移动群智服务 HelPal 数据管理系统设计与实现

摘要

在这个大数据已经兴盛且正在快速发展的时代，大数据的概念已经渗透了几乎各行各业。“大数据”这三个字可以被理解为是传统的数据处理工具无法处理的大量复杂数据的集合。当下大数据所面临的挑战包括数据分析、数据处理、数据查询、数据存储、数据可视化、数据隐私等。面对这些挑战，目前市面上也存在许多各种各样的工具或系统，它们从不同的侧重点切入去解决这些问题。在该论文中，我将构建一个名字为“Datar”的架构，它是从系统架构的角度出发，类比计算机结构而形成的一个大数据管理系统的统一架构。文中，我将总结大数据管理系统的发展历史和目前发展状态，同时分别阐述 Datar 的五个关键组成部分：数据输入、数据存储、数据计算、数据控制、数据输出。最后，我将用代码实现 Datar 模型的一个实例，命名为 HelpalDB，作为 Helpal 应用的数据管理系统，并通过示例详细描述它的操作细节。该模型展现了一个构建大数据管理系统的统一的途径，它从独特的角度看待大数据管理系统的结构，并构建大数据管理系统。

关键词：大数据管理系统　数据输入　数据存储　数据计算　数据控制　数据输出

Abstract

Big data is emerging and reaching every corner of industries and every field of researches. Big data can be interpreted as a term for large or complex datasets that traditional data processing application are inadequate. Challenges in big data include data analysis, data transaction, data query, data storage, data visualization, data privacy and so on. Various tools and systems are proposed and developed to face these challenges on different emphases. In this paper, I propose "Datar", a new prospective and unified architecture for Big Data Management System (BDMS) from the point of system architecture by leveraging ideas from computers. I introduce five key components of Datar, which are data input, data storage, data computation, data control and data output, by reviewing the current status of BDMS. Moreover, the architecture of Datar is presented as an implementation called "HelpalDB", which is supposed to be the data management system of HelPal application. Manipulation detail of HelpalDB are demonstrated by several examples. This paper shows the envisioned Datar as a feasible solution and unified architecture that can manage big data automatically and intelligently with specific functionalities.

Key Words: BDMS Input Storage Computation Control Output

内容目录

1. 背景介绍 1
1.1 从 Computer 到 Datar 1
1.2 什么是 Datar 2
1.3 Datar 的发展前景 3
2. 大数据管理系统的五个组成部分 4
2.1 数据输入 4
2.1.1 数据生成 4
2.1.2 数据传送 4
2.2 数据存储 5
2.2.1 原始数据存储 5
2.2.2 索引存储 7
2.3 数据计算 7
2.3.1 数据查询 7
2.3.2 数据分析 8
2.4 数据控制 8
2.4.1 数据事务 9
2.4.2 数据恢复 10
2.4.3 资源管理 10
2.5 数据输出 10
2.5.1 数据可视化 11
2.5.2 数据分享 11
3. 大数据管理系统的主流解决方案及对比 11
3.1 主流的大数据管理系统解决方案 11
3.1.1 AsterixDB 11
3.1.2 BDAS 12
3.1.3 小米数据平台 12
3.1.4 TiDB 13
3.1.5 Apache Beam 14
3.2 各方案与 Datar 的对比 14
4. 实现 Datar 实例 —— HelpalDB 16
4.1 Datar 的前提与假设 16
4.2 Datar 的架构和它的实例 HelpalDB 16
4.3 HelpalDB 使用实例 18
4.3.1 HelpalDB 的安装和管理 18
4.3.2 HelpalDB 实例 hpdb 的操作 19
5. 结论和未来展望 21
参考文献 22

图表目录

图 1 大数据管理系统的一种架构..... 1
图 2 计算机和 Datar 的结构对比..... 2
图 3 数据存储模型比较..... 6
图 4 CAP 理论..... 8
图 5 AsterixDb 结构..... 12
图 6 BDAS 结构..... 12
图 7 小米数据平台结构..... 13
图 8 TiDB 结构..... 13
图 9 Apache Beam 运行流程..... 14
图 10 HelpalDB 的定位..... 16
图 11 HelpalDB 框架结构..... 17
图 12 几个关键概念之间的联系..... 18
图 13 HelpalDB 的配置文件..... 18
图 14 安装和管理 HelpalDB..... 19
图 15 HelpalDB 实例 hpdb 的操作示例..... 20
图 16 hpdb 中 D3 实现的可视化效果..... 20

表 1 常见的不同类型大数据存储系统..... 7

1. 背景介绍

1.1 从 Computer 到 Datar

众所周知，图灵对计算机的人工智能进行了突破性的探索，发表了论文“Can Machines Think？”0，并设计了著名的“图灵测验”。其实早在 1945 年，在图灵的标志性探索之前，冯诺伊曼就对计算机进行了工程性的研究，并基于当时存储程序的概念，提出了计算机的一种逻辑设计，也就是著名的“冯诺伊曼体系结构”[2]。而更早之前，英国著名数学家和机械工程师 Charles Babbage[3]就曾提出差分机与分析机的设计概念。这些先驱者的目标都是要设计一个可以媲美人脑的计算机器，以节约人工成本。

在过去的几十年里，数据管理中的一些概念比如逻辑独立性，物理独立性，和基于代价的优化已经开始主导一些研究方向，并给一些产业带来了繁荣的前景。这些技术上的进步同时也促进了第一批商业智能应用的产生。面对如今大数据相关的新颖的挑战和机会，我们有必要重新思考大数据管理平台的方方面面，并将之前的概念与如今的发展相结合。 英国关系数据库之父 E. F. Codd 和美国计算机科学家 Charles Bachman 分别在理论[4]和实践[5]上为数据库的发展奠定了基础，许多前辈也在数据管理的方向做出了很多贡献，包括 Ingres[6]，Postgres[7]，Mariposa[8]，C-Store[9]和 VoltDB[10]。

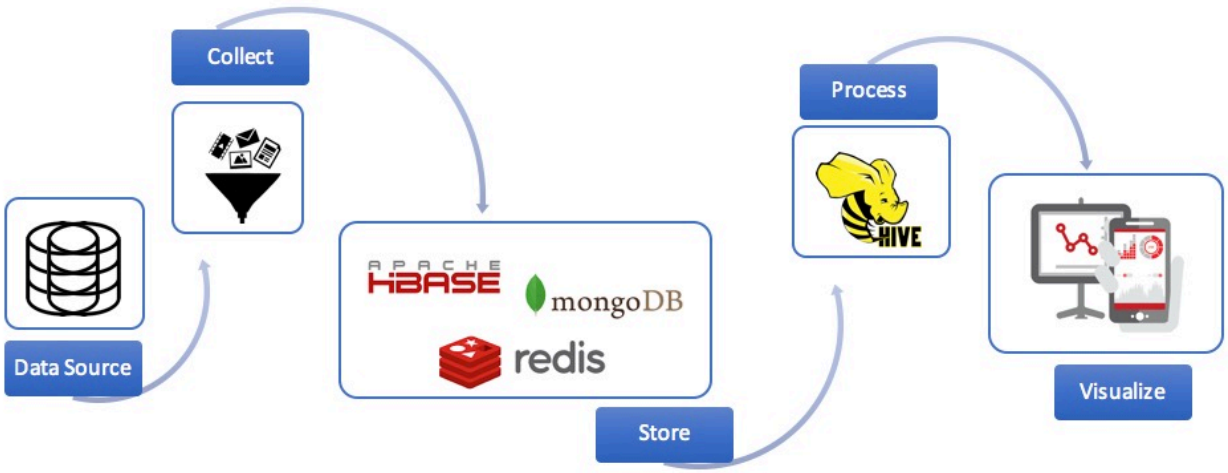

图 1 大数据管理系统的一种架构

随着时代进步，机器的计算能力逐渐变强，我们也逐渐发现了通过大数据获取更多信息的研究重心逐渐从计算到管理的转变。同之前一样，从计算机到大数据管理系统，我们的目标始终都是将人类从复杂工作中解救出来，以节约人工成本。正因如此，大数据管理系统（BDMS：Big Data Management System）应运而生。笼统来看，大数据管理系统可以看作是一系列复杂功能的集合体，包括数据收集、数据存储、数据处理和数据可视化等。对比之前传统的数据库系统，面对当下庞大的数据量和普遍的数据管理需求，我们有必要提出一个统一的架构去指导大数据管理系统的设计与开发。这个统一的架构应当是更加灵活且开源的，能够针对不同需求提供不同的功能侧重点。在本论文中，我将提出一个统一的架构，命名为 Datar。

我们都知道，典型的计算机结构可以被分为五个部分：输入、存储、计算、控制和输出，其中计算是中心。如果我们仔细观察图 1 所示的大数据管理系统结构，我们可以发现大数据管理系统的架构和计算机有着相通之处：BDMS 的数据输入对应计算机的输入；BDMS 的数据存储对应计算机的存储；BDMS 的查询和分析对应计算机的计算；BDMS 的事物系统对应计算机的控制机制；BDMS 的数据输出对应计算机的输出。大数据管理系统的五个部分中，数据存储是中心，即“Data”是中心，这也是它“Datar”名字的由来。图 2 可以帮助人们更好地理解 Computer 和 Datar 之间在结构上的相同与不同之处。在 2（a）中，计算机的五个核心部分分别由五个矩形表示，在 2（b）中，五个矩形分别代表大数据管理系统的五个部分。通过图 2 可以很直观地感受到 Computer 和 Datar 之间五个部分的对应关系以及它们不同的侧重部分。

