表结构来确定.以一个拥有 20 个字段的表为例,满足峰值 INSERT 的缓冲区大小设置为 3 000 左右即可.

从数据中还能看出另一个问题:随着表的列数增加,INSERT 操作效率的提升降低了.这是由于受测列式数据库的特殊存储结构,列式数据库中每一列都对应一个文件,这样的文件组织结构导致当表中的列增多时,会对性能有较大的影响.

4.4 兼容性实验

兼容性实验主要测试 Cynomys 在不同操作系统上是否能正常运行以及不同行列数据库能否正常同步.

Cynomys 采用 JAVA 语言开发完成,目的就是为了能兼容各个操作平台.本节选取了当前主流的操作系统包括 Windows 7,Windows 8,Ubuntu,CentOS 等不同版本测试 Cynomys 在各个操作系统平台的运行情况,实验结果见表 6.

Table 6 Experimental result of compatibility of Cynomys

表 6 Cynomys 平台兼容性实验结果

操作系统	运行情况
Windows 7 32bit	正常
Windows 7 64bit	正常
Windows 8 64bit	正常
Ubuntu 14.04 LTS 64bit	正常
Ubuntu 16.04 LTS 64bit	正常
CentOS 6.5 64bit	正常
CentOS 7.2 64bit	正常

从表 6 可以看出,Cynomys 在当前主流操作系统上都能正常运行.此外,还对不同版本的行列数据库进行了软件兼容性实验.Cynomys 是基于 Binlog 实现的,因此所有基于 Binlog 且能执行标准 SQL 语句的行数据库理论上都能运行,本节测试了 MySQL 与 MariaDB(ColumnStore)的不同版本进行软件兼容性实验,实验结果见表 7.

Table 7 Experimental result of compatibility of database in Cynomys

表 7 Cynomys 数据库兼容性实验结果

MySQL 版本(InnoDB)	MariaDB 版本(ColumnStore)	同步情况
MySQL 5.5	ColumnStore 1.1.0	正常
MySQL 5.5	ColumnStore 1.1.1	正常
MySQL 5.5	ColumnStore 1.1.2	正常
MySQL 5.6	ColumnStore 1.1.0	正常
MySQL 5.6	ColumnStore 1.1.1	正常
MySQL 5.6	ColumnStore 1.1.2	正常
MySQL 5.7	ColumnStore 1.1.0	正常
MySQL 5.7	ColumnStore 1.1.1	正常
MySQL 5.7	ColumnStore 1.1.2	正常

除表 7 给出的 MySQL 和 MariaDB 外,还对其他列存储数据库如 MonetDB 进行了测试,在这些行存储数据库之间,Cynomys 都能正常同步数据,不同列存储数据库对数据查询时间的影响如图 12(以 MariaDB 和 MonetDB 进行对比)所示.

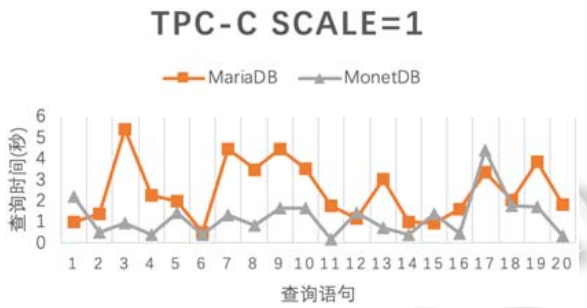

Fig.12 Influence of query time for different column-based database

图 12 不同列存储数据库对查询时间的影响

4.5 正确性实验

正确性实验主要测试在多用户(进程)并发的情况下,Cynomys 同步数据的准确性与源数据是否一致.本节实验所选取的主数据库为 MySQL5.6,列数据库为 MonetDB11.27,数据集为 TPCH-DBGEN 生成的 TPC-H 数据中的 LINEITEM 表的前 100 万条数据.

实验假设有 4 个用户同时对 MySQL 写入共 100 万条数据,每个用户写入 25 万条.从 MySQL 写入数据一共用时 5.68s,由于 Cynomys 对列存储数据库 SQL 语句提交的优化,数据写入 MonetDB 的速度与 MySQL 写入的速度几乎是同步的,仅用 6.12s.实验为采样实验,测试采样为 30%,50%,100%的情况下,数据响应速度与数据同步的准确率.对比是否与原数据相同采用将所有字段进行拼接,比较与原数据的字符串是否相同的方法.同步实验准确性结果如图 13 所示.

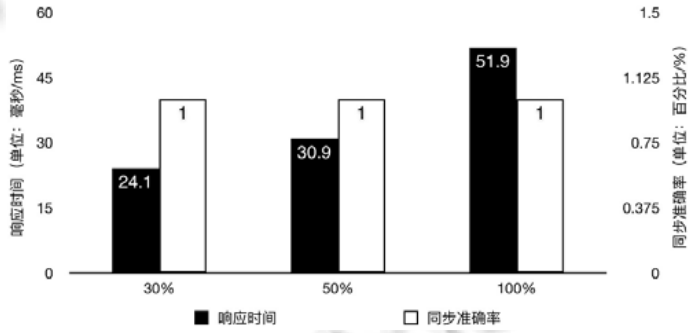

Fig.13 Accuracy of synchronization

图 13 同步准确性实验

实验结果显示,在不同采样大小的情况下,Cynomys 同步到从数据库的数据都能与源数据保证完全一致的准确性,验证了第 2.5 节中对数据同步正确性的说明分析.

