一种基于中间件的异构数据库融合访问方法及系统

潘明明 李丁丁 汤 庸 刘 海
(华南师范大学计算机学院 广州 510631)

摘 要 在大数据时代,信息化数据呈爆炸式增长,传统关系型数据库和新兴的NoSQL数据库都难以全面且高效地面对这些挑战。因此,提出一种基于中间件的异构数据库访问方法(MingleDB),以结合NoSQL和传统关系型数据库的优点。MingleDB透明融合了NoSQL数据库和传统数据库的主要运行逻辑,同时又能够根据当前用户请求的读写特征,自动选取合适的处理路径以避免二者的不足;它还支持轻量级的事务处理框架,该框架按需实施以保证异构数据库数据的最终一致性和完整性。将MingleDB分别与MongoDB,MySQL数据库进行读写性能对比,实验证明了MingleDB方法的正确性和合理性。同时将MingleDB部署在实际的社交网络系统中进行实际验证,结果亦证明了其实用性和可移植性。

关键词 关系型数据库,NoSQL,异构数据库,中间件,事务提交
中图法分类号 TP392 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.05.027

Design and Implemention of Accessing Hybrid Database Systems Based on Middleware

PAN Ming-ming LI Ding-ding TANG Yong LIU Hai
(School of Computer,South China Normal University,Guangzhou 510631,China)

Abstract In the big data era,with the explosive growth of informational data,traditional relational databases(SQL) and emerging NoSQL databases are difficult to face these challenges in a comprehensive and efficient manner. Therefore,this paper proposed MingleDB,a high-efficient middleware for incorporating both merits of traditional database and NoSQL systems. MingleDB is transparent for the underlying hybrid database systems,namely SQL and NoSQL,without any intrusive modifications on the original systems. MingleDB can detect the specific characteristic inside a user query,for example,the data is either unstructured or structured. Then it puts this user query into the suitable procedure(SQL or NoSQL) presented by MingleDB,and also provides a lightweight transaction processing framework which is implemented on demand to ensure the final consistency and completeness of hybrid database data. Extensive experiments were conducted to test the effective of MingleDB. Furthermore,this paper used a realistic scenario to verify the advantage of our work. The results are positive.

Keywords Relation database,NoSQL,Hybrid database,Middleware,Transaction commitment

1 引言

在大数据时代,全球信息化数据爆炸式增长。传统关系型数据库在应对此类变化时表现出查询效率较低、扩展能力不足等性能缺陷。尽管可以从水平方向上用分布式策略来加入额外的计算与存储能力以应对此变化,但维护其一致性协议较为复杂,且投入管理开销较大。因此,更具性价比的解决方案——NoSQL数据库应运而生。MongoDB是新兴的NoSQL数据库,能在不同的数据场景下满足不同规模的用户查询请求。但其在面临强事务操作和数据一致性要求较高的场景时,存在先天缺陷。

基于此,本文提出一种名为MingleDB的混合数据库软件结构,其主要包括底层存储架构、中间件和事务处理框架。底层混合数据库由MySQL和MongoDB数据库组成,MongoDB只存储了MySQL中相应的非结构化数据副本。中间件可以在不更改数据库复杂源码的前提下,透明融合NoSQL数据库和传统数据库的主要运行逻辑;同时能根据当前用户请求的读写特征,自动选取合适的处理路径以避免二者的不足;还能支持事务特性。事务处理框架使用按需实施的策略将逻辑相关的一组操作绑定在一起,以保证基于混合数据库

到稿日期:2017-03-16 返修日期:2017-05-04 本文受国家自然科学基金(61502180),广东省自然科学基金(2014A030310238,2015B010129009,2016A030313441),广州市珠江科技新星项目(201710010189),广州市面向大数据安全产业链的协同创新项目(201508010067),广东省省级科技计划项目(2015B010109003)资助。

潘明明(1994-),女,硕士生,主要研究方向为数据库,E-mail:panmingming@m.scnu.edu.cn;**李丁丁**(1982-),男,博士,讲师,主要研究方向为计算机系统结构,E-mail:dingdingli@m.scnu.edu.cn(通信作者);**汤 庸**(1964-),男,教授,博士生导师,主要研究方向为学术社交网络和协同教学软件,E-mail:ytang@m.scnu.edu.cn;**刘 海**(1974-),男,博士,副教授,主要研究方向为数据挖掘和机器学习。