（a）计算机结构　　（b）Datar 结构

图 2 计算机和 Datar 的结构对比

1.2　什么是 Datar

定义（Datar）Datar 是一系列具有特定功能的相关软件的集合体，它可以被用于从持久数据中自动挖掘有价值的信息，其中特定功能指的是大数据的输入、存储、计算、控制和输出。在该论文中，我将用一个简单的实例实现 Datar，作为 Helpal 产品的数据管理系统，命名为“HelpalDB”。它的组成部分包括 AsterixDB[11]及其扩展功能、Spark 和 D3。它是一个以数据存储为中心的 Datar 的实现。

简而言之，Datar 就是一个功能齐全的大数据管理系统的统称，它由五部分组成：数据输入、数据存储、数据计算、数据控制和数据输出。与以计算为中心的计算机相比，Datar 以数据为中心。以下以 AsterixDB 为例，它也是一个功能齐全的大数据管理系统。数据输入指的是数据如何进入系统，比如在 AsterixDB 中，数据输入由 Data Feed（数据传送）来实现，Data Feed 是它的一个内置机制，它可以不断地将数据从数据源传送到一个或多个数据集当中，逐渐填充各个数据集及补充相关索引[12]；数据存储是指数据如何存储在系统中，以及它们的索引如何建立，比如在 AsterixDB 中，数据和索引的存储基于 LSM 结构 [13]；数据计算是指如何从存储的数据中提取有价值的信息，目前有许多流行的方法可以被应用于此，比如 Spark，AsterixDB 中的 Hyracks[14]层也提供了这样的功能；数据控制指的是在处理数据的时候如何

控制事务执行，如何保证数据的相关特性等，比如 AsterixDB 基于 2PL 去实现事务的并发控制；数据输出，或者数据可视化，也十分重要，比如 Cloudberry[①][15]是一个以 AsterixDB 为基础的，用于支持大量时空数据交互式分析和可视化的研究模型。

基于以上提到的大数据管理系统的必要功能以及 AsterixDB 对 Datar 的若干功能的支持，用 AsterixDB 及其扩展做为 HelpalDB 的核心去构建我们需要的大数据管理系统是十分理想的。在实现 HalpalDB 的过程中，还会根据我们的需求基于 AsterixDB 添加其他功能支持，比如 Spark 提供的复杂的数据分析功能，和 D3 提供的数据可视化功能。

1.3 Datar 的发展前景

在 1970 年代末，“Data Machine”的概念出现了，它描述了一类专门用于存储和分析数据的技术。面对数据量的不断增加，在 1980 年代，“Shared Nothing”数据仓库架构应运而生，它解决了单一计算机的能力逐渐跟不上产业需求的难题。“Shared Nothing”指的是不同的节点分别在它们各自的磁盘和分区上执行它们自己的工作，不共享任何东西，它实现了很好的扩展性[16]。在 20 世纪末 21 世纪初，数据库行业开始广泛认可这种并行数据库的优势。

紧接着，为应对大数据的发展趋势，谷歌在 2003-2006 年作出了杰出贡献：面对数据量的不断增长，它连续发表了三篇很有影响力的文章，专注于发展大数据技术[17]，分别是 2003 年开发出的一套存储系统“The Google File System”（GFS）[18]，2004 年提出的谷歌 MapReduce（GMR）模型[19]，和 2006 年的 BigTable[20]。谷歌开发这三项技术去应对这种网络级别的大数据，当时它们也被视为谷歌技术的三个典型代表。现在大数据行业运用最广泛的 Hadoop 实际上就是谷歌这三种技术的开源实现，其核心思想 MapReduce，是来源于 GMR；Hadoop 文件系统 HDFS，是来源于谷歌文件系统 GFS；HBase 则对应于谷歌的 BigTable。在 2007 年 1 月，数据库软件的先驱 Jim Gray 把这种向大数据的转型称作“The Fourth Paradigm”，即“Data-Intensive Scientific Discovery”[21]。他同时认为，面对这个转型的有效方法是开发一套全新的工具去管理、分析和可视化大量数据。

基于以上背景，该论文将从一个全新的角度描述大数据管理系统，将大数据管理系统的结构与计算机进行类比，从而提出 Datar 这样一个统一的架构描述。其重点在它的构成，它的五个主要组成部分。同时基于提出的这样一个 Datar 架构，还会针对 Helpal 产品实现一个实例，叫做 HelpalDB。该论文的主要贡献可以被总结为以下几点：

(1) 从计算机结构的角度看待大数据管理系统，回顾了当前流行的大数据管理系统，并从数据输入、数据存储、数据计算、数据控制、数据输出五个方面分别进行了科普。
(2) 提出了一个统一的大数据管理系统的架构，命名为 Datar。它为实现智能、动态、可插拔地管理大数据提供了基础和思路。
(3) 在 AsterixDB 及其他扩展和应用的基础上实现了 HelpalDB，它是 Datar 的一个简单实例，是 Helpal 产品的一部分，具有简单的数据输入、数据存储、数据计算、数据控制和数据输出的功能，并简单做了展示。

[①] http://cloudberry.ics.uci.edu/

该论文的结构如下：在第二部分，具体从五个角度探索并总结大数据管理系统，包括数据输入，数据输出，数据控制，数据计算和数据存储；在第三部分，对比当下主流的大数据管理系统解决方案，并与该论文提出的 Datar 架构进行对比；第四部分具体介绍 Datar 的结构，引入一个简单的实例 HelpalDB，并给出了 HelpalDB 的使用方法和操作细节；最后的第五部分是总结和未来展望。

2. 大数据管理系统的五个组成部分

2.1 数据输入

数据输入指的是将数据源的数据进行一定的格式转换，并传输给目标处理端或存储端的过程。数据可以在产生之后就被直接输入进存储系统，也可以经过其他应用或服务的处理之后被插入存储系统。

2.1.1 数据生成

数据的生成有很多种途径，大致归类如下：

(1) **网络数据**：是一种来自于网络的非结构化的数据，它也包括社交网络数据、基于位置的服务数据和链接网址等。

(2) **企业数据**：企业数据主要是由线上交易数据和线上分析数据组成，是大数据的主要来源之一。与网络数据相比，它们一般情况下都是历史静态数据，具有特定的结构。

(3) **政府数据**：从政府机构收集而来的数据。随着数据库模式、文档、邮件、网页内容和 XML 的泛滥，政府机构面临着巨大的一致性问题挑战。

(4) **其他数据**：除以上三类之外的其他来源数据。

另外，自然、商业和社会环境的普遍活动和计算正在产生前所未有的复杂性的异构数据。这些数据集无论在规模上，还是在时间维度或者数据类别上，都有他们独特的特性。

2.1.2 数据传送

数据传送也叫做数据采集，是从联机数据源生成并流向目标文档或应用程序的一个或多个数据流。ETL（Extract Transform Load：提取转换加载）系统的工作原理也是如此。对于 BDMS 而言，想要将数据连续输入大数据管理系统的一个简单的方法是通过一个简单的程序去抓取程序，这个程序可以不断地从外部源抓取数据，解析数据，然后针对每条记录或者每一批次记录出发一条插入语句。如果只是简单地加和使用各种不同的工具完成这样的过程，会很难兼顾到数据的一致性，可扩展性，以及容错性。因此，一个完整的大数据管理系统应当能够支持数据采集的管理功能。

数据采集的一个例子是 Flume[②][22]，它是一个分布式的可靠的服务，用来高效采集、聚集数据，并且能够用来移动大量的日志数据，定制各类数据的发送方。和其他数据采集工具的架构类似，它也有 Source 和 Sink，除此之外它的 Agent 还有 Channel 部分，其中 Source 代表生

[②] https://flume.apache.org/

产者，是数据录入源；Channel 是中间的数据传输通道；Sink 是消费者，是数据接收端。一种常见的大数据分析环境搭建中的数据采集架构是 Hadoop + HBase + Flume + Kafka[23]。其中 Kafka[③]是一个用来构建实时数据管道和流式数据处理应用的服务，他将数据以不同的 Topics 分类存储，将数据从生产者搬运到消费者。在这套数据分析环境架构中，Kafka 扮演中间者的角色，它将 Flume 从各种来源采集来的多种形式的数据统一分发给 HBase 集群。 但由于 Flume 和 Kafka 在兼容上还存在很多问题，这并不是一个易于使用的架构。

在 AsterixDB 中， 集成了一个 Data Feed 的功能，它能够将外部来源的数据连续地注入一个或多个数据集，增量地填充数据集和相关的索引。AsterixDB 之所以提供这样一个功能，是因为大数据的时代带来了快速流动的数据，如果没有 AsterixDB 提供的这样一个集成的功能去加载流动的数据和索引，我们就需要像 Flume + Kafka 一样去结合其他不同的工具，这个过程繁琐复杂，并且由于这些工具往往不是针对某一另外的工具而产生，它们需要不同的接口同时支持不同的外部工具，就经常会出现之前提到的兼容性问题。所以，就像数据库管理系统的出现是为了给以数据为中心的应用提供一套统一的功能集，大数据管理系统也需要提供这样的数据传送功能，同时也要负责其数据管道的管理和容错性。“BAD”系统[24]扩展了 AsterixDB 的 Data Feeds 功能，构建了一个容错性的数据传送机制，通过使用高级语言构建分区进行扩展。这种通用的模型有助于 Datar 去适配于各种数据源和应用。

