CryptDB密文数据库系统并行方案研究

王 伟,杨 庚,张成果
(南京邮电大学 计算机学院,江苏 南京 210003)

摘 要:加密数据库对隐私数据的加密存储保护是解决当前互联网中用户隐私数据泄露的一种可行方案。鉴于互联网中每日产生的用户隐私数据规模巨大,传统的串行计算会导致隐私数据的加密存储时间消耗较长。为提高加密存储等数据处理的速度,将MapReduce并行框架与CryptDB密文数据库系统进行有机结合,设计并实现了CryptDB密文数据库系统并行加密和分布式存储的方案。并行方案采用任务调度算法、文件分割算法来提高其并行性和可控性,通过重写MapReduce框架中的Map方法来实现CryptDB密文数据库系统的并行加密和分布式存储。基于由1个Master节点、3个CryptDB节点和3个MySQL服务器构成的实验平台,进行了并行方案的实验验证及其性能分析。实验结果表明,所构建的并行方案在3个CryptDB节点集群中的加速比可达到2.51,加密和存储时间节省了60.2%,可用于大规模关系数据的加密存储。

关键词:加密数据库;MapReduce;CryptDB系统;并行加密;分布式存储
中图分类号:TP31 **文献标识码**:A **文章编号**:1673-629X(2017)02-0090-06
doi:10.3969/j.issn.1673-629X.2017.02.021

Investigation on Parallel Scheme of CryptDB Encrypted Database System

WANG Wei, YANG Geng, ZHANG Cheng-guo
(College of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract: Encrypting data with encrypted DBMS is a feasible way to protect privacy of customer's sensitive data on the Internet. Due to the massive privacy data generated by Internet every day, the traditional serial calculation can lead to longer time consumption of encrypted storage of privacy data. Combined the characteristic of MapReduce and CryptDB, a parallel insert and distributed storage scheme is designed and realized to improve the accelerate the speed of encrypting. Another two algorithms named JobControl and FileSplit are also proposed to improve the parallelism and controllability of this scheme. Map method in MapReduce is rewritten to achieve the parallel encryption and distributed storage of CryptDB system. After doing the experiments and performance analysis on the platform consists of 1 Master node and 3 CryptBD nodes and 3 MySQL server nodes, the experimental results show that the speed-up radio of this proposed scheme can reach 2.51, and the total time cost of CyrptDB parallel scheme can be reduced to 39.8% on the cluster consisting of 3 CyrptDB nodes. It can be used into the encryption and storage of large-scale relational data.

Key words: encrypted DBMS; MapReduce; CryptDB system; parallel encryption; distributed storage

0 引 言

随着互联网的普及和互联网技术的高速发展,网络数据库安全成为了新兴的研究领域。计算机安全研究所(CSI)和FBI统计发现,平均每年有超过70%的用户曾遭受过网络攻击。这表明,传统的防火墙技术已经无法保证网络数据库的绝对安全。攻击者可以利用网络应用的安全漏洞窃取用户的隐私数据,对数据好奇的数据库管理员也会对用户隐私数据造成泄漏的风险。因此,如何设计并实现可检索的加密方案并运用到传统的数据库中,以加密用户隐私数据并支持在密文上进行SQL操作,成为了目前亟需解决的问题。

1995年,Benny Chor等提出了私有信息检索的概念,并在不泄露检索信息的前提下,实现从数据库中检索到用户所需的信息[1]。2000年,D. Song等提出一种可检索的对称加密方案(SSE)[2],开创了在不解密的情况下查询密文的先例,具有很高的实用性。在此

收稿日期:2016-04-11 **修回日期**:2016-08-05 **网络出版时间**:2017-01-10
基金项目:国家自然科学基金资助项目(61272084,61572263)
作者简介:王 伟(1992-),男,硕士研究生,研究方向为加密数据库、并行计算;杨 庚,博士,教授,博士生导师,CCF高级会员,研究方向为网络与信息安全、分布与并行计算、大数据隐私保护。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20170110.1019.060.html