的服务器数据的最终一致性和完整性。最后，本文将 MingleDB 部署在实际的社交网络系统中进行实际验证，结果证明了 MingleDB 方法的实用性和可移植性。

本文第 2 节本具体阐述了 MingleDB 软件结构的分析和设计；第 3 节分析了学者网[1]的数据特点和访问性能缺陷，并将 MingleDB 方法应用于其中；第 4 节从 MySQL、MongoDB 以及 MingleDB 的查询事务性能 3 个方面进行对比分析；第 5 节介绍相关工作；最后总结全文。

2 MingleDB 系统的分析和设计

如图 1 所示，MingleDB 软件结构包含了两种数据库类型：1）传统的关系型数据库，它以 MySQL 为实例，提供事务特性支持；2）NoSQL 数据库，它以 MongoDB 为 NoSQL 的具体系统实现，提供高效的半结构化和非结构化的数据访问路径。此外，MingleDB 还在上述两种异构数据库的边界封装了虚拟软件层[2]，也即中间件，以达到屏蔽底层数据库的异构差异性、提供统一的数据库访问接口的目的，同时向上层大数据应用提供有效的控制策略和应用程序编程接口。

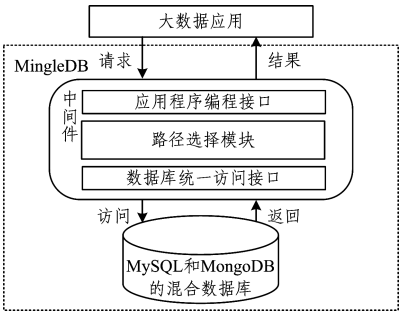

图 1 MingleDB 的系统结构

Fig. 1 System architecture of MingleDB

2.1 MingleDB 的特性

2.1.1 支持事务特性

MingleDB 支持事务特性，并针对 MySQL 和 MongoDB 的混合数据库提出了一个特有的轻量级事务模型。该模型遵循“按需实施”的原则，当用户显式提交事务语义时，其才将相关的用户请求导入事务框架。通过该事务框架，MingleDB 将逻辑相关的一组操作绑定在一起，以保证基于混合数据库的服务器数据的最终一致性和完整性。

2.1.2 高效的用户查询性能

1）基于直接 I/O 的非结构化数据访问方法：MingleDB 通过使用基于 mmap()系统调用的内存映射[3]存储引擎，将非结构化的数据文件直接映射到内存中，以绕开内核的干预，因此避免了内存中数据在磁盘 I/O 上的开销，发挥了类似读缓存的作用。同时，其对写入数据进行协调重组等操作，这种机制可以适度地把随机写操作转换成顺序写操作，尽可能地提升底层存储设备的运转性能。2）细粒度的内存数据缓存结构：不同于基于表级的 MySQL 缓存粒度[4]，MingleDB 通过 MongoDB 的行（文档）级查询缓存来实现更优的内存利用率和更加轻量级的缓存管理方式。在大数据环境下，非结构数据没有呈现出过于明显的局部性访问规律和特征，相比于 MySQL 缓存的粗粒度，此类基于对象的缓存系统能够提升该场景下的内存效率，进而提升 MingleDB 的缓存性能。

2.1.3 对原有系统透明

MingleDB 结合了 NoSQL 和传统关系型数据库的优点，不仅支持事务特性，还能够提供高效的用户查询性能，同时支持两类数据库在其水平方向上的分布式，以实现计算和存储能力的动态伸缩性。这些特性的实现不需要修改现有系统的源码，而是通过对一层中间件施加具体的控制策略来实现。此中间件为 MingleDB 的系统核心，它利用了不同数据库的优势，在前端应用程序和后台异构数据库之间自动判断用户对数据请求的特性，选择合适的路径导向底层异构数据库。

