数据库新技术结课论文——NoSQL 数据库学习报告

郭宇航
202021080728

摘要

Web2.0 服务时代的大背景下，越来越多的数据以及越来越高的并发信息要求关系数据库进行水平的一些扩展以实现大数据时代下对于高性能实现的需求。在大规模数据和高并发性的应用程序的现实场景下，NoSQL 数据库技术应运而生。本文主要介绍了 NoSQL 的发展历史，理论的核心包括 CAP 理论，BASE 模型以及最终一致性; 常见的四类不同的 NoSQL 数据库：键值数据库，面向文档数据库，列族数据库以及图数据库并比较了它们之间的异同。最后部分分析了 NoSQL 这种数据库新技术的一些缺陷和不足以及未来的发展方向：NoSQL 拥有更好的可伸缩性和灵活性，但缺乏标准的查询语言和安全机制，其具有自己的优势同时也存在不小的挑战。

Abstract

Under the background of Web2.0 service era, more and more data and more and more concurrent information require some horizontal expansion of relational database to realize the demand for high-performance implementation in the era of big data. NoSQL database technology comes into being in the real world of large-scale data and highly concurrent applications. This paper mainly introduces the development history of NoSQL, and the core of the theory includes CAP theory, BASE model and final consistency property; four common different types of NoSQL databases are: key-value databases, document-oriented databases, column family databases, and graph databases and we compared the similarities and differences between the four different databases. The last part analyzes some defects and shortcomings of NoSQL database technology and its future development direction: NoSQL has better scalability and flexibility, but it lacks standard query language and security mechanism, so it has its own advantages and challenges.

NoSQL 应用需求产生的背景

在计算机数据存储领域一直以来都是关系数据库占据主导地位，关系型数据库(RDB)从 20 世纪七十年代发展至今，因为高效的管理系统的存在而非常容易使用和维护，同时其产生了广泛地应用[8]。然而由于大数据技术的推广和应用，一些大型网站遇到了关系数据难以克服的缺陷，海量数据处理能力以及僵硬的约束设计，越来越多的公司或者机构需要存储远远多于以前的数据同时还需要保持良好的获取速度。当场景出现需要获取多重关系的复杂信息时，关系型数据库往往需要进行大量的 SQL 的连接操作，这大大降低的数据库的效率。这时候越来越多的其他类型的数据库被提出来解决这方面的问题，NoSQL[7]是一个目前非常流行的数据库，广泛地被各类公司机构采用。在互联网 web2.0 时代，网站存储的数据类型发生了重大的变化，数据朝着弱结构化，多元化，海量化发展，网站需要应对高并发请求，例如 Facebook，twitter 等大型社交网站，每天产生上百 TB 的数据，热点数据的请求次数达到上百万次，单一的关系型数据库是完全无法满足这样的需求的，而新型的 NoSQL 数据库架构模式却能够胜任这方面的工作。NoSQL 数据库可以快速转换弱结构化甚至非结构化的数据[9]，并通过更灵活地替换“有组织的”存储来避免 SQL 的僵化。大量云计算的需求和互联网的发展推动了 NoSQL 的发展，大规模数据的存储以及高并发数据的处理问题是 NoSQL 解决的主要问题。

NoSQL 国内外研究现状

目前人们广泛接受的关于 NoSQL 的解释是“Not Only SQL”，这一表达强调了在 NoSQL 中有很多系统还是会支持类似 SQL 查询语言。NoSQL 至少在一开始是作为对 web 数据、对处理非结构化数据的需求和对更快处理的需求的响应而开发的。‘NoSQL’这一缩写最早是在 1998 年由 Carlo Strozzi[1]提出，他提出这一概率是用于命名自己开发出来的轻量级，开源的不使用 SQL 的关系型数据库。真正意义上的 NoSQL 的起源来自于 1991 年的一个开源数据库产品 Berkeley DB，这是一个键-值类型的数据库，使用 hash 表存储数据，这种类型的数据存储结构相对简单，当时主要应用于嵌入式系统中。现代 NoSQL 数据库的发展主要开始于 2007 年，2009 年 Johan Oskarsson 发起了一次关于分布式开源数据库的讨论，NoSQL 再次被提出，NoSQL 的概念发生了翻天覆地的变化[2]。从诞生至今 NoSQL 的数据库模型一直在爆炸性的增长中，目前已经有超过 255 个 NoSQL 的数据库模型。Apache 的 Cassandra 曾经一度是 Facebook 的专用数据库，它于 2008 年开