之后,可检索加密机制进展迅速,并且支持对密文数据进行多关键词和模糊关键词搜索[3-6]。2004年,D. Boneh等提出了真正意义上的可搜索公钥加密方案(PEKS),为此业界把2004年定义为可搜索公钥加密的元年[7]。另外,可检索加密机制不但在理论上有了多样化的发展,许多研究者也将它们应用到了实际的场景之中。例如,MIT计算机科学和人工智能实验室(CSAIL)研发了一个加密数据库系统CryptDB[8]。该数据库系统允许用户查询加密的SQL数据库,而且能在无需解密储存信息的情况下返回结果。由于CryptDB系统采用了多层加密的洋葱加密方案来加密数据,并把数据加密成多份以适用于多种检索场景,当数据规模巨大时,其加密存储对服务器的运算和存储开销是巨大的。为了解决大规模数据加密时间开销巨大的问题,Kamara等在2013年提出了并行同态加密方案[9]。该方案支持通过评估算法对加密数据进行并行处理;并且研究了在MapReduce并行计算模型下的各种操作,包括关键词检索。2015年,F Wang等提出了使用Hadoop集群对明文数据库中已存在的数据进行并行加密并存储的方案,适用于企业级大型数据库中明文数据加密成密文存储的场景,相比于单一服务器时间开销更小[10]。

文中以CryptDB系统的加密存储过程为出发点,利用云计算MapReduce框架对海量数据运算高效的特点,设计并实现了基于任务并行的CryptDB系统优化方案,实现了对大规模隐私数据进行高效加密及存储的处理。该方案对DML/DDL语句进行分组,然后把每组操作分发到部署有CryptDB系统的节点上,并对其进行加密计算并存储,以达到缩短加密及存储时间的目的。

1　并行方案设计

1.1　CryptDB密文数据库系统

CryptDB[8]是一个密文数据库系统,它在可信的MySQL-Prxoy代理服务器端对明文SQL语句进行拦截,之后对隐私字段进行加密,随后将改写后的SQL语句提交到不可信的MySQL服务器端执行存储以实现对隐私数据的保护。CryptDB系统可直接对密文数据进行SQL操作。实现该操作的核心是三个创新的方案[11]:

(1)利用支持SQL的加密策略将SQL操作映射到加密框架中;

(2)可调整的加密方案使得可以根据用户的查询语句来调整数据的加密等级;

(3)洋葱加密方案可以高效地调整数据的加密等级。

1.2　并行方案模型

为了保证数据的安全性及密文的可检索,CryptDB系统利用多个洋葱模型对数据进行加密。因此利用CryptDB密文数据库系统对隐私数据进行加密存储的计算开销是巨大的。文中提出基于MapReduce计算框架的CryptDB系统任务并行方案,结合云计算环境的特性与CryptDB密文数据库系统的特点,将巨大的计算任务分发到各个部署有CryptDB系统的节点上进行计算并提交到对应的MySQL服务器上存储,方案中负责任务调度的主节点和负责加密的CryptDB节点都是可信服务器,因此不会造成明文的泄漏。并行方案如图1所示。