2.2 数据存储

当下大数据应用的迭代更新加速了数据存储技术的变革，直接推动了数据存储技术的发展。在该论文中，数据存储的重心主要放在原始数据存储和索引存储。除此之外，数据存储还包括还有模式存储、配置文件存储和视图存储，但这些都暂时不在本论文的讨论范围之内。

2.2.1 原始数据存储

针对大数据而言，目前已经存在各种各样的存储系统能够满足大数据存储的不同需求，其中该论文主要关注 NoSQL 数据库技术。如今的企业，不管是哪个行业，都需要数据的存储，而面对当下数据量的增长，数据存储的技术也趋向于向更加灵活、更加兼容的角度发展，这也是 NoSQL 技术产生并替代之前关系型数据库技术的一个原因。NoSQL 数据库提供了现代应用所需要的高效性、可扩展性和灵活性，这点同时也可以说是不同 NoSQL 系统类别之间的唯一共同点。具体来说，NoSQL 数据库可以分为如下四个类别：

(1) **Key-Value DB（键值数据库）**：这是最简单的数据存储类型。它使用的是最基础的关联数组，数据根据它们的键值存储，每一个 key 都是唯一的，并且每一个 key 都有且仅有一组 value 值对应。这样的存储关系也被叫做键值对。与之前传统的关系型数据库相比，键值数据库具有高可扩展性和更短的查询反馈时间。键值数据库的典例包括

[③] https://kafka.apache.org/

亚马逊的 Dynamo[④][25]，LinkedIn 的 Voldmort[⑤][26]，Redis[⑥][27]，Memcache DB[⑦][28]，Scalaris[29]，Riak 和 Tokyo Tyrant。

(2) **Column-oriented DB（列导向数据库）**：根据列而不是行来存储和处理数据，即在磁盘或内存上，表中的每一列都会存储在连续的分区中。为了实现高可扩展性，列导向数据库和行导向数据库都采用节点的结构。对于在少量列上执行聚合操作的分析查询，以这种格式检索数据的速度非常快。主流的列导向数据库包括谷歌的 BigTable[⑧]，Facebook 的 Cassandra[⑨][30]，根据 BigTable 研发出的 Hbase[s][31]，和 HyperTable。

(3) **Document-oriented DB（面向文档的数据库）**：能够解析文档，将它们存储为一种能够用来搜索的、有组织的形式。与键值存储相比，面向文档的存储能够支持更加复杂的数据形式。也是由于文档没有严格的结构模型，在面向文档的数据库中就没必要执行模式迁移。比如开源的 MongoDB[32]，亚马逊的 SimpleDB，和 Apache 的 CouchDB[33]。

(4) **Graph-oriented DB（面向图形的数据库）**：图形数据库是节点和边的集合，它的每一个节点都由一个唯一的标识、一组向外连的边或一组向内连的边、以及键值对组成的集合构成。每一条边都由一个唯一的标识、一个起始节点和一个终端节点、以及一组特征构成。Apache 的 Giraph，Neo4J，和 OrientDB 都是为图形存储而研发的数据库。

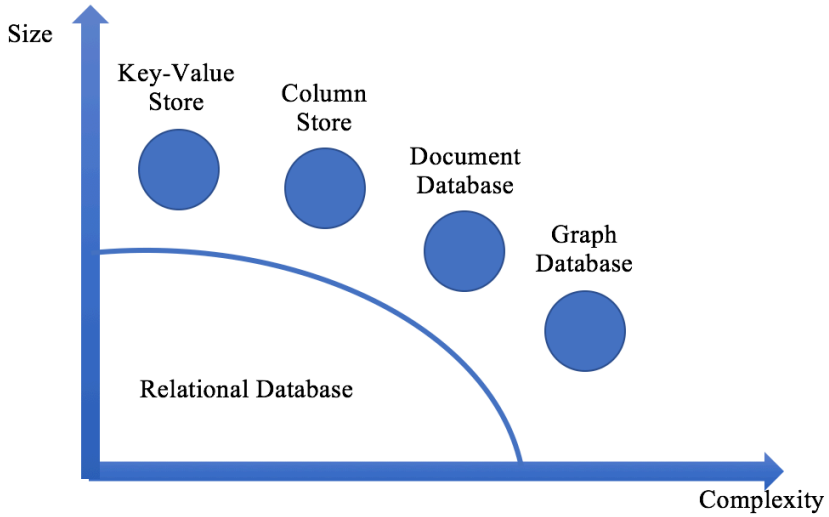

图 3 数据存储模型比较

除了以上提到的数据库之外，其他公司和研发机构也有它们各种各样的数据库模型。比如 Voldemort 提供的可插拔的存储引擎，以及 TiDB 也支持外部数据库的链接。另外，NewSQL[34]也是当下流行的一类关系型数据库管理系统，它追求高可扩展性，支持在线事务处理（OLTP）

[④] https://aws.amazon.com/dynamodb/
[⑤] http://www.project-voldemort.com/
[⑥] http://redis.io/
[⑦] http://memcachedb.org/
[⑧] https://cloud.google.com/bigtable/
[⑨] http://cassandra.apache.org/

的同时保留传统数据库的 ACID，在这种情形下，数据库的存储就像磁盘一样可以被添加、取代或者移除。

综上所述，表 1 是当下流行的大数据存储系统的一个小结，图 3 是不同种类的数据管理系统的规模和复杂程度比较。

表 1 常见的不同类型大数据存储系统

Key-Value	Column	Document	Graph	Other(NewSQL)
Dynamo	BigTable	MongoDB	Giraph	SAP HANA
Voldemort	Cassandra	SinpleDB	Neo4j	TiDB
Redis	HBase	CouchDB	OrientDB	VoltDB
MemcacheDB	HyperTable		Pregel	NuoDB
Scalaris	C-store		FlockDB	Google Spanner
TiKV				

2.2.2 索引存储

无论是在传统的关系型数据库中管理结构型数据，还是在其他技术中管理半结构和非结构型数据，索引向来是减小磁盘读写开销和增加插入、删除、修改、查询语句速度的有效方式。然而索引的缺点也很突出：它需要额外的开销去存储索引文件，这个文件还需要随着数据的更新同步地动态变化。

传统的索引结构包括哈希表，树形索引，多维索引和位图索引。大数据的索引在这些结构的基础上又添加了额外的需求，比如并行化和为并行处理而适配的分区性。针对大数据的索引应运而成了人工智能索引技术。它之所以有“人工智能”这个名字是因为它能够检测大数据中的未知行为，发觉潜在的模式，将对象根据不同的特性进行分类，在数据之间建立联系。潜在语义索引和隐式马尔可夫模型是当下比较流行的两个人工智能索引方式。反观非人工智能的索引方式，索引的形成不依赖于数据或对象的语义，也不依赖于文本之间的联系，而是根据例如某一特定数据集中的最多查询次数之类特性。

2.3 数据计算

数据计算的范畴可以包括从简单的 SQL 式查询到复杂的机器学习技术。目前它们之间没有合适且明确的分界线，以下简单按照两个类型来介绍，分别是数据查询和数据分析。

2.3.1 数据查询

数据查询是通过结构性的查询语句在数据中查找答案的过程，结构性的查询语句即 SQL。流行的模型有 MapReduce，Dryad[35]，All-Pairs，Pregel[36]和 Spark[37]，并且许多内存数据库是专门被设计出来去加速查询计算的。

在这些模型中，MapReduce 是 Apache Hadoop 软件架构的核心组件,它的主要功能可以用两个词来形容，即“mapper”和“reducer”，前者指它能够将工作过滤并分组到集群中的各个节点，后者指它能够将来自每个节点的结果合并归纳为对查询的一致性答案。谷歌的 Pregel 是一个大规模分布式图计算平台，它的特点主要体现在两方面，一方面是它易于编程，它鼓励开发者用节点和边等术语去思考解决问题，而不是从数据流的角度或者在图的某些部分使用转换操作；另一方面是它在解决图表相关的问题上尤其高效。这个框架设计的初衷是比 MapReduce 更有效地支持迭代计算，因为它将数据集保存在内存中，而不是在每次迭代后将其写入磁盘。这个做法用处很大，因为许多图算法也都是迭代执行的。

2.3.2 数据分析

数据分析包括从简单的统计分析到复杂的深度学习挖掘技术，例如基于 Java 的 MLlib[38]，Scipy，Theano，基于 Python 的 Caffe[39]，基于 C++的 TensorFlow[40]，Torch 等。