源发布，其他的数据库包括谷歌的 BigTable，Apache 的 Hadoop，MapReduce 的 MemcacheDB 等十分流行，各类 NoSQL 数据库相互竞争同时相互促进。国外使用 NoSQL 的大型公司包括 Netflix，LinkedIn，Twitter 等等。Google[2]在 BigTable 中提出了一个有利于多列历史数据排序存储的数据模型，这种数据模型中的数据具有严格的一致性，通过分层基于范围分区方案分发到多台服务器上；亚马逊的 DynamoDB 是一个完全托管式的 NoSQL 数据库服务，可以提供快速的可以预期到的性能，并且能够实现无缝扩展，其数据模型较为简单，按照键-值的映射关系到达数据应用程序特定的分区，分区模型的失败更具有一定的弹性。在国内，NoSQL 还处于快速发展的阶段，很多公司都在研究适合自己公司业务场景的 NoSQL 产品：阿里云集团是最早提出数据战略的公司机构，也是最早投入 NoSQL 数据库计数研发的，阿里云 NoSQL 目前覆盖了所有主流的 NoSQL 数据库，如 Redis，MongoDB，HBase 等等，在国内阿里云 NoSQL 数据库的多项独家关键技术领先竞争对手，是国内 NoSQL 数据库的排头兵；国内知名豆瓣网自主研发了针对于大数据，高可用的分布式键-值存储系统 BeansDB[8]，其采用 HashTree 等实线最终的一致性，其最大的特点是具有高可伸缩性，扩展数据集群较为方便。在上述背景下，NoSQL 数据库目前正处于蓬勃发展时期，未来还会有更加丰富且完善的数据库模型出现。

NoSQL 数据库技术核心

CAP, BASE 和最终一致性是 NoSQL 数据库存在的三大基石。 20 年前 Eric Brewer[11]教授提出了著名的 CAP 理论，后来 Seth Gilbert 和 Nancy lynch[12]两人证明了 CAP 理论的正确性。CAP：C：consistency 一致性；A：availability 可用性（指的是快速获取数据）；P：Tolerance of network Partition 分区容忍性（分布式）。CAP 理论核心理论告诉我们一个分布式的系统不可能满足一致性，可用性以及分区容错性这三个需求，最多只能够满足其中两个。

BASE[3]表示的是：Basically Available：基本可用；Soft-state：软状态或者说柔性事务（可以理解为无连接的，hard state 表示面向连接的）；Eventual Consistency：最终一致性。

最终一致性[6]是一个非常重要的概念，它将会贯穿 NoSQL 的研究过程，一言以蔽之，最终一致性表示的是过程松，结果紧，最终结果必须要保持一致性。为了更好的描述客户端一致性，我们通过以下的场景来进行，这个场景中包括三个组成部分：存储系统：存储系统可以理解为一个黑盒子，它为我们提供了可用性和持久性的保证，process A 主要实现从存储系统 write 和 read 的操作，process B 和 process C 都是独立于 A 的且 B,C 之间也相互独立，他们同时也实现对于存

储系统的读写操作。强一致性表示：假设 A 写入值到系统中，存储系统保证后续的 A,B,C 读取操作都是返回最新的值。弱一致性：假如 A 先写入了一个值到存储系统，存储系统不能保证后续 A,B,C 的读取操作能读取到最新值。此种情况下有一个“不一致性窗口”的概念，它特指从 A 写入值，到后续操作 A,B,C 读取到最新值这一段时间。最终一致性：最终一致性是弱一致性的一种特例。假如 A 首先 write 了一个值到存储系统，存储系统保证如果在 A,B,C 后续读取之前没有其它写操作更新同样的值的话，最终所有的读取操作都会读取到最 A 写入的最新值。

BASE 模型[4]反 ACID 模型，完全不用与 ACID 模型，牺牲高一致性，获得可用性和可靠性。数据库状态可以存在某段时间不同步，是异步的，而最终一致性保证了最终的数据一致即可，而不需要时时一致。BASE 思想主要强调基本的可用性，如果你需要高可用性，也就是纯粹的高性能，那么就要以一致性或容错性为牺牲，BASE 思想的方案在性能上还是有潜力可挖的。