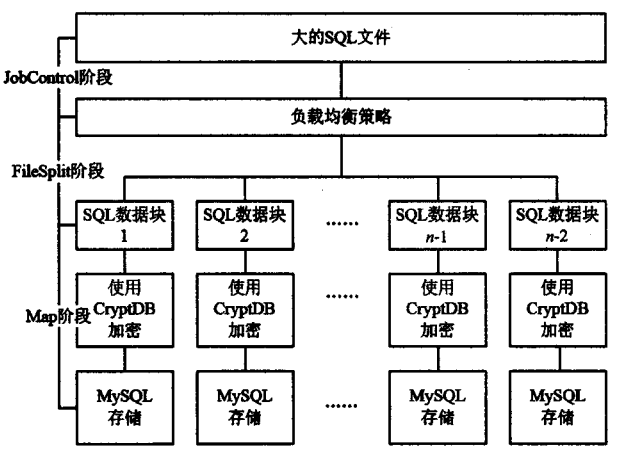

图1　任务并行方案

MapReduce的编程模型一般包含三个重要的组成部分:分片、map阶段和reduce阶段。首先,该方案根据负载均衡策略将大的SQL指令分割成多个分块;ResourceManager根据MapReduce框架的调度机制将各个分块分发到相应的CryptDB节点上,并启动mapper对分块中的明文SQL进行字段提取;最后调用CryptDB系统对数据进行加密。加密后的密文SQL将提交到MySQL服务器端执行。由于方案中每个CryptDB节点都对应一台MySQL服务器,并且加密后的SQL仍为标准的SQL语句,可直接被MySQL解析。提交到MySQL服务器执行后,数据将直接以密文形式存储在MySQL数据库中。因此方案中不需对数据进行Reduce操作。

1.3　任务分配算法

该方案中的并行策略是基于任务的分割,考虑到每个节点的计算能力不同,若将任务平均分配,计算能力较弱的节点完成任务的时间会较长,得不到最优解。方案中提出了任务分配算法,该算法致力于在任务并行阶段提高并行度,得到较优的map时间。

算法1:任务分配算法。

输入:节点个数m,SQL语句的条数N;

输出:每个节点的任务规模X_m。

(1)输入参数为集群中 CryptDB 节点的个数 m，以及待加密的任务规模即 SQL 语句的条数 N。

(2)随机生成 n 条与待加密 SQL 语句属性个数及类型相同的测试 SQL 语句。

(3)将测试 SQL 语句通过 JDBC 分发到 m 个 CryptDB 节点上进行加密存储，得到时间开销 t_m。

(4)分别用测试 SQL 的条数除以每个节点的时间开销，计算出每个节点的计算能力 x_m，即 $x_m = \frac{n}{t_m}$。

(5)用待加密 SQL 语句的规模除以各节点计算能力的总和，计算出平均加密存储时间，即最优的 map 时间 t_{ava}，即 $t_{ava} = \frac{N}{\sum_{1}^{m} x_m}$。

(6)分别用最优的 map 时间乘以各节点计算能力，得到每个节点应分配到的任务规模 $X_m = t_{ava} \cdot x_m$。

说明：

(1)该算法不会影响整个方案的效率。第 2 步中的参数 n 为预先设定的一个常量($n \ll N$)。因此该算法时间开销相对于之后的加密存储操作而言，可以忽略不计。

(2)该算法应用于 JobControl 阶段，产生任务分配的策略，以提高方案的整体性能。

(3)该算法在每次执行任务前都会被执行，以根据实时计算能力产生分配策略。

1.4 文件分割算法

由于集群中有 m 个节点，所以文中方案加入文件分割算法，根据每个节点的计算能力，将大的 SQL 文件分割成相应大小的 SQL 文件，并按相应的编号命名文件块。map 阶段，每个 mapper 会根据文件名处理对应的文件。

算法 2：文件分割算法。

输入：大的 SQL 文件；

输出：分割后小的 SQL 文件。

(1)将大的 SQL 文件读到内存中。

(2)从文件的起始位置开始，按行读取 SQL 语句，保存到 temp 变量中。

(3)当累加器的 SQL 语句达到当前节点所需规模时，将 temp 中的内容保存到 HDFS 文件系统中，并以节点编号命名。

MapReduce 框架中的 InputFormat 算法是将大的文件分割成对应的文件块，分配给相应的 mapper 处理。文中提出的文件分割算法有别于 InputFormat 算法，将大的文件以小的文件格式保存在 HDFS 文件系统中。该算法使得 mapper 根据文件名处理相对应规模的小任务文件，而不是 MapReduce 框架中的随机分配任务。提高了并行方案的可控性，有利于提高方案的并行度。