在这些例子中，MLlib 是 Spark 的机器学习库，它开发的背后动机是让机器学习具有可扩展性和易用性。它包含具有各种机器学习算法的机器学习库，一些较低级别的机器学习原语也包括其中。Caffe 是由伯克利 AI Research（BAIR）和社区贡献者开发[41]的深度学习框架，主要用于图像和视频的处理。TensorFlow 也是一个用于执行复杂数据计算以从头开始构建机器学习模型的库，它自 2005 年 11 月起由 Google Brain 团队开始开源。TensorFlow 使用 Directed Graph 作为其计算模型，这点类似于 Spark。一些其他的高级库，比如 Keras，是使用 TensorFlow 作为其后端的。一个更加出名的应用 TensorFlow 的例子是 AlphaGo，众所周知，它击败了世界职业选手并卫冕冠军。

2.4 数据控制

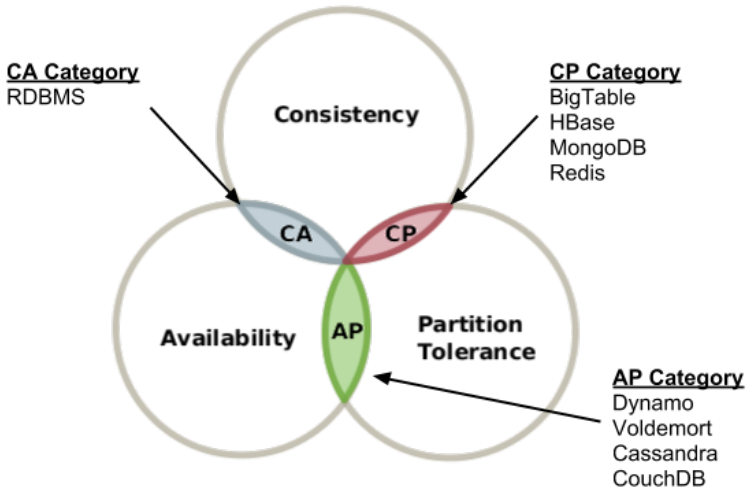

图 4 CAP 理论

大数据管理系统不可能实现传统数据库所要求的 ACID（原子性、一致性、隔离性、持久性），CAP 理论如图 4 所示。 为了应对这个问题，BASE 模型应运而生。BASE 模型不同于 ACID

模型的地方就在于，它牺牲了高一致性来获得高可用性和高可靠性。BASE 思想的应用很广泛，比如谷歌应用引擎使用的 BigTable 和运行在 Hadoop 之上的 HBase 声称在数据中心内具有高度一致性，并且是高度可用的，意味着数据中心之间最终保持一致，更新以异步方式传播到所有副本；亚马逊的 Dynamo，Cassandra 和 Riak 则是牺牲了一致性去支持更好的可用性和分区容忍性。它们都实现了一种较弱的一致性，叫做最终一致性，这种一致性不保证副本的更新顺序和更新时间。前文提到的 NoSQL 也丰富拓展了 BASE 思想[42]。

下面从数据事务、数据恢复和资源管理的角度简单解释数据控制：

2.4.1 数据事务

在分区数据库中，牺牲一致性以获得高可用性的做法同时也能极大提升可扩展性，但我们很难权衡 NoSQL 的速度和规模与传统关系型数据管理系统的事务处理强度和一致性。在现如今的数据库领域，NewSQL[43]是新一代 SQL 数据库产品类别之一，可解决传统针对联机交易处理（OLTP）的关系型数据库管理系统带来的性能问题和可扩展性问题。这样的系统旨在实现 NoSQL 系统的可扩展性，同时保持了传统关系型数据库系统所确保的 ACID 属性。NewSQL 数据库主要为需要高性能和可扩展性的用户处理数据，但它同时也需要确保比 NoSQL 数据库提供的一致性更加高效。尽管各种 NewSQL 数据库在其内部体系结构上有所不同，但它们都利用了关系型数据模型，并且基于 SQL 运行。

NewSQL 有很多著名的实例，其中一个典型的是 Google Spanner[44]，它是在 Google Cloud 上运行的分布式关系数据库服务，旨在支持全球联机事务处理部署、SQL 语义、高可用的横向扩展性以及事务一致性。用户对 Google Cloud Spanner 的关注点集中在云数据库能同时提供的可用性和一致性能力上，这些特征通常被认为是彼此不能兼容的，开发者通常会进行权衡以强调其中的可用性或者一致性。前面提到的 CAP 定理描述的也是这个道理，该定理是向 NoSQL 数据库普遍迁移的基础，这样的迁移有利于获取云系统的可用性和可扩展性。

除此之外，其他各种各样的 NewSQL 型数据库都有它们各自的特点，比如 NuoDB 是面向 SQL 的数据库管理系统，专为在云端进行分布式部署而设计。它可以归类为一种保留传统 SQL 数据库特性的 NewSQL 数据库，同时还包含支持云计算环境中扩展处理的功能。但是，NuoDB 框架的建立和关系型框架是截然不同的，它由三层架构组成，分别是管理层，事务层和存储层。这种分层的构架方法意味着 NuoDB 可以在没有将应用程序及其数据紧密耦合到磁盘驱动的情况下工作，这在一些云应用程序中被证明是一种负担。ClustrixDB 也是一个分布式的横向扩展数据库，适用于高价值，高事务性的，大规模和快速增长的联机交易处理应用。当基于 MySQL 简历的应用阻碍了业务和应用程序的扩展时，ClustrixDB 是一个很好的选择。当业务的扩展需要开发者考虑复制还是分片去扩展读取和写入时，基于 MySQL 构建的服务，无论复制还是分片，都需要重新开发应用程序并重新设计数据架构，这样做非常昂贵，耗时，并且有可能成为一个永无止境的过程。ClustrixDB 通过嵌入式 MySQL 兼容性和线性水平可扩展性解决了这个问题，同时它声称比任何替代方案的 TCO 都要低。VoltDB[10]是一个拥有极快读写速度的内存数据库，它所实现的可扩展性级别在主流的关系型数据库中实现起来可能十分困难或代价高昂。VoltDB 旨在避免可能在关系型数据库中造成处理开销的大部分日志记录和锁操作。尽管 NewSQL 系统

各种实例的内部结构不尽相同，它们都有两个显著的共同特点：它们都支持关系型数据模型，并且使用 SQL 作为它们主要的接口。

2.4.2 数据恢复

大数据应用程序和运行系统都需要强大且快速的恢复过程支持。随着数据库结构为了应对日新月异的应用要求而作出基础性的改变，数据保护的过程也需要重新定义和构造。

目前常见的一些备份恢复机制包括：

(1) 创建多个数据副本，并分布到多个服务器上，这种方法不需要另外的备份和恢复工具。

(2) 从原始数据（raw data）快速轻松地重建丢失的数据。

(3) 周期性地整个备份 PB 数量级的大数据，但这种方法不够经济实用。

(4) 为大数据编写备份恢复脚本，在数据量小并且只有一个大数据平台的情况下，脚本的编写和维护十分简单。

(5) 有时快照也是大数据的有效备份机制，但它只有在数据变化不快时有效。

大数据时代的到来震撼了数据库领域，引入了一类新的“扩展”技术。正确的备份和恢复需要投资，但鉴于大数据在提升商业价值方面发挥的作用，这样的投资是值得的。

2.4.3 资源管理

大数据计算总是会运行在数千台机器上，这就需要集群的资源管理。Mesos[45]是一个开源的集群管理器，它通过动态资源共享和隔离来处理分布式环境中的工作。它将集群中机器/节点的现有资源汇集到一个池中，之后各种工作负载都可以从这个池中获取资源。这种做法也被称为“node abstraction”，它消除了为不同工作负载分配特定机器的需要。Mesos 位于操作系统层和应用程序层之间，基本上可以充当数据中心内核。Mesos 会隔离集群中运行的进程，例如内存、CPU、文件系统和 I/O，以防止它们相互干扰，这种隔离使得 Mesos 能够为工作负载创建单一的大型资源池。Twitter、Airbnb 和 Xogito 等公司都在使用 Apache Mesos。Hadoop YARN[46]是开源 Hadoop 分布式处理框架中的资源管理和作业调度技术，负责将系统资源分配给在 Hadoop 集群中运行的各种应用程序，并调度在不同集群节点上执行的任务。在集群体系结构中，Apache Hadoop YARN 位于 HDFS 和用于运行应用程序的处理引擎之间。它将中央资源管理器与容器，应用程序协调器和节点级代理结合起来，以监视单个集群节点中的处理操作。与 MapReduce 的更多静态分配方法相比，YARN 可以根据需要动态地为应用程序分配资源，这是一种旨在提高资源利用率和应用程序性能的功能。Apache ZooKeeper[47]则相当于一个具有最终一致性的复制同步服务器，它具有很好的鲁棒性，因为持久数据是分布在多个节点上的，并且客户端可以连接到它们中的任何一个，如果一个节点失败了则进行迁移。具体来说，主节点是通过集合中的节点一致动态选择出来的。

2.5 数据输出

展示大数据管理结果的最好方法就是数据可视化。另外在这个部分，也会提到数据分享。

2.5.1 数据可视化

大数据分析通过减少大数据应用程序中的数据规模和复杂性而起着关键性的作用，可视化是帮助大数据分析获得更完整的数据视图和发现数具价值的重要途径。可视化提供了一种交互式和图形式的方法来帮助我们更深入地理解大数据。目前常见的工具有 Tableau，Plotly，Visual.ly 等。另外，Zeppelin 也是一款基于网络的笔记本，它为用户提供了一中简单，直接的方法来在网络笔记本中执行任意代码，用户可以执行 Scala，SQL，甚至可以安排一个作业去定期运行。交互式的可视化能够更直观地展现数据，它可以通过很多方法实现，比如缩放等。可扩展性和动态性是视觉分析中的两个主要挑战[47]。