5 总 结

在当前分布式系统得到广泛应用的背景下,本文提出了一种利用数据库所提供的 Binlog 机制,设计了日志解析器 BinParser 和日志还原器 BinReducer.为了应对分布式场景,也为工具实现了包括序列化、压缩加密、消息分发等一系列的工作.同时,针对列存储数据库本身在 INSERT 性能上存在的不足,提出了延迟提交的思想.基于以上各项技术实现了同步工具 Cynomys,用于同步行存储数据库和列存储数据库之间的数据.该方法弥补了不同存储模式的异构数据库之间无法进行实时数据同步的空白,解决了从分布式行存储数据库到列存储数据库的同步问题,且在实验中表现出了良好的性能.

此外,目前 Cynomys 只支持部分版本的 MySQL 数据库,在 MySQL5.6 之后,由于 Binlog 引入了 CRC 校验,Cynomys 对于如何适应 CRC 校验并没有给出解决方案,所以对于最新版本的 MySQL 数据库使用场景还不能做到同步,需要人为地对高版本 MySQL 进行配置才可以.这也是未来需要改进的方面.

References:

[1] Stonebraker M, Aoki PM, Litwin W, Pfeffer A, Sah A, Sidell J, *et al*. Mariposa: A wide-area distributed database system. VLDB Journal, 1996,5(1):48−63.

[2] Chen K, Zhou Y, Cao Y. Online data partitioning in distributed database systems. In: Proc. of the Int'l Conf. on Extending Database Technology (EDBT 2015). 2015. 1−12.

[3] Corbett JC, Dean J, Epstein M, *et al*. Spanner: Google's globally-distributed database. In: Proc. of the Usenix Conf. on Operating Systems Design and Implementation, Vol.31. 2012. 251−264.

[4] Wang J, Zhang DS. Research and design of distributed database synchronization system based on middleware. In: Proc. of the Modern Electronics Technique. 2016. 685−688.

[5] Lahiri T, Chavan S, Colgan M, *et al*. Oracle database in-memory: A dual format in-memory database. In: Proc. of the IEEE, Int'l Conf. on Data Engineering. 2016. 1253−1258.

[6] Mukherjee N, Chavan S, Colgan M, *et al*. Distributed architecture of oracle database in-memory. Proc. of the VLDB Endowment, 2015,8(12):1630−1641.

[7] Mukherjee N, Kulkarni K, Jin H, *et al*. How does oracle database in-memory scale out? In: Proc. of the Int'l Joint Conf. on Software Technologies, Vol.1. 2015. 1−6.

[8] Färber F, May N, Lehner W, *et al*. The sap hana database—An architecture overview. Bulletin of the Technical Committee on Data Engineering, 2012,35(1):28−33.

[9] Wang Z. Research and implementation of load balancing algorithm for offline data migration [MS. Thesis]. Shenyang: Northeastrn University, 2015 (in Chinese with English abstract).

[10] Li GX, Liu S, Liu JC, *et al*. Research and application of data synchronization service platform based on achived logs. Electric Power Information and Communication Technology, 2010,8(2):31−35 (in Chinese with English abstract).

[11] Song FL. The research and implementation of massive data synchronization system for database based on log parser [MS. Thesis]. Guangzhou: South China University of Technology, 2016 (in Chinese with English abstract).

[12] Lin Y, Chen ZB. Implementation of synchronization system for distributed database. Computer Engineering and Design, 2010, 31(24):5278−5281 (in Chinese with English abstract).

[13] Zheng HM. Research and implementation of heterogeneous database synchronization technology based on SQL restore method. Computer Era, 2008(10):15−18 (in Chinese with English abstract).

[14] Prisco RD, Lampson B, Lynch N. Revisiting the Paxos algorithm. In: Proc. of the Int'l Workshop on Distributed Algorithms, Vol.243. 1997. 111−125.

[15] Xu JX, Hou ZS. Notes on data-driven system approaches. Acta Automatica Sinica, 2009,35(6):668−675.

[16] Boncz PA, Zukowski M, *et al*. MonetDB/X100: Hyper-pipelining query execution. In: Proc. of the Int'l Conf. on Innovation Database Research (CIDR), Vol.5. 2005. 225−237.

[17] Bouchenak S, Hagimont D, Palma ND. Techniques for implementing efficient Java thread serialization. In: Proc. of the ACS/IEEE Int'l Conf. on Computer Systems and Applications, Vol.34. 2003. 355−393.

[18] Zeng CH, Zhang JJ, Xiong SF. Design and implementation of P2P remote assistance system based on JXTA. Journal of Jiangxi University of Science and Technology, 2009,30(3):36−40 (in Chinese with English abstract).

[19] Gutierrez F. Messaging with Redis. Berkeley, Apress, 2017. 120−155.

附中文参考文献:

[9] 王智.负载均衡的离线数据迁移算法的研究与实现[硕士学位论文].沈阳:东北大学,2015.

[10] 李功新,刘升,刘金长,等.基于归档日志的数据同步服务平台研究与应用.电力信息与通信技术,2010,8(2):31−35.

[11] 宋芳利.基于日志解析的数据库海量数据同步系统的研究与实现[硕士学位论文].广州:华南理工大学,2016.

[12] 林源,陈志泊.分布式异构数据库同步系统的研究与应用.计算机工程与设计,2010,31(24):5278−5281.

[13] 郑海明.基于 SQL 还原法的异构数据库同步技术的研究与实现.计算机时代,2008(10):15−18.

[18] 曾传璜,张晶晶,熊圣芬.基于 JXTA 的 P2P 远程协助系统的设计与实现.江西理工大学学报,2009,30(3):36−40.

徐梓荐(1994－),男,硕士,主要研究领域为关系型数据库,数据同步.

叶盛(1993－),男,硕士,主要研究领域为数据迁移,数据同步.

张孝(1972－),男,博士,副教授,CCF 高级会员,主要研究领域为数据库,大数据.