2.2 MingleDB 的内部实现

MingleDB 的底层存储架构主要包含 MySQL 存储相关和 MongoDB 存储相关。MySQL 采用一个 Master 多 Slave 的主从架构，MongoDB 则采用自动分片的 shard 模式。MySQL 服务器群存储部署所有业务节点的数据，MongoDB 则存储业务中所有的非结构化数据，后者的数据存储范围为前者的子集。MongoDB 可以根据存储在某一特定 shard 上的负载状况对非结构化数据采取自动分片机制，以此来降低每个节点的压力。

2.2.1 MingleDB 中间件的处理流程

如图 2 所示，中间件的核心是路径选择模块。数据存储功能是把用户的原始数据存入底层数据库；非结构化数据分离功能是把 MySQL 数据库中的非结构化数据分离出来后再存入 MongoDB 数据库；数据查询功能是根据用户的查询请求（结构化/非结构化数据）来决定导向底层异构数据库的路径；事务提交功能是用户在进行写入操作时按需实施，以保证底层混合数据库的数据一致性和完整性。

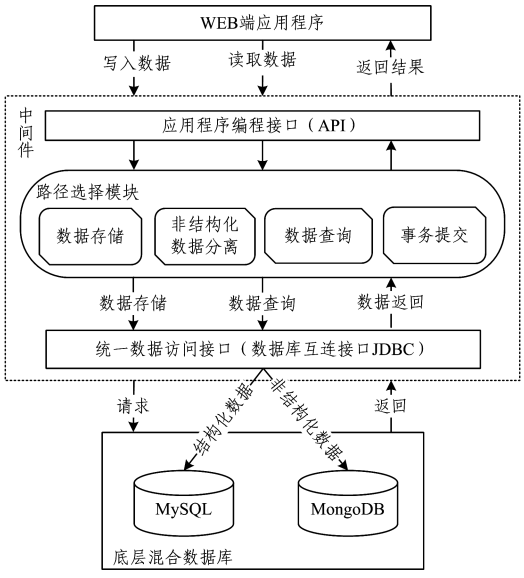

图 2 MingleDB 中间件的处理流程

Fig. 2 Middleware processing flow of MingleDB

当前端往后台数据库写入数据时，MingleDB 中间件开始进行路径选择：1）执行数据存储功能将该数据写入 MySQL 的存储引擎中。2）对该数据进行一系列同步操作，即执行非结构数据分离功能，将非结构化的数据全部从原始数据中剥离出来并存入 MongoDB 中，目的是加快之后的用户查询操

作，因为后者提供了更加优良的内存读取性能。3）选择事务提交路径以实现该写入操作（亦或事务提交）在异构数据库中的数据一致性，其中 MySQL 的存储引擎发挥了支持事务特性的作用。4）具体而言，当前端发来查询非结构化数据的请求时，中间件先访问 MongoDB 数据库，旨在通过此快速处理路径找到用户所需要的数据，达到比 MySQL 缓存性能更佳的处理流程。相比 MySQL 而言，MongoDB 在存储和管理海量非结构化数据方面有一定优势，可以提升非结构化数据在后台数据库中的内存命中率，优化整个系统中的任务吞吐量。

2.2.2　事务框架处理流程

如图 3 所示，为了保持事务的一致性，MingleDB 主要将 MySQL 数据库生成或更改的数据信息同步地写到 MongoDB 数据库中，同时检查和验证 MongoDB 上的相关操作是否执行成功。具体流程如下：在系统创建初期，MingleDB 同时构建 MySQL 和 MongoDB 数据库连接池，并向上层前端应用导出具有兼容性且简洁的访问接口；在接收到用户提交的事务 t 时，先连接 MySQL 连接池以获取到底层关系型数据库的相关数据，标记为 $Data_{trans}$；再连接 MongoDB 连接池，将 $Data_{trans}$ 冗余地写到 MongoDB 数据库中；上述过程皆成功后再向前端应用提交成功信息；若 t 提交失败，则 MySQL 立即回滚事务，而 MongoDB 则采用后期异步的方式清理脏数据，以提高后台数据库 MongoDB 的部分实时性能。