2.5.2 数据分享

由于共享数据能够通过为科学发现提供更好的机会来增加主体参与对社会的潜在益处，Brakewood 和 Poldrack 认为研究人员有道德义务去分享他们的数据，除非这样做会增加主体的风险[49]。在研究领域，许多机构都通过分享他们的数据来促进研究的进展。各国政府也开始向公众分享数据以获得共同利益。但是，由于隐私和机密，个人数据和企业内部数据等数据无法实现共享。因此，我们应该制定一些指导方针，引导我们正确地共享数据，而不是对所有数据一视同仁。

3. 大数据管理系统的主流解决方案及对比

3.1 主流的大数据管理系统解决方案

3.1.1 AsterixDB

AsterixDB[11]是由 UCI，UCR，Google 等合作开发的，以数据存储为核心的大数据管理系统（BDMS），具体结构如图 5 所示。它丰富的功能集是将他与其他大数据平台区分开来的主要特点。这个特点也使得 AsterixDB 非常适合现代工业需求，比如网络数据仓库和社交数据存储与分析。其具体特点有：

(1) 数据模型：AsterixDB 的数据模型称作 ADM（The Asterix Data Model），是由扩展 JSON 和对象数据库相结合的想法产生的，半结构化的 NoSQL 风格的数据模型。
(2) 查询语言：两种表达式和声明式查询语言（SQL++和 AQL），支持对半结构化数据进行广泛的查询和分析。
(3) 可扩展性：Apache Hyracks 是一个并行运行查询执行引擎，已经在多达 1000 多个核和 500 多个磁盘上进行了规模测试。
(4) 本地存储：分区的基于 LSM 的数据存储和索引，支持对半结构化数据的有效管理。
(5) 外置存储：支持查询访问外部存储的数据（例如 HDFS 中的数据）。
(6) 数据类型：支持一组丰富的原始数据类型，除了整数，浮点数和文本数据意外，还包括空间和时间数据。
(7) 索引：二级索引选项，包括 B+树，R 树和倒排关键字索引类型。

(8) 事务：基本的事务（并发和恢复）功能，类似于 NoSQL 存储实现的事务功能。

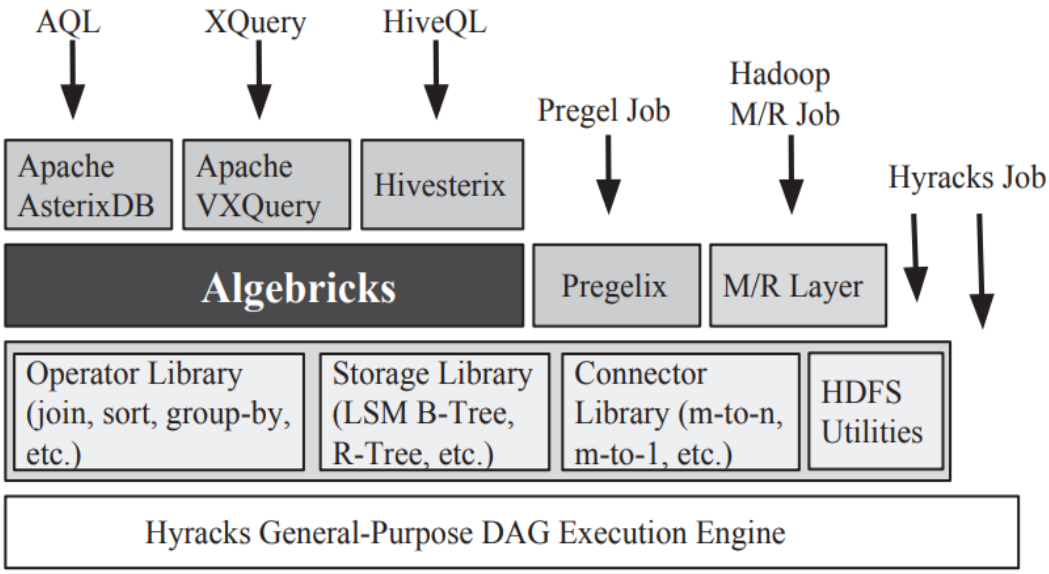

图 5 AsterixDb 结构

3.1.2 BDAS

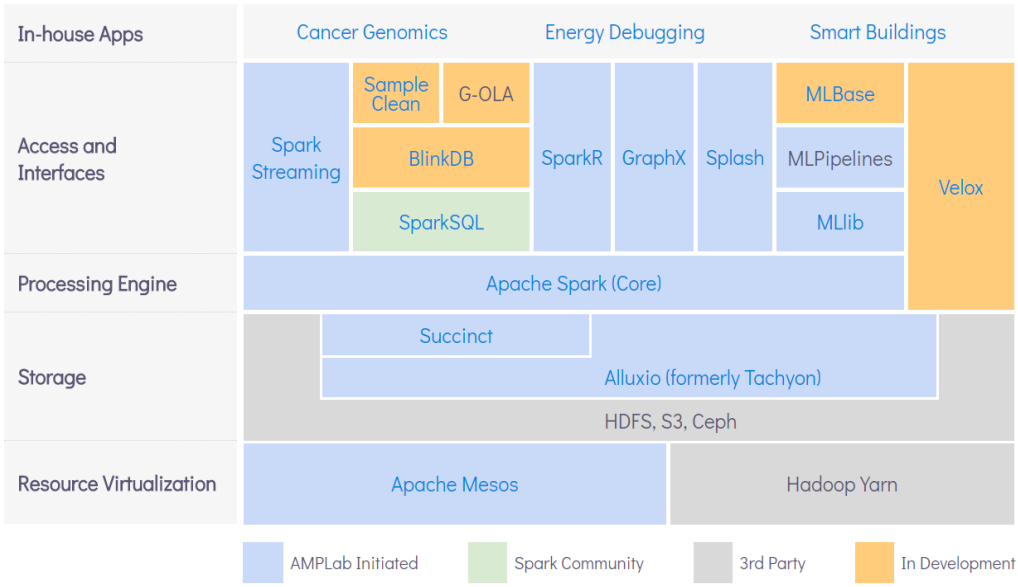

图 6 BDAS 结构

Berkeley Data Analytics Stack[50]是一个开源软件栈，它集成了 AMPLab 为大数据儿构建的各种软件组件，具体见图 6。顶层的应用根据底层的架构进行开发，具体底层 BDAS 的架构可以分为三层：资源管理层，数据管理层，数据处理层。基于 BDAS 构建的应用程序有很多，比如 Yahoo 的广告定制及投放主要基于 Spark 和 Shark，Ooyala 的视频联机分析处理平台等。

3.1.3 小米数据平台

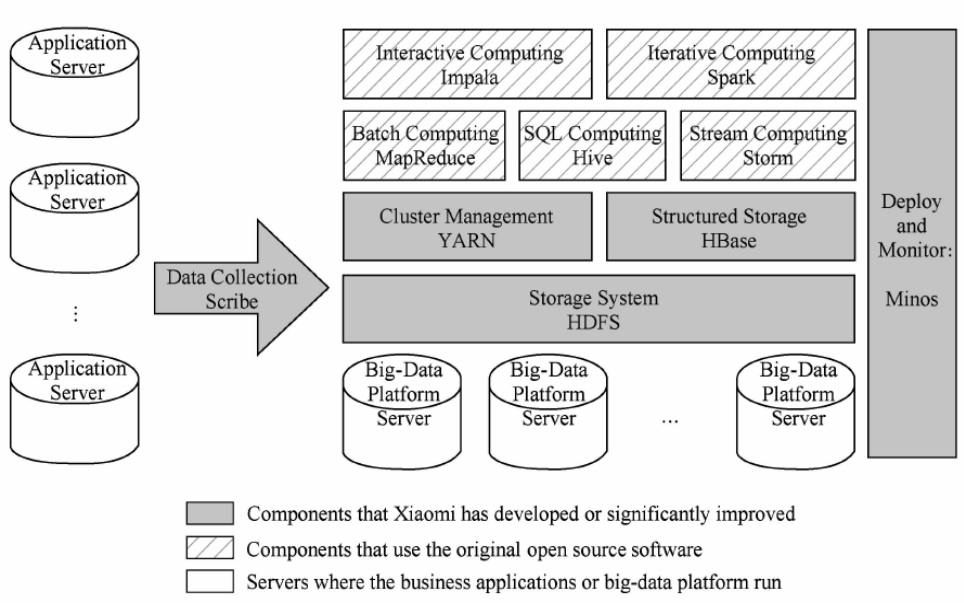

图 7 小米数据平台结构

图 7 是由小米创始人雷军等人于 2017 年提出的基于开源生态系统的大数据收集与处理平台[51]，它主要也依赖于一些已经开源且相对成熟的组件，比如用于资源管理的 Zookeeper，HBase，Hadoop 的 HDFS、MapReduce、YARN，Hive 等。除此之外，小米还开发了一个用于监控的系统，叫做 Minos。