NoSQL 的基本特征[7]：NoSQL 数据库不需要预定义的表模式，通常横向的扩展可以避免连接操作。由于 NoSQL 系统的模式较少且涉及更小的子集分析，因此该数据库可以更好地描述为结构化数据存储。NoSQL 数据库的三个重要基本特性是向外扩展性、灵活数据结构性和可复制性。下面详细的给出一些解释。向外扩展性是指通过使用许多通用机器在分布式环境中实现高性能。NoSQL 数据库允许在具有分布式处理负载的大量机器上分布数据。许多 NoSQL 数据库在添加到集群时允许将数据自动分发到新机器。我们一般通过可伸缩性和弹性来评估数据库的向外扩展性；数据结构的灵活性对于 NoSQL 数据库而言就是说我们不需要为数据库定义一个模式，NoSQL 不需要预先定义一个设计好的模式，它允许用户在同一个数据库表格中存储不同结构类型的数据。但是大多数的 NoSQL 数据库不像 SQL 一样，大多数情况下它是不支持高级查询语言的；最后一个典型特征：数据复制性，数据复制性是指在这个过程中，数据的一个副本被分发到不同的系统，以实现冗余和负载分布。

NoSQL 数据库主要分类及其对比

（1）面向高性能并发读写的 Key-value 数据库[5]：key-value 数据库的主要特点就是具有极高的并发读写性能，Redis, Tokyo, Cabinet, Flare 等就是这类的典型。在 key-value 存储中每个数据由一对唯一的键和值组成，同时为了存储数据应用程序需要生成一个键，值与键之间相互关联。这个键-值对被提交到数据存储。存储在键值存储中的数据值可以有附加到其上的动态属性集合，并且对于数据库管理系统而言是不透明的。因此键就是访问数据库中数据值的唯一方法。从键到值

的绑定类型取决于应用程序中使用到的编程语言。应用程序需要向数据存储提供一个键，以便检索数据。许多键-值数据存储使用散列函数。应用程序对键进行散列并找出数据在数据库中的位置。键-值数据存储以行为中心。这意味着它使应用程序能够检索完整实体的数据。key 的设计应该支持在数据存储上触发的最频繁的查询。（有点类似 cache 块的意思）

下面给出一个 key-value 数据库的实例：如图 1 所示，假设应用程序为数据存储指定了一个键值“A102”用于检索数据，应用程序使用 hash 函数对于键值进行散列处理，从而追踪需要的数据在数据库中的具体位置。hash 函数的效率、键的设计和存储的值的大小是影响键值数据存储性能的因素。我们不难发现在这些数据存储上执行的操作大多限于读和写操作。由于键-值数据存储的简单性，它为用户提供了存储和获取数据的最快方法。所有其他类别的 NoSQL 都构建在键-值数据存储的简单性、可伸缩性和性能之上。

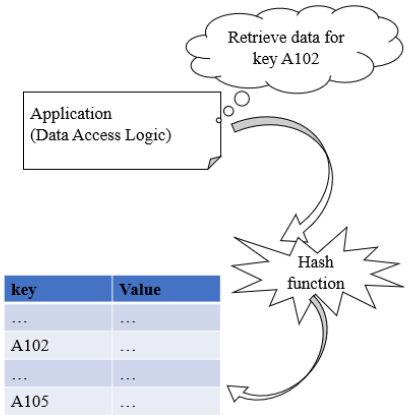

图 1：键-值对数据存储实例

（2）面向海量数据访问的面向文档数据库：这类数据库的特点就是可以在海量的数据中快速地查找数据，典型代表为：MongoDB 和 CouchDB. 从一个非常抽象的角度来说，面向文档类的数据库是与键值对数据存储十分相似的。面向文档类数据库也存在很多的 values，这个 value 对应于键-值数据库中的值，同样的一个应用程序需要读取数据需要使用一个 key. 在这类数据库中有不少是在创建一个新文档时自动生成唯一的 key. 文档数据库中的文档是一个实体，它是命名字段的集合。而面向文档数据库与键-值对数据库之间的最大差别在于数据对于数据库的透明度。也就是说在面向文档数据库中，查询的可能性并不仅仅局限于键。为了支持不同的应用场景下应用程序的查询，检索时不仅需要根据键查询数据库，还需要根据属性值查询，这个在文档数据库中是可以进行切换的。在面向文档数据库中，一个文档是自描述的，也就是说文档的信息存储是存储在一个非常便携的且容易理解的格式，例如：XML, BSON, JSON 等。图 2 展示了一种