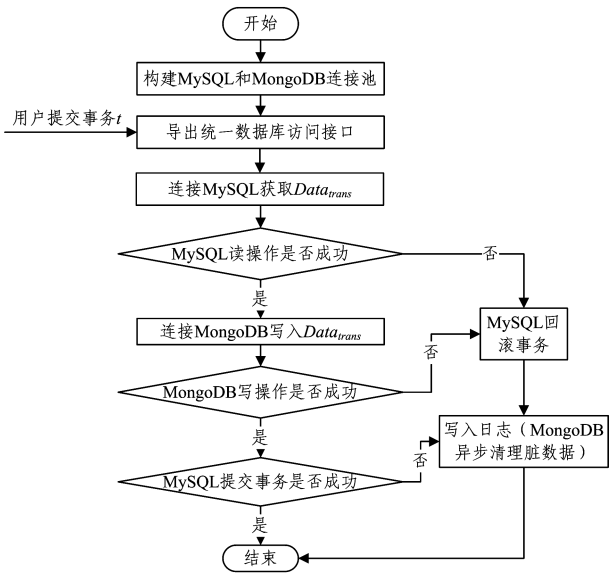

图 3　MingleDB 事务框架模型图

Fig. 3　Transaction framework of MingleDB

总之，MingleDB 采用冗余思想来应对用户的事务提交并维护异构数据库之间的数据一致性。此种方法逻辑清晰，实现简单。

2.3　相关问题讨论

（1）如何应对 MingleDB 查询缓存失效？

当 MingleDB 内部数据发生更新或者用户查询请求在内存中没有命中时，查询缓存的相应部分就会失效。1）MySQL 层次的缓存失效。若是由于用户查询请求未命中而导致，则 MingleDB 将此用户的请求直接传递至 MySQL 存储以寻找相关数据，将找到的结果返回给前端应用之前需同步至其查询缓存结构，以用于下一次的用户查询；若是由于数据库底层存储发生更新而导致的内存副本失效，则只需将此数据预读至内存以覆盖失效数据即可。2）在 MongoDB 层次的内存出现用户查询非结构化数据失效。该情况表明存在一个或者多个用户事务的提交对 MySQL 数据库的非结构化字段产生了更新或者插入，并且该变化还没有传递至 MongoDB 数据库以实现新数据的一致性，此时，MingleDB 进行显式同步，将此非结构化数据传递至 MongoDB 的内存子系统。

（2）如何缓解 MingleDB 的事务性能损失问题？

与传统的单一数据库结构相比，MingleDB 的事务提交框架不仅要将一次用户的事务提交（记为 t）持久化存储在 MySQL 的底层存储介质中，同时还要将其同步至 MongoDB 数据库，因而带来了额外的性能延迟。1）在默认的情况下，MingleDB 将 t 保存在 MySQL 中后，立即将 t 发送至 MongoDB 节点的内存，当 t 到达后，再向发出 t 的前端应用返回应答以确认。因此，MingleDB 无需等待 t 进入 MongoDB 的底层存储后再返回，从而避免了一定的存储开销。这种情况下，如果发生了 MongoDB 的节点崩溃事件（例如断电），则需要借助 MySQL 的存储将 t 再次转移至 MongoDB 的内存，以实现崩溃恢复机制。2）MingleDB 的事务提交受制于 MySQL 节点和 MongoDB 节点之间的通信延迟，这里可以采用 RDMA 技术进行缓解；另一种方法是利用系统虚拟化技术，将 MySQL 的运行例程和 MongoDB 例程封装在同一台物理机器上的不同虚拟机之内，再运用基于共享内存的域间通信技术来改善通信开销。

（3）如何进行数据恢复？

当混合数据库系统的某个或某些节点发生故障或遭受灾难时，需要对发生故障的节点进行数据一致性恢复。可以使用 MingleDB 捕获发送至混合数据库的所有事务请求，并以特定的格式存储到日志文件（包含 MySQL 和 MongoDB 对数据的所有操作）。若是事务故障，则反向扫描日志文件，对该事务中的修改操作执行撤销操作，直到事务开始标记。若是系统故障，则正向扫描日志文件，查找故障发生前已经提交的事务，将其标记为重做队列，同时查找还没有完成的事务，将其标记为撤销队列。若因介质故障导致数据丢失，则先重新安装混合数据库，再装入离故障发生时刻最近的转储副本，重新执行所有已完成的事务，从而实现各个故障节点的关键数据的恢复。