3.1.4 TiDB

TiDB[52]是由 PingCAP 公司基于 Google Spanner / F1 论文实现的开源分布式 NewSQL 数据库，它是由若干模块组成的分布式系统。TiDB 的存储层依赖于 TiKV[53]，TiKV 采用 Key-Value 的方式存储数据。TiDB 的计算层即 SQL 层，也就是 TiDB Server[54]这一层，主要用来处理用户请求，执行 SQL 运算。

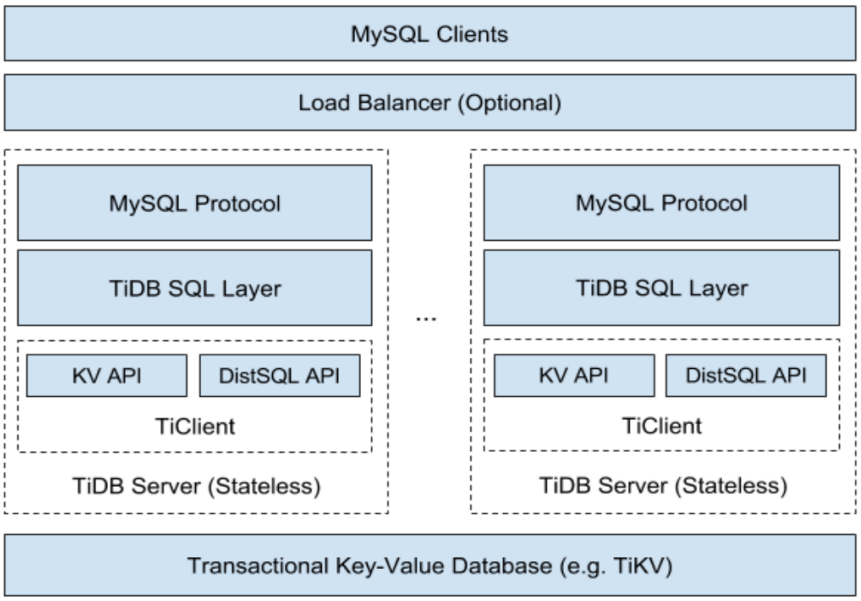

图 8 TiDB 结构

3.1.5 Apache Beam

Apache Beam 由谷歌开发，本身并不是一个处理平台，而是提供了统一的编程模型，相当于在现有的框架上封装了一层 API，目的是通过统一的 Beam 编程模式来简化底层框架的配置操作。

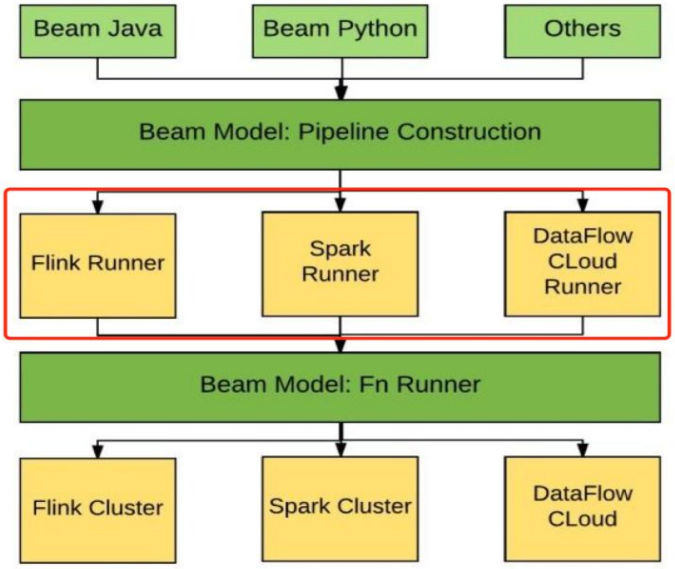

图 9 Apache Beam 运行流程

3.2 各方案与 Datar 的对比

以上大数据解决方案中，Datar 的思想与 Apache Beam 最为相似，目的也是开发一套封装的接口，用来统一地管理各部分插件，简化大数据平台的开发流程。但它与 Apache Beam 的不同之处主要在于，Datar 的思想来自于计算机的结构，有明确区分的五部分组成（即输入，存储，计算，管理，输出），而 Apache Beam 的核心在中间层 Runner，仅仅是一系列主流框架的适配器。设计 Datar 的初衷是为了在五个部分分别实现灵活可扩展的管理，因此与 Apache Beam 相比，Datar 还有更大的发展空间。Datar 的主要精力并不是集中在新引擎的开发，而是框架结构的设计思想。

将 Datar 设计为五部分分开的架构，尤其是将计算、存储、控制三部分的分离，符合未来 BDMS 的发展趋势。随着科技的发展，宽带的速度获得了极大的提升，但是磁盘在速度上基本没有太大变化，这种硬件的改变推动了软件架构的迭代更新。另外，计算资源的变化速度总是快于存储资源，两者之间产生的木桶效应往往会造成集群资源的浪费，实现存储与计算的分离，会对大数据方案的开发提供了极大的便利。正因如此，存储与计算的分离在近几年已初见成效，比如阿里云开发的 E-MapReduceError! Reference source not found.，采用 OSS 作为云存储，Hadoop、Spark 可以作为计算引擎直接分析 OSS 中存储的数据，节约成本的同时提升了扩展性，具有很好的发展前景。之前提到的 TiDB 也实现了计算和存储的分离，然而对于控制的分离，还不是十分彻底。

Datar 实例 HelpalDB 与前四个大数据解决方案在系统耦合性和系统原创性上的比较，参见图 10。HelpalDB 在设计中间接口层时，明确按照五部分的功能区分开实现，这是 HelpalDB 与其他大数据解决方案的主要不同点。

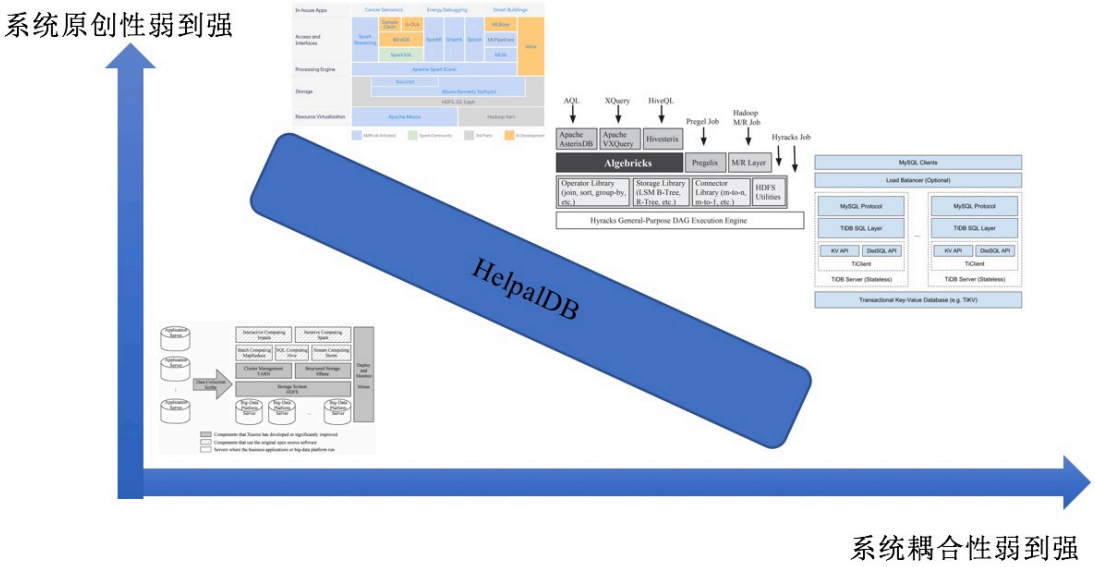

图 10 HelpalDB 的定位

4. 实现 Datar 实例 —— HelpalDB

4.1 Datar 的前提与假设

正如之前提到的一样，该论文设想一个通用的大数据管理系统架构，即 Datar，去帮助 HelPal 更好地管理数据。这个架构的想法本质上来自于计算机的结构，即计算机的输入，存储，控制，计算和输出。Datar 是一系列集成的软件/系统，它期望于通过可插拔的特定的功能去自动而智能地管理大数据。可插拔性意味着 Datar 的五个部分中任何一个都可以很容易地被替换，自动化意味着从数据输入到数据输出的流程可以自动一致地执行，智能意味着 Datar 的最终目标是深入挖掘有价值的信息。

4.2 Datar 的架构和它的实例 HelpalDB

为了实现预想的 Datar 模型，该论文用代码实现了一个叫做 HelpalDB 的实例。HelpalDB 主要基于 AsterixDB 和其他几个工具来实现数据输入、数据存储、数据计算、数据控制和数据输出功能。目前，HelpalDB 将主要依赖于 AsterixDB， Spark-MLlib 和 D3。AsterixDB 是数据输入，数据存储，数据控制和数据查询的核心组件，Spark-MLlib 用于数据分析，D3 用于数据输出。