基本的面向文档数据库的实例。文档数据库按照键值对存储数据。但是应用程序在获取每个数据值的时候，不仅仅可以通过 key 去访问，也可以通过数据中定义的一些属性来访问，例如：FirstNm, LastNm 等等。

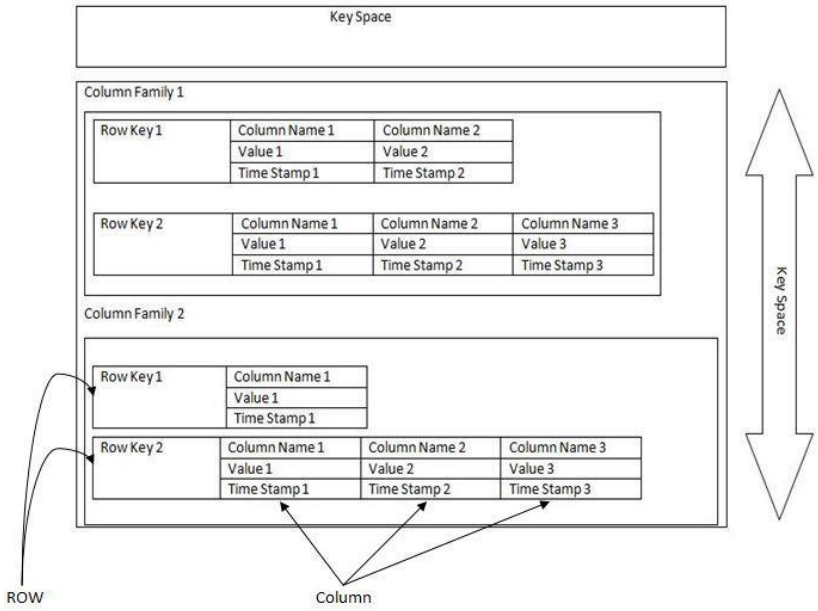

图 2：面向文档数据库实例

（3）列族数据库(Column Family Data Stores) 有时候如果一个应用程序希望读取或者检索一个字段的部分子集，就类似于 SQL 查询语言中的投影操作。列族数据存储支持以列为中心的方式存储数据。列族数据存储对键空间进行分区。在 NoSQL 中，键空间被认为是一个对象，它将一个设计的所有列族放在一起。它是数据存储中最外层的数据分组。键空间的每个分区都是一个表。列族由这些表声明且每个列族由若干列组成。在列族中的每一行都是被结构化为任意数量的列的集合。每一个列都是一个键值对的映射。在这个映射中键是每一列的名称，

Key (ID Number)	Document
A101	FirstNm: Jeet LastNm: Sethi Age: 26
A102	…
…	…
…	…
…	…
A705	FirstNm: Meera LastNm: Patnaik Age: 22

图 3：列族数据库存储模式

同时值是每一列自己。每一个这样的映射我们都称之为是一个 cell，在列族中的每一行都被认为是唯一的 row-key，这个 row-key 由每一个在数据库中进行查询

的应用程序定义。使用这些 row-key 使得我们的数据检索操作更加快速。为了避免覆盖每一个 cell 中的值，一些常见的列族数据库会自动向各个列添加时间戳信息：每一次存在一个更新就会创建一个新版本的 cells 表示这些 cells 在进行更新值的操作之后的结果。同时用户读到的数据永远是最后一次写入到或者提交到数据库中的信息。因此综上来看，列族数据库中的 key 由一个 row-key, 列族，列以及时间戳构成。图 3 给出了列族数据库的基本结构。

与传统数据库相比，使用列族数据存储的一个优点是处理 NULL 值。在关系数据库中，当属性的值不适用于特定行的时候，将存储为 NULL。而在列族数据库中当一个数据不可使用时，直接简单的删除对应行的列即可。因为这样的特性，Google 将其称为是一个稀疏的数据库。这个数据库的关键特性之一是，它可以以十亿单元的形式分布在数千台机器上。所有的 cells 都是根据 row-key 排序的。键排序允许搜索一系列键的数据。由于这种模型中的数据被组织为一组行和列，因此该数据库的表示方式最类似于关系数据库。这个数据库在运行时，可以灵活地添加行和列，但通常必须预先定义好所有的列族，这导致数据存储不如键值或文档数据存储灵活。一个设计良好的列族数据库应该具有这样的属性：应用程序能够通过访问尽可能少的列族来满足大多数查询。与保存同等数量数据的关系数据库相比，列族数据存储具有更强的伸缩性和更快的速度。HBase[10]与 Hypertable 数据库系统就是基于这样的列族数据库模型构建起来的。