3　MingleDB 的场景应用

3.1　应用场景分析

学者网是一个学术社交网络平台，其存储处理的数据分为结构化数据和非结构化数据。其中，结构化数据一般为轻量级数据，特点是占用空间少、传输开销少。而非结构化数据是指不方便用数据库的二维逻辑来表现的数据，特征是存储空间和计算资源需求大。目前学者网中非结构化数据占 80% 以上。

目前，学者网使用 MySQL 的 InnoDB 存储引擎对这些数据进行统一的管理和存储，但它在处理大量消息文本和用户上传文件时性能较差。同时，学者网的用户增长量高于预期，

这对系统的扩展性也提出了一定的要求，因此相关开发和管理人员决定使用 MongoDB 来缓解上述问题。但 MongoDB 在事务处理方面只能达到弱一致性，其对于一些较为核心的数据并没有 MySQL 可靠，甚至会出现用户数据丢失的情况。因此，学者网恰恰是本文提出 MingleDB 混合数据库访问方法的动机和实践场所。

3.2 MingleDB 在学者网的具体实施

以学者网的课程相关表为例，“courses”存储课程表的基本信息（课程简介、大纲）；“course notice”存储课程公告的具体信息；“course resource”存储课程相关的资源文件。具体的非结构化数据分离的方法如下：将课程简介、大纲、公告、图片和文件等具备非结构化特征的字段从相关表中分离出来并存储到 MongoDB 中，同时把进行非结构数据分离操作后的表中的字段合并，去除冗余字段，生成一个新的表“course_sum”并将其保留在 MySQL 中，以供用户快速查询结构化数据。

为了保证事务的一致性，通过中间件冗余地在 MongoDB 中写入 3 张表的关键标识数据。此外，对于课程简介、大纲等信息类的非结构化数据，直接以“BSON”格式存入 MongoDB 中。对于课程图片、课程资源等文件类数据，使用 GridFS 格式进行存储，GridFS 中的“files”域存储元数据信息（图片或文件的基本信息），而具体的内容存储在“chunks”域。如果存储的文件过大，MingleDB 则会将一个文件分割成很多块，同时在“file”域中保存该文件块在原文件中的偏移。当“chunks”集合中存储了大量的数据时，MongoDB 使用自动分片机制来减小服务器的压力。

4 实验性能

实验主要分为 3 个部分。第 1 部分为 MingleDB 事务框架提交性能的分析：随着数据量的等量递增，比较了 MingleDB 和 MySQL 在百万级数据量下进行事务提交操作的时间开销。第 2 部分为 MingleDB 读取非结构化数据的性能分析：随着数据量的等量递增，比较了 MingleDB，MongoDB 和 MySQL 在百万级数据量下对非结构化数据进行读取操作的时间开销。第 3 部分为 MingleDB 读写混合操作的性能分析：将 MingleDB 应用在学者网上，并将其与之前单独使用 MySQL 管理学者网数据的读写操作进行对比，在此实验中，服务区将接收并发的写入和读取数据。3 个部分的实验结果如图 4－图 6 所示。测试环境为：Win10 Pro，6 GB 内存，i5 处理器，256 GB SSD，500 GB 硬盘，MongoDB3.2.9，MySQL 5.7.12。

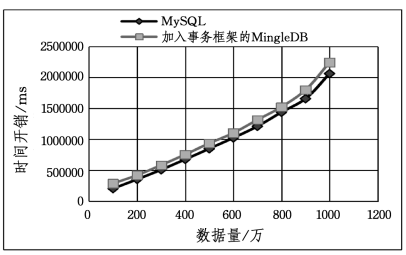

图 4 MingleDB 和 MySQL 事务提交性能的对比

Fig. 4 Comparison of transaction committing performance of MingleDB and MySQL

图 4 表明 MySQL 写入操作的事务提交性能随着数据量的增长而线性增长，但在接近千万级数据量时性能增长趋势放缓。加入了事务框架的 MingleDB 的时间开销与 MySQL 写入操作事务提交的时间开销相比略大，但是与 MySQL 的变化曲线几乎相同，这是因为 MingleDB 访问方法的中间件包含了事务提交的判断逻辑，其消耗了少许 CPU 资源，使性能稍有损失，但是在合理范围之内。