就数据输入而言，AsterixDB 本身提供了一个 Data Feed 功能，Feed 适配器可以与数据源建立连接并且接收，解析和将数据转换成能够在 AsterixDB 中存储的对象。一个 Feed 适配器相当于一个接口的实现，其详细信息由特定数据源决定。用户可以选择给适配器提供参数来配置其运行时的某些行为。根据数据源提供的数据传输协议/API，Feed 适配器可以按照 Push 的模式运行，也可以按照 Pull 的模式。Push 模式只包含 Feed 适配器向数据源发起的一个初始请求，这个请求用于设置连接。一旦连接被授权，数据源将“推送”数据到 Feed 适配器，而不需要适配器的任何后续请求。相反，当以 Pull 的模式操作时，Feed 适配器每次都会发出一个

单独的请求来接收数据。AsterixDB 目前为几种常用数据源提供了内置适配器，比如 Twitter 和 RSS 源。AsterixDB 还提供了一个通用的基于 Socket 的适配器，可用于提取针对制定 Socket 的数据，Socket 适配器也是 HelpalDB 所采用的 Feed 适配器。关于存储，需要先了解 AsterixDB 中最顶层的概念，即 dataverse，它是“data universe”的缩写，是创建和管理数据类型、数据集、各种功能和其他应用工作的地方。当用户第一次开始使用 AsterixDB 实例时，它从“空”开始，不包含 AsterixDB 系统目录以外的其他内容。要将数据存储在 AsterixDB 中，用户需要首先创建一个 dataverse，然后将它作为管理数据类型和数据集的地方。关于数据控制，它的目的是使得所有数据管理操作都能无冲突地成功执行。在 AsterixDB 中，它是一个内置的模块，不可以更改。AsterixDB 的查询语言是 AQL 查询语言。AQL 主要基于 XQuery，该语言是由 W3C 于 2000 年初期至中期开发和标准化的，用于查询以 XML 格式存储的半结构化数据。AQL 不是 SQL，也不是 SQL，换句话说，AsterixDB 完全是“NoSQL 兼容”的。

Spark 的 MLbase 致力于简化可扩展的机器学习管道的开发和部署流程，MLlib 是其中核心的机器学习库。它最早由 MLbase 组在 AMPLab 开发，所用语言是 Scala 和 Java，后来又加入了 Python。作为 Spark 的一部分，它给完整的数据分析工作流带来了便利，同时又有很好的性能。

D3 是由 Mike Bostock 开发的一个开源 JavaScript 数据库，专注于数据和构建数据可视化项目，用于在 Web 浏览器中使用 SVG，HTML 和 CSS 创建自定义交互式数据可视化项目。随着如今大量数据的产生，传递这些信息也变得越来越困难。数据的可视表示是传达有意义信息的最有效手段，D3 为创建这些数据可视化提供了极大的方便性和灵活性。它是动态的，直观的，并且是易于实现的。

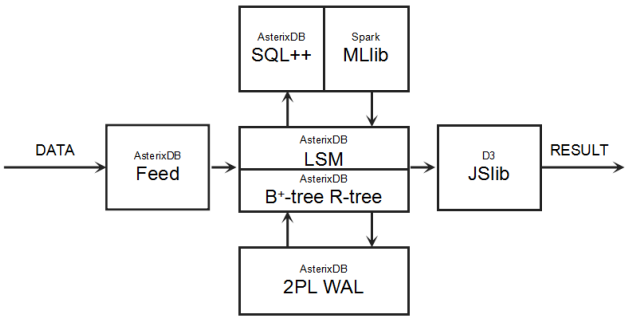

图 11 HelpalDB 框架结构

图 11 展示了 HelpalDB 的实现结构。如图中所示，AsterixDB 提供了基于 LSM 的数据存储，具有 B+树和 R 树的索引结构，2PL 并发控制和预写日志（WAL）恢复策略。对于数据计算，AsterixDB 使用 AQL 数据查询和 Spark-MLlib 进行数据分析。数据输入是基于 AsterixDB

的 Feed 机制。数据输出是基于 D3 的 JSlib。如果能够实现智能化，该系统的理想状态是能够支持更多的流行系统（比如 TensorFlow）作为其插件。

4.3 HelpalDB 使用实例

4.3.1 HelpalDB 的安装和管理

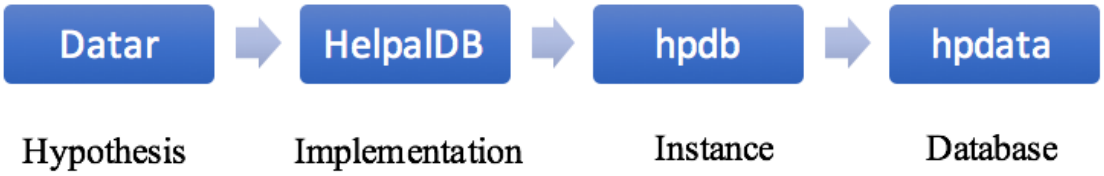

图 12 几个关键概念之间的联系

为了更好地理解各个部分之间的关系，图 12 给出了图形化的解释，即 hpdb 是 HelpalDB 的一个实例，hpdata 是存储数据的物理文件。图 13 展示了 HelpalDB 的配置。在安装和使用 HelpalDB 之前，我们有必要首先配置好五个模块的配置信息。对每个模块，应当都有一个配置器使得该模块能够配置到 HelpalDB 当中，这个配置器应当是共享给用户去使用的。这也是开源 HelpalDB 的原因，而且所有按照 HelpalDB 的设计实现的模块/系统/库都可以嵌入到 HelpalDB 中使用。图 14 展示了如何安装和管理 HelpalDB，其中 hpdb 是 HelpalDB 的一个实例。管理 HelpalDB 包括创建、开始、使用、停止、删除和详细信息。

config.ini 〉 No Selection

```
1  [INPUT]
2  module = AsterixDB
3  path = /AsterixDB/bin/
4
5  [STORAGE]
6  module = AsterixDB
7  path = /AsterixDB/bin/
8
9  [COMPUTATION]
10 module = Spark
11 path = /Spark/bin/
12
13 [CONTROL]
14 module = AsterixDB
15 path = /AsterixDB/bin/
16
17 [OUTPUT]
18 module = D3
19 path = /D3/bin/
```

图 13 HelpalDB 的配置文件

- install HelpalDB project

```
./hpdb.sh install HelpalDB
```

- manage [create, start, use, stop, delete, describe] HelpalDB instance named hpdb

```
./hpdb.sh [new, start, use, stop, delete, info] HelpalDB hpdb
```

图 14 安装和管理 HelpalDB

4.3.2 HelpalDB 实例 hpdb 的操作

图 15 展示了 hpdb 的操作，包括输入、存储、计算、控制和输出。

- input [feed, file] path_from path_to

```
*TBD module*
./hpdb.sh use HelpalDB hpdb
using HelpalDB hpdb
HelpalDB>>> input [-feed, -file] data/source.xls data/destination.table
input...
*some excuting info here*
inputed.
HelpalDB>>> quit
use HelpalDB hpdb end.
```

（a） 数据输入

- store [create, delete] datastore

```
./hpdb.sh use HelpalDB hpdb
using HelpalDB hpdb
HelpalDB>>> store [-new, -delete] hpdata
store...
*some excuting info here*
stored.
HelpalDB>>> quit
use HelpalDB hpdb end.
```

（b）数据存储

- compute [query, analysis] code_query

```
./hpdb.sh use HelpalDB hpdb
using HelpalDB hpdb
HelpalDB>>> compute [-query, -analysis] use dataverse hpdata; for $ds in dataset Metadata.Dataset
*HelpalDB>>> compute -query use dataverse hpdata; create type UserType as {userName: string, user
*HelpalDB>>> compute -query use dataverse hpdata; create dataset Users(UserType) primary key user
*HelpalDB>>> compute -query use dataverse hpdata; insert into dataset Users({"userName":"Tom","us
*HelpalDB>>> compute -query use dataverse hpdata; for $user in dataset Users return $user;*
*HelpalDB>>> compute -analysis sc.parallelize(1 to 1000).count()*
compute...
*some excuting info here*
computed.
HelpalDB>>> quit
use HelpalDB hpdb end.
```

（c）数据计算

- control

```
*BUILDIN module*
./hpdb.sh use HelpalDB hpdb
using HelpalDB hpdb
HelpalDB>>> control
control...
Built-in Module. NO Configuration!
controlled.
HelpalDB>>> quit
use HelpalDB hpdb end.
```

（d）数据控制

- output [visual, share] path_data

```
./hpdb.sh use HelpalDB hpdb
using HelpalDB hpdb
HelpalDB>>> output -visual
output...
*some excuting info here*
outputed.
HelpalDB>>> quit
use HelpalDB hpdb end.
```

（e）数据输出

图 15 HelpalDB 实例 hpdb 的操作示例

(1) **hpdb 输入：**数据输入是将从外部来源的连续数据输入到 HelpalDB，并逐渐填充数据集和相关索引。外部数据源可以是文件，另一个数据库中的数据，或 Feed 功能馈送的流式数据。

(2) **hpdb 存储：**数据存储是将数据存储到物理存储设备上。在这个流程中可以建立一个新的数据库。

(3) **hpdb 计算：**数据计算通过简单的查询语句，数据挖掘或者深度学习，从永久数据中探寻有价值的信息。基础的数据操作语句包括插入、删除、更新等。复杂的数据分析将由 Spark-MLlib 提供。