（4）图结构数据库(Graph Database) 图数据库被认为是高度关联数据最好用的数据库之一。它用于解决存在大量相互关系的数据。理解图数据库需要考虑三个基本的元素：节点，连边以及属性。其中边表示的是实体之间存在的关系。每个关系都有一个关系类型，并且具有一个起点和一个终点。终点可以是起点之外的某一个节点当然也可以是同一个节点。键-值属性不仅与节点关联，而且与关系关联。关系的属性提供了关于关系的附加信息。关系的方向决定了图形数据库中从一个节点到另一个节点的遍历路径。图 4 给出了一种图数据库的基本模型。

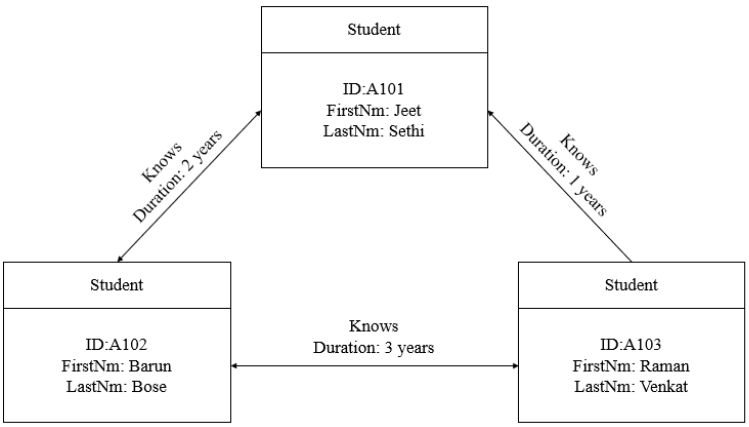

图 4：图数据库实例

上述介绍了多种不同类别的 NoSQL 数据库，而在实际应用种没有任何硬性规则来决定哪个 NoSQL 数据库最适合企业或者某个机构来进行使用。商业模式、战略、成本和交易模式需求是在选择数据库时需要考虑的几个重要因素。如果应用程序只是使用键作为标识符来存储和检索数据库管理系统以及不透明的数据项，那么键-值存储是最好的选择。当应用程序的选择性更强，并且需要根据非关键字段过滤记录，或者检索或更新记录中的单个字段时，文档数据库是一种有效的解决方案。文档数据存储提供了比键-值数据存储更好的查询可能性。当应用程序需要存储具有数百或数千个字段的记录，但在执行的大多数查询中检索这些字段的一个子集时，列族数据存储是一种有效的选择。这样的数据存储适合大规模的大型数据集。如果应用程序需要存储和处理实体之间关系高度复杂的重关联数据信息，则图数据库是最佳选择。在图数据库中，实体和实体之间的关系被同等重视。

下表 1 给出了一些关于常见的数据库的一些属性特征[9]。

表 1：不同 NoSQL 数据库之间的比较

Database Tool	Data Model	Transaction Model(CAP)
DynamoDB	Key-value	AP
Riak	Key-value	AP
CouchDB	Document	AP
Tokyo Cabinet	Key-value	PC
Redis	Key-value	AP
MongoDB	Document	AP
Hbase	Column-family	PC
Neo4j	Graph	CA

对于 NoSQL 数据库而言，根据不同的需求其产生了不同的类别但是在所有的 NoSQL 数据库的背后都存在着许许多多的共通原则。首先 NoSQL 数据库都是假设存在失效的情况发生的（根据 CAP 原理以及最终一致性的特点）。因此他们都采用对数据进行分区的策略，通过对数据进行分区处理能够尽可能地最小化存在失效而带来的影响，同时这样做的另一个好处就是将读写操作的负载分布到了不同的机器上，集群中如果存在一个节点失效了那么只有该节点存储的数据会受到影响，而不会是全部的数据。除此之外，大部分 NoSQL 实现都是基于数据副本的热备份来保证连续的高可用性。其中一些可以控制备份的数量，备份的位置，控制副本的同步还是异步等等，从而我们不必再每个节点上都保存太多的副本。要掌控不断增长的数据，大部分 NOSQL 实现提供了不停机或完全重新分区的扩展集群的方法。一个已知的处理这个问题的算法称为