1.5 Map 函数

根据 Map 函数循环执行的机制，结合 SQL 语句按条插入的特性，该方案重写了 MapReduce 框架中的 Map 函数。该函数用于 SQL 语句的执行与加密结果的存储。

算法 3：Map。

输入：文件路径、容器大小、节点个数。

(1)从 HDFS 中获取待处理文件的文件名，即节点的编号，该文件的文件名由算法 2 给出。

(2)根据编号从 properties 文件中读取节点的地址等信息，将该文件分发到对应的 CryptDB 节点进行解析处理。

(3)CryptDB 节点接收到待处理的 SQL 文件后，首先读取文件内容，然后对 SQL 语句中的关键词进行处理，提取出待加密的属性值。

(4)将提取出的属性值根据预先设定的容器大小进行拼接，构建成新的标准 SQL 语句。

(5)通过 JDBC 调用 CryptDB 密文数据库系统，执行 SQL 语句，加密并存储数据。

说明：

(1)该算法第 2 步提到的 properties 文件为构建 CryptDB 集群时所记录的配置信息，所包含的内容为节点编号及 IP 地址。

(2)该算法第 4 步中提到的容器大小为新建 MapReduce 任务时给出的值，该值应小于 CryptDB 密文数据库系统单次所能处理 SQL 的最大属性值个数。

2 并行方案性能分析

2.1 串行性能分析

文中首先讨论 CryptDB 系统 INSERT 操作的串行执行时间。CryptDB 系统 INSERT 操作可以看成三个部分：第一部分是查询元数据表，第二部分是对数据进行加密，第三部分是将加密后的数据插入到 MySQL 数据库。设这三个部分的时间分别为 T_{meta}、T_{enc}、T_{insert}，如式(1)所示。

$$T_s = T_{meta} + T_{enc} + T_{insert} \tag{1}$$

CryptDB 系统中任何类型的数据都利用“等值洋葱”和“保序洋葱”进行加密[12]，数值型数据还会利用“HOM 洋葱”加密，字符型数据会利用“SEARCH 洋葱”加密。其中，等值洋葱中分别对数据使用了 DET-JOIN 算法、DET 算法和 RND 算法进行加密。保序洋葱使用了 OPE 算法和 RND 算法对数据进行加密。HOM 洋葱使用同态算法加密。SEARCH 洋葱使用了 SEARCH 算法进行加密[13]。

对于数值型的数据，RND 采用的是添加初始化向

量IV的Blowfish算法,DET采用的是Blowfish加密算法,DETJOIN采用的是基于ECC(椭圆加密)的ECJoin算法。对于字符型数据,RND采用的是带初始化向量IV(即salt值)的CBC模式的AES加密算法,它能保证密文加密的随机性,即相同的明文加密后密文可能不相同;DET采用的是初始向量相同的CMC模式(即初始化向量都为"0"的CBC模式)AES加密算法,所以相同的明文加密后能得到相同的密文;OPE采用的是mOPE加密算法;HOM采用的Paillier加密算法;SEARCH采用的SWPSearch加密算法。设Blowfish加密算法Blowfish、AES、ECJoin、mOPE、HOM的复杂度分别为$R_{blowfish}$、R_{aes}、R_{ecjoin}、R_{mope}、$R_{paillier}$。由于CryptDB系统的代码中没有实现OPEJOIN算法和SEARCH洋葱,所以不进行讨论。并且由于CryptDB使用的加密算法明文在加密前都会进行填充操作,所以输入和输出的规模都是定长。对于数值型的数据,"等值洋葱"的复杂度为:

$$U_{oEq\_int} = R_{ecjoin} + 2 \cdot R_{blowfish} \tag{2}$$

"保序洋葱"的复杂度为:

$$U_{oOrder\_int} = R_{mope} + R_{blowfish} \tag{3}$$

"HOM洋葱"的复杂度为:

$$U_{oAdd} = R_{eAdd} \tag{4}$$

对于字符型数据,"等值洋葱"的复杂度为:

$$U_{oEq\_str} = R_{ecjoin} + 2 \cdot R_{aes} \tag{5}$$

保序洋葱的复杂度为:

$$U_{oOrder\_str} = R_{mope} + R_{aes} \tag{6}$$

由式(2)~(4)可知,加密一个数值型的数据的复杂度为:

$$U_{int} = U_{oEq\_int} + U_{oOrder\_int} + U_{oAdd} = R_{ecjoin} + 3 \cdot R_{blowfish} + R_{mope} + R_{paillier} \tag{7}$$

由式(5)、式(6)可知,加密一个字符型数据的复杂度为:

$$U_{str} = U_{oEq\_str} + U_{oOrder\_str} = R_{ecjoin} + 3 \cdot R_{aes} + R_{mope} \tag{8}$$

设待加密字段有a个数值型数据和b个字符型数据,则:

$$U_{enc} = a \cdot U_{int} + b \cdot U_{str} = (a+b) \cdot (R_{ecjoin} + R_{mope}) + 3a \cdot R_{blowfish} + (a+4b) \cdot R_{aes} + a \cdot R_{paillier} \tag{9}$$

由文献[14]可知,ECJoin的椭圆计算和mOPE编码的复杂度都为$O(\log n)$。由文献[15-16]可知,AES、Blowfish、Paillier的复杂度为$O(n)$。ECJoin中除了椭圆计算外还会使用AES对明文进行编码,因此ECJoin的复杂度为$O(n+\log n)$。mOPE编码后还会使用DET对B树的叶子进行加密,所以mOPE的复杂度也为$O(n+\log n)$。由于当$n>0$时,$n>\log n$,因此$n+\log n<2n$。所以ECJoin、mOPE的复杂度都可表示为$O(n)$。因此可设R_{ecjoin}、$R_{blowfish}$、R_{mope}、$R_{paillier}$分别为R_{aes}的$\alpha_1,\alpha_2,\alpha_3,\alpha_4$倍,即:

$$U_{enc} = (a+b) \cdot (\alpha_1 + 3\alpha_2 + 2\alpha_3 + \alpha_4 + 3) R_{aes} \tag{10}$$

设查询元数据表的时间和插入MySQL数据库的复杂度分别为δ次和ε次浮点计算,设浮点计算的时间为T_{fc},则:

$$\begin{aligned} T_{meta} &= \delta \cdot T_{fc} \\ T_{insert} &= \varepsilon \cdot T_{fc} \end{aligned} \tag{11}$$

因此,由式(1)、式(9)、式(10)得:

$$T_s = (U_{enc} + \delta + \varepsilon) \cdot T_{fc} \tag{12}$$

2.2 并行性能分析

设共有p个CryptDB节点处理包含a个数值型数据和b个字符型数据的SQL文件。该SQL文件被分割成了p个数据块,对应p个mapper。则每个分块中包含a/p个数值型数据和b/p个字符型数据。

因为CryptDB系统并行化方案没有Reduce阶段,所以只讨论map部分。设Map算法的复杂度为M,则得到:

$$M = (U_{enc} + \varepsilon)/p + \delta \tag{13}$$

在map阶段,每个mapper开始前和结束后都会和ResourceManager进行通信。又因为数据被分割成了p块,所以至少有$2p$次通信。设通信耗时为:

$$T_{tr} = \zeta \cdot p \cdot T_{fc} \tag{14}$$

则并行时间为:

$$T_p = M \cdot T_{fc} + T_{tr} \tag{15}$$

则加速比可表示为:

$$\eta = \frac{T_s}{T_p} = \frac{(U_{enc} + \delta + \varepsilon) \cdot T_{fc}}{M \cdot T_{fc} + T_{tr}} = p \frac{(a+b)(\alpha_1 + 3\alpha_2 + 2\alpha_3 + \alpha_4 + 3) R_{aes} + \varepsilon + \delta}{(a+b)(\alpha_1 + 3\alpha_2 + 2\alpha_3 + \alpha_4 + 3) R_{aes} + p\delta + \varepsilon + \zeta p^2} = p\left[1 - \frac{(p-1)\delta + \zeta p^2}{(a+b)(\alpha_1 + 3\alpha_2 + 2\alpha_3 + \alpha_4 + 3) R_{aes} + p\delta + \varepsilon + \zeta p^2}\right] \tag{16}$$