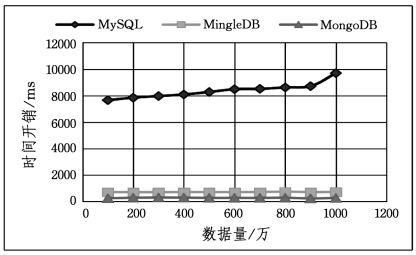

图 5 MySQL，MingleDB 和 MongoDB 对非结构化数据的读取性能对比

Fig. 5 Reading performance comparison of MySQL，MingleDB and MongoDB for unstructured data

图 5 表明对于非结构化数据，MingleDB 和 MongoDB 的读取性能较为稳定，尽管 MingleDB 的时间开销略大，但是两者接近。这是因为中间件包含了对读取数据的逻辑判断，使性能稍有损耗，但是二者的绝对数值优于 MySQL。同时，亦可发现在数据量接近千万级别时，MySQL 的读取性能并没有展现出扩展性，时间开销反而较大，这说明 MySQL 在处理海量数据时存在不足。

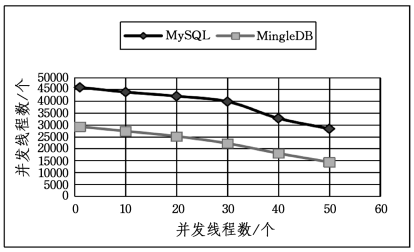

图 6 MingleDB 和 MySQL 应用在学者网上的性能对比

Fig. 6 Performance comparison of MingleDB and MySQL applied in scholar online

图 6 表明随着并发数的增加，MingleDB 和 MySQL 的用户请求完成数呈递减趋势，但是相对于学者网之前的 MySQL 存储方法，MingleDB 的读写性能得到了显著提高。这说明在应对突发性的海量数据的用户读写请求时，MingleDB 具备更好的性能扩展性。

综上所述，MingleDB 的事务提交性能与 MySQL 相似，对海量非结构化数据的读取性能接近于 MongoDB；将其部署在实际的学者网中，其读写性能相比于 MySQL 存储方法更高效。

5 相关工作

Yesquel[5] 存储系统可以提供 NoSQL 系统的性能和可扩展性，并且具备传统关系型数据库系统的事务特性。Yesquel

是为服务 Web 应用程序而设计的，不作为一般用途的数据库系统。与 MingleDB 相比，Yesquel 是一个崭新的数据库软件，尽管它是开源的，但是其可靠性还没有在实际场景中得到验证。

Unity[6]是一个集成和虚拟化的系统，允许 SQL 查询跨越多个数据库源，并且使用其内部查询引擎在不同的数据库源之间进行连接操作。MingleDB 与 Unity 的主要区别在于其轻量级的事务提交框架在理论上能够达到更优的事务性能，但是它不支持异构数据库之间的连接操作。

面向电子政务的 MongoDB 和 MySQL 混合云存储系统[7]采用云存储技术将 MySQL 和 MongoDB 结合起来，解决了传统电子政务使用关系型数据库 MySQL 存储管理非结构化数据的困难和扩展性能问题；并且利用了 MongoDB 对于海量的非结构化数据的存储和管理优势。与 MingleDB 的通用性相比，它更加适用于电子政务的垂直领域，其功能的灵活性略有限制。

结束语　本文提出了一种基于 NoSQL 和传统数据库的访问方法 MingleDB。这种数据库访问方法透明融合了 NoSQL 数据库和传统数据库的主要运行逻辑，既可以支持传统数据库特有的事务机制，又能利用 NoSQL 来实现更优的非结构化数据查询缓存性能，同时缓解了传统数据库进行水平扩展导致的成本过高问题，具备轻量级的事务处理流程，保证了融合数据库数据的最终一致性和完整性。将 MingleDB 部署在学者网上，利用学者网的实际使用场景来验证 MingleDB 的读写性能，并分别对比了基于 MongoDB 和 MySQL 的单一数据库结构的非结构化数据的读取性能和写入操作的事务提交性能。实验结果证明了 MingleDB 的优势。

参 考 文 献

[1] LI Y Y，TANG Y，HUANG Y H，et al. Research on Network Teaching Platform Based on Scholar's Social Model [J]. Computer Education，2015，252(24)：112-115. (in Chinese)
李宇耀，汤庸，黄泳航，等. 基于学者社交模式的网络教学平台研究[J]. 计算机教育，2015，252(24)：112-115.