一致哈希。有很多种不同算法可以实现一致哈希。一个算法会在节点加入或失效时通知某一分区的邻居。仅有这些节点受到这一变化的影响，而不是整个集群。有一个协议用于掌控需要在原有集群和新节点之间重新分布的数据的变换区间。另一个方法是使用逻辑分区，这种方法的优势在于是可预测的同时是一致的，使用一致哈希算法可能会导致分区之间的重新分布并不平稳，当一个新的节点添加到网络中时可能会消耗很多的时间。逻辑分区的缺点在于可伸缩性受限于逻辑分区的数量。

NoSQL 数据库存在的问题

相对于关系型数据库而言，NoSQL 存在的问题主要是在其没有关系数据库所具有的可靠性功能（因为其不支持 ACID），这也就意味着 NoSQL 数据库在性能和可伸缩方面提供了一致性。为了支持 ACID 一些开发人员必须要实现自己的代码，这使得他们的系统更加复杂。这可能会减少提交事务的安全应用程序的数量，例如在银行系统中，NoSQL 和 SQL 之间根本不能够兼容。另外一方面现在有部分的 NoSQL 数据库是支持结构化查询语言的，但是还有大多数都是不支持的[6]，因此在使用这类数据库的时候往往需要手动查询语言，这样往往会降低效率。同时 NoSQL 的稳定性目前相较于关系型数据库还有一些差距，另外在面对关键功能出现障碍时一些常见的关系型数据库往往能够确保数据管理系统的及时恢复，但是 NoSQL 目前的整体生态链还没有完全构建起来，很多一些意外事故的处理服务机制还不够完善。NoSQL 数据库是根据现在 web2.0 的应用程序的需求而构建起来的，因此大部分的功能都是为了满足这些需求，但是当新的数据应用程序的需求超过 “插入-读取-更新-删除”的循环外很少提供到一些专门的分析和查询等某些方面的功能。

结论

本文中我们介绍了 NoSQL 数据库产生的时代背景，NoSQL 产生的一些理论基础以及常见的类别。我们知道关系型数据库的特性特别适合处理数量有限的结构化数据。而 NoSQL 的特性是根据其可伸缩性，高性能而设计。其灵活性的特点使得在面对海量数据以及高并发的情况能够高效地处理事务。然而 NoSQL 的生态链，体系的完备性还没有达到关系型数据库那么完善，同时不像关系型数据库的独有的标准查询语言 SQL 语言，因此这将会是未来 NoSQL 发展的趋势和方向。

参考文献

[1] Jeang-Kuo Chen,Wei-Zhe Lee. An Introduction of NoSQL Databases Based on Their Categories and Application Industries †[J]. Algorithms,2019,12(5).

[2] Cornelia Gyorödi,Robert Gyorödi,Roxana Sotoc. A Comparative Study of Relational and Non-Relational Database Models in a Web- Based Application[J]. International Journal of Advanced Computer Science and Applications,2015,6(11). Amandeep Kaur,Kanwalvir Singh. Study of Various NoSQL Databases[J]. Research Cell: An International Journal of Engineering Sciences,2016,21(0).

[3] Kunda D, Phiri H. A comparative study of NoSQL and relational database[J]. Zambia ICT Journal, 2017, 1(1): 1-4.

[4] Chandra D G. BASE analysis of NoSQL database[J]. Future Generation Computer Systems, 2015, 52: 13-21.

[5] Martins P, Abbasi M, Sá F. A study over NoSQL performance[C]//World Conference on Information Systems and Technologies. Springer, Cham, 2019: 603-611.

[6] Chen J K, Lee W Z. A study of NoSQL Database for enterprises[C]//2018 International Symposium on Computer, Consumer and Control (IS3C). IEEE, 2018: 436-440.

[7] 仝野. 基于 NoSQL 数据库的系统设计与开发[D].南京邮电大学,2018.

[8] 杨阳. 基于 NoSQL 的数据分析技术的应用研究[D].东南大学,2016.

[9] 肖光昭. 基于 SQL 和 NoSQL 的混合存储系统的设计与实现[D].北京理工大学,2016.

[10] 梁力源. 基于 NoSQL 存储系统的研究与应用[D].重庆交通大学,2016.

[11] Brewer E A. Towards robust distributed systems[C]//PODC. 2000, 7(10.1145): 343477.343502.

[12] Gilbert S, Lynch N. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services[J]. Acm Sigact News, 2002, 33(2): 51-59.