实际工程应用中,CryptDB节点数远小于待加密的数据的规模,即$p \ll a+b$,此时通信时间可以忽略不计,即$\zeta \to 0$。由于加密数据规模较大时,查询元数据表的时间相对于加密时间可忽略不计,即$\delta \to 0$。由式(16)可知,加速比理想值为p。

3 实　验

3.1 实验环境

实验的硬件平台包括1个Master节点和3个CryptDB节点及3个MySQL服务器。Master节点负责工作的监控和调度,CryptDB节点负责密文计算,

MySQL 服务器负责执行密文 SQL 并存储的任务，其配置均为：

CPU：Intel（R） Xeon E3－1225 v3，3.2 GHz/8 M Cache；

Memory：16 GB（2x8 GB）1 333 MHz Dual Ranked RDIM；

Disk：1 TB 3.5－inch 7.2 K RPM SATA II Hard Drive。

软件平台为：Ubuntu 12.04，Java 1.7.0，Hadoop 版本为 Hadoop－2.5.2，CryptDB 为 2015 年 3 月下载版本。

3.2 CryptDB 并行方案 INSERT 性能

实验的数据集为随机生成的 SQL 文件。测试文件所含的 SQL 语句条数为 1 000 ~ 10 000，间隔为 1 000，共十个文件。每条数据有五个数值型字段和五个字符型字段。实验中，当 mapper 的个数为 1 时，为串行时间。当 mapper 的个数为 n 时，表示 SQL 文件被分割成了 n 个数据分块，对应 n 个 CryptDB 节点，即 n 个 CryptDB 节点计算并行时间。

文中用相同的测试数据在原生的 CryptDB 系统中进行实验，并记录了不同规模的测试数据插入操作的总时间、系统吞吐量、方案的加速比以及方案的效率（加速比/节点个数）。结果如表 1 和图 2 所示。

表 1 部分实验结果

规模/条	节点个数	时间/s	吞吐量条/s	加速比	效率
2 000	1	117.8	17.0	1.00	1.00
	2	70.8	28.2	1.66	0.83
	3	49.6	40.3	2.38	0.79
4 000	1	252.9	115.8	1.00	1.00
	2	142.6	28.1	1.77	0.89
	3	115.6	34.6	2.19	0.73
6 000	1	336.3	17.8	1.00	1.00
	2	177.2	33.9	1.90	0.95
	3	130.5	46.0	2.58	0.86
8 000	1	434.1	18.4	1.00	1.00
	2	232.6	34.4	1.87	0.93
	3	176.8	45.2	2.46	0.82
10 000	1	540.5	18.5	1.00	1.00
	2	292.0	34.2	1.85	0.93
	3	215.2	46.5	2.51	0.84

从表 1 的数据和图 2 中的曲线变化趋势可以得出：

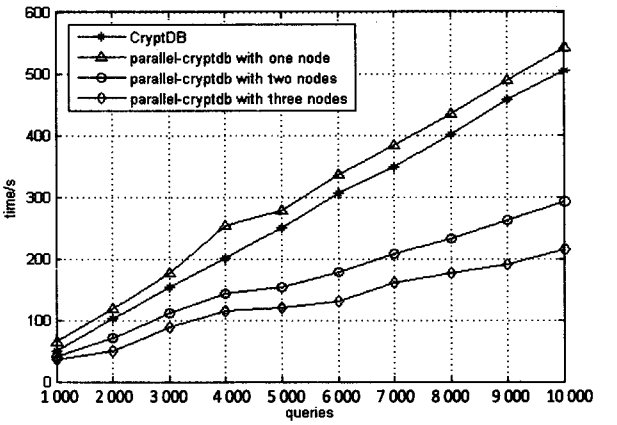

图 2 INSERT 操作时间

（1）对于相同规模的 SQL 语句，CryptDB 系统的执行时间比 CryptDB 并行方案单节点执行时间短。这是由于 Hadoop 并行框架中有任务调度和通信时间开销。