[2] WU S，CHEN G，ZHOU X，et al. PABIRS：A data access middleware for distributed file systems[C]// IEEE International Conference on Data Engineering. IEEE，2015：113-124.

[3] AHN J S，SEO C，MAYURAM R，et al. ForestDB：A Fast Key-Value Storage System for Variable-Length String Keys[J]. IEEE Transactions on Computers，2016，65(3)：902-915.

[4] MARINO M D，LI K C. Last level cache size heterogeneity in embedded systems[J]. The Journal of Supercomputing，2016，72(2)：503-544.

[5] AGUILERA M K，LENERS J B，WALFISH M. Yesquel：scalable sql storage for web applications [C]// Proceedings of the 25th Symposium on Operating Systems Principles(SOSP'15). New York：ACM，2015：245-262.

[6] LAWRENCE R. Integration and Virtualization of Relational SQL and NoSQL Systems Including MySQL and MongoDB[C]// International Conference on Computational Science and Computational Intelligence. IEEE，2014：285-290.

[7] WU X J. Hybrid Storage Strategy with MongoDB and MySQL for E-government[J]. Computer and Modernization，2014(8)：62-66. (in Chinese)
吴秀君. 面向电子政务的 MongoDB 与 MySQL 混合存储策略[J]. 计算机与现代化，2014(8)：62-66.

(上接第 142 页)

[9] FU W，YAN B，WU X P. Data Possession Provability on Semi-trusted Cloud Storage [C]// Cloud Computing－ 4th International Conference. Wuhan，China，2013：199-209.

[10] AGRAWAL R，KIERNAN J，SRIKANT R，et al. Order preserving encryption for numeric data[C]// ACM SIGMOD International Conference on Management of Data. Paris，France：ACM，2004：563-574.

[11] NAVEED M，KAMARA S，WRIGHT C V. Inference Attacks on Property-preserving Encrypted Databases[C]// ACM Conference on Computer and Communications Security. Denver：ACM，2015：644-655.

[12] GENTRY C，HALEVI S. Implementing Gentry's Fully-Homomorphic Encryption Scheme [C]// IACR Cryptology ePrint Archive. 2010：520.

[13] MELCHOR C A，FAU S，FONTAINE C，et al. Recent advances in homomorphic encryption：A possible future for signal processing in the encrypted domain [J]. Signal Processing Magazine，IEEE，2013，30(2)：108-117.

[14] CHATTERJEE A，KAUSHAL M，SENGUPTA I. Accelerating sorting of fully homomorphic encrypted data [M]// Progress in Cryptology－ INDOCRYPT 2013. Mumbai：Springer International Publishing，2013：262-273.

[15] LIN H Y，TZENG W G. An efficient solution to the millionaires' problem based on homomorphic encryption[C]// International Conference on Applied Cryptography and Network Security. Springer-Verlag，2005：456-466.

[16] YAO A C. Protocols for secure computations[C]// Symposium on Foundations of Computer Science. Piscataway：IEEE Computer Society，1982：160-164.

[17] PRENEEL B. HMAC [M]// Encyclopedia of Cryptography and Security. New York：Springer，2005.

[18] DAEMEN J，RIJMEN V. The Design of Rijndael AES-The Advanced Encryption Standard [M]// Information Security and Cryptography. New York：Springer，2002.

[19] DIJK M，GENTRY C，HALEVI S，et al. Fully Homomorphic Encryption over the Integers [M]// Advances in Cryptology－ EUROCRYPT 2010. Springer Berlin Heidelberg，2010：24-43.