（2）随着 CryptDB 节点个数的增加，算法的总耗时呈下降趋势。这是由于集群的计算能力会随着节点个数的增加而增大。加速比和效率总体呈上升趋势，这是由于当数据规模足够大时，算法中对数据的加密时间占主导地位。通信的耗时在整体时间中可以忽略不计，所以加速比和效率都会逼近理想值。算法的加速比在最好的情况下，接近于 CryptDB 节点个数 p，与理论分析相符。并行方案的吞吐量随着节点个数增加接近于比例上升。这是由于在任务的分配阶段，使用了任务调度算法。该算法会根据集群中计算节点的计算能力动态分配任务，使得所有参与计算的节点不会存在闲置或等待的情况，表明了该方案有较高的并行度。

3.3 CryptDB 并行方案空间性能

加密后数据库空间大小也是衡量数据库并行方案的一个重要指标。图 3 和表 2 记录了 CryptDB 系统、并行 CryptDB 系统数据库文件的大小（并行方案的数据库文件大小为各节点数据库文件的平均大小 $\frac{\sum_{1}^{n} S_n}{n}$，$n$ 为节点个数），及并行 CryptDB 系统数据库文件大小增长百分比。

表 2 数据库大小

规模/条	数据库大小/MB			增长百分比/%
	单节点	两个节点	三个节点	
2 000	7.16	3.63	2.48	3.91
4 000	13.28	6.69	4.53	2.33
6 000	18.45	9.28	6.25	1.63
8 000	25.17	11.77	8.49	1.19
10 000	31.59	15.85	10.63	0.95

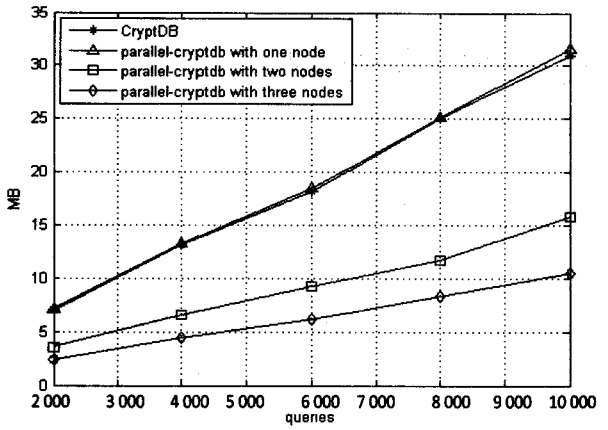

图 3　数据库文件平均大小

图 3 和表 2 反应的是:集群中平均每个 CryptDB 节点上的加密数据库文件所占的空间随着节点个数改变的变化趋势。并行方案单节点数据库文件大小接近于 CryptDB 的数据库文件大小。这表明该并行方案不会造成额外的空间开销。变化趋势表明,每个节点上的数据库平均大小接近于比例下降,当插入数据规模较大时,数据库的增长百分比接近于零。这是由于该并行方案中仅在每个数据库节点中冗余了表信息及部分索引信息,并且随着数据库插入内容规模的增大,该部分信息所占的存储空间的比例呈下降趋势。结论表明,该方案达到了分布存储的目的,数据规模较大时,对减小服务器的存储压力具有重要意义。

4　结束语

提出并实现了一种基于 MapReduce 框架的并行 CryptDB 加密数据库系统加密存储的并行方案。实验结果表明,该方案能够提高大规模关系数据库数据的加密和存储速度。方案中设计并实现了任务调度算法以产生任务分配策略,并对 SQL 文件按照产生的分配策略进行分割以提高方案的并行度。在 SQL 语句预处理、加密并存储的过程中,充分利用了 MapReduce 框架的并行性,提高了 CryptDB 加密数据库系统对于大规模数据加密并存储的效率。方案整体的加速比接近于 CryptDB 节点个数 p。和传统单节点加密数据库相比,每个 CryptDB 节点上的数据库大小和节点个数成反比,即在每个 CryptDB 计算能力相同的理想环境下,数据库大小为单节点的 $1/p$。当数据规模较大时,对减小服务器存储压力有着重要意义。

未来的工作重点是:将任务调度方案调整为任务执行过程中动态实时分配;设计并实现其他 SQL 操作的并行化方案。

参考文献:

[1] Chor B, Goldreich O, Kushilevitz E, et al. Private information retrieval[J]. Journal of the ACM, 1998, 45(6): 965-981.

[2] Song D X, Wagner D, Perrig A. Practical techniques for searches on encrypted data[C]//Proceeding of the IEEE symposium on security and privacy. [s. l.]: IEEE, 2000: 44-55.

[3] Xia Z H, Wang X H, Sun X M, et al. A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data[J]. IEEE Transactions on Parallel & Distributed Systems, 2016, 27(2): 340-352.

[4] Sun W, Wang B, Cao N, et al. Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking[J]. IEEE Transactions on Parallel & Distributed Systems, 2014, 25(11): 3025-3035.

[5] Li J, Wang Q, Wang C, et al. Fuzzy keyword search over encrypted data in cloud computing[C]//International conference on computer communications. [s. l.]: [s. n.], 2010: 1-5.

[6] Cao N, Wang C, Li M, et al. Privacy-preserving multi-keyword ranked search over encrypted cloud data[J]. IEEE Transactions on Parallel & Distributed Systems, 2011, 25(1): 829-837.

[7] Boneh D, Crescenzo G, Ostrovsky R, et al. Public key encryption with keyword search[C]//Proceedings of the EUROCRYPT. Berlin: Springer-Verlag, 2004: 506-522.

[8] Raluca A, Popa N, Zeldovich H, et al. CryptDB: a practical encrypted relational DBMS[R]. Cambridge, MA: Computer Science and Artificial Intelligence Laboratory, 2011.

[9] Kamara S, Raykoya M. Parallel homomorphic encryption[M]. Berlin: Springer-Verlag, 2013: 213-225.

[10] Wang F, Mathias K, Andreas S. Initial encryption of large searchable data sets using Hadoop[C]//Proceedings of the 20th ACM symposium on access control models and technologies. [s. l.]: ACM, 2015: 165-168.

[11] Tu S, Kaashoek M F, Madden S, et al. Processing analytical queries over encrypted data[J]. Proceedings of the VLDB Endowment, 2013, 6(5): 289-300.

[12] Popa R A, Li F H, Zeldovich N. An ideal-security protocol for order-preserving encoding[C]//Proceeding of the IEEE symposium on security and privacy. [s. l.]: IEEE, 2013: 463-477.

[13] Popa R A, Redfield C M S, Zeldovich N, et al. CryptDB: protecting confidentiality with encrypted query processing[C]//Proceeding of the SOSP. [s. l.]: [s. n.], 2011: 85-100.

[14] Popa R A, Zeldovich N. Cryptographic treatment of CryptDB's adjustable join[R]. Cambridge, MA: Computer Science and Artificial Intelligence Laboratory, 2012.

[15] Martin L. XTS: a mode of AES for encrypting hard disks[J]. IEEE Security & Privacy, 2010, 8(3): 68-69.

[16] Brakerski Z, Vaikuntanathan V. Efficient fully homomorphic encryption from (standard) LWE[J]. SIAM Journal on Computing, 2014, 43(2): 831-871.