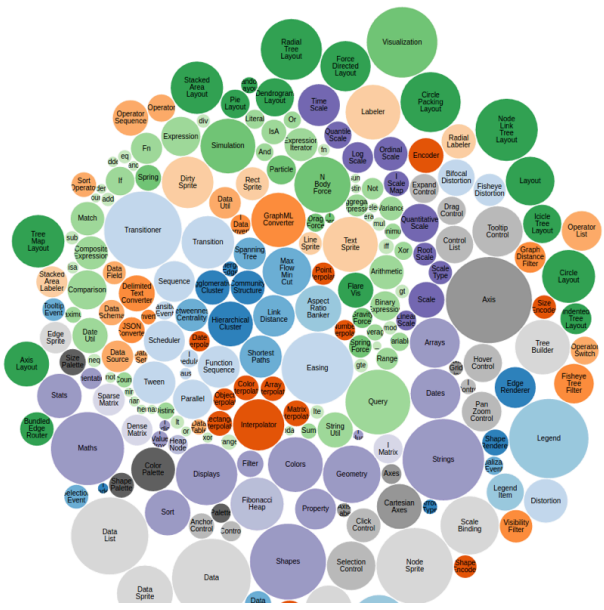

图 16 hpdb 中 D3 实现的可视化效果

(4) **hpdb 控制**：数据控制是协调所有的数据管理操作，使得其中不存在冲突，差错或失败操作。目前，它是由 AsterixDB 中的一个内置模块实现的，因此也不可以更改。其他 Datar 的实现可以去适配其他数据控制的工具。对 HelpalDB 而言，目前倾向于相对依赖 AsterixDB。

(5) **hpdb 输出**：数据输出提供了一种展示结果的方式。在 HelpalDB 中，该论文应用 D3 作为可视化工具，实例见图 16。

5. 结论和未来展望

该论文由计算机的结构出发，阐述了一个预想的大数据管理系统结构 Datar 的五个组成部分，并给出了 Datar 的实现项目 HelpalDB，它实现了可插拔地、自动且智能地运用特定功能管理大数据。

如今，我们已经进入了大数据的纪元。通过更好地分析大量可用的数据，我们可以在许多科学领域加快进步，并提高许多企业的盈利能力和成功率。但是，在实现这个目标之前，我们必须首先解决许多技术难题。更重要的是，解决这些难题需要的将是变革性的解决方案，因为这些难题不会由下一代工业产品的发展而自然解决。如果我们要实现大数据带来的高利润承诺，我们必须支持并鼓励基础研究的进行，以更快地解决这些架构和技术上的挑战。

参考文献

[1] Turing, A. M. "Computing machinery and intelligence." *Parsing the Turing Test*. Springer Netherlands, 2009:44-53.

[2] Neumann, J. Von. "First draft of a report on the EDVAC." *IEEE Annals of the History of Computing*, 15.4(2002):27-75.

[3] Allan G. Bromley. "Charles Babbage's Analytical Engine, 1838." *IEEE Annals of the History of Computing* 20.3(1998):29-45.

[4] Bachman, Charles W. "On a generalized language for file organization and manipulation". *ACM,* 1966.

[5] Codd, E. F. "A relational model of data for large shared data banks." *Readings in Database Systems* Morgan Kaufmann Publishers Inc. 1994:377-387.

[6] M. Stonebraker, Michael, et al. "The design and implementation of INGRES." *The INGRES papers: anatomy of a relational database system Addison-Wesley Longman Publishing Co. Inc.* 1986:5-45.

[7] M. Stonebraker, "The postgres DBMS," in Proceedings of the *1990 ACM SIGMOD International Conference on Management of Data*, Atlantic City, 1990:394-395.

[8] M. Stonebraker, Michael, et al. "Mariposa: a wide-area distributed database system." *Vldb Journal* 5.1(1996):48-63.

[9] M. Stonebraker, Michael, et al. "C-Store: A Column-oriented DBMS. " *Eeding, VLDB* 2005:6741--6747.

[10] M. Stonebraker, A. Weisberg, "The voltdb main memory DBMS," *IEEE Data Eng. Bull.*, 2013:21－27.

[11] Alsubaiee, Sattam, et al. "AsterixDB: a scalable, open source BDMS." Proceedings of the *Vldb Endowment* 7.14(2014):1905-1916.

[12] R. Grover and M. J. Carey, "Data ingestion in asterixdb." in *EDBT*, 2015:605－616.

[13] Alsubaiee, Sattam, et al. "Storage management in AsterixDB." Proceedings of the *Vldb Endowment* 7.10(2014):841-852.

[14] Borkar, Vinayak, et al. "Hyracks: A flexible and extensible foundation for data-intensive computing." *IEEE, International Conference on Data Engineering* IEEE, 2011:1151-1162.

[15] Clifton, Harry. "Cloudberry." *Irish Pages* 4.1(2007):23-23.

[16] 杨思意. Shared Nothing v.s. Shared Disk - 为程序员服务 [EB/OL]. http://ju.outofmemory.cn/entry/75412.2018 年 3 月 22 日访问.

[17] 陈晨, 陈达丽. "谷歌大数据技术的研究及开源实现." *软件产业与工程* 5(2015):31-36.

[18] Fumin. 谷歌技术“三宝”之谷歌文件系统 [EB/OL]. https://blog.csdn.net/opennaive/article/details/7483523.2018 年 3 月 25 日访问.

[19] Fumin. 谷歌技术“三宝”之 MapReduce [EB/OL]. https://blog.csdn.net/opennaive/article/details/7514146.2018 年 3 月 25 日访问.

[20] Fumin. 谷歌技术“三宝”之 BigTable [EB/OL]. https://blog.csdn.net/opennaive/article/details/7532589.2018 年 3 月 25 日访问.

[21] Hey, Tony. "The Fourth Paradigm - Data-Intensive Scientific Discovery." *International Symposium on Information Management in a Changing World* Springer, Berlin, Heidelberg, 2012:1-1.

[22] Hari Shreedharan，马延辉，石东杰. *"Flume:构建高可用可扩展的海量日志采集系统."* 电子工业出版社, 2015.

[23] Qiaqia. Flume 各种坑 [EB/OL]. https://blog.csdn.net/qiaqia609/article/details/78973334.2018 年 3 月 31 日访问.

[24] Qader, Mohiuddin Abdul. "A BAD Demonstration: Towards Big Active Data." *VLDB* 2017.

[25] Decandia, Giuseppe, et al. "Dynamo: amazon's highly available key-value store." *ACM Sigops Symposium on Operating Systems Principles* ACM, 2007:205-220.

[26] Sumbaly, Roshan, et al. "Serving large-scale batch computed data with project Voldemort." *Usenix Conference on File and Storage Technologies* USENIX Association, 2012:18-18.

[27] Carlson, Josiah L. "Redis in Action." *Media.johnwiley.com.au* 2013.

[28] Fitzpatrick, Brad. "Distributed caching with memcached." *Linux Journal* 2004.124(2004): 72-76.

[29] Thorsten Schütt, F. Schintke, and A. Reinefeld. "Scalaris: reliable transactional p2p key/value store." *ACM Sigplan Workshop on Erlang* ACM, 2008:41-48.

[30] Lakshman, Avinash, and P. Malik. "Cassandra: a decentralized structured storage system." *Acm Sigops Operating Systems Review* 44.2(2010):35-40.

[31] Vora, Mehul Nalin. "Hadoop-HBase for large-scale data." *2011 international conference on computer science and network technology* 2011:601-605.

[32] 王光磊. "MongoDB 数据库的应用研究和方案优化." *中国科技信息* 20(2011):93-94.

[33] 刘臣. "非关系数据库 CouchDB 的应用." *电脑知识与技术* 14(2013):3220-3222.

[34] Bonelee. NewSQL - 优化的SQL存储引擎 [EB/OL]. https://www.cnblogs.com/bonelee/p/6265291.html.2018年4月1日访问.

[35] Isard, Michael, et al. "Dryad: distributed data-parallel programs from sequential building blocks." *Acm Sigops Operating Systems Review* 41.3(2007):59-72.

[36] Malewicz, Grzegorz, et al. "Pregel: a system for large-scale graph processing." *ACM SIGMOD International Conference on Management of Data* ACM, 2010:135-146.

[37] Zaharia, Matei, et al. "Spark: cluster computing with working sets." *Usenix Conference on Hot Topics in Cloud Computing* USENIX Association, 2010:10-10.

[38] Meng, Xiangrui, et al. "MLlib: machine learning in apache spark." *Journal of Machine Learning Research* 17.1(2015):1235-1241.

[39] Jia, Yangqing, et al. "Caffe: Convolutional Architecture for Fast Feature Embedding." *Acm International Conference on Multimedia* ACM, 2014:675-678.

[40] Abadi, Martín. "TensorFlow: learning functions at scale." *Acm Sigplan Notices* 51.9(2016):1-1.

[41] 百度百科. Caffe（卷积神经网络框架）[EB/OL]. https://baike.baidu.com/link?url=n2F1_dypgCmwqpRNCwnSecQlxm1RN25PKR-gNBjH5_mlznZoHn22npbHj2FPWTxLIwmjJagVGC7uibskOCWa0q.2018年4月3日访问.

[42] Summer_ZJU. CAP理论和BASE思想. https://blog.csdn.net/xiaqunfeng123/article/details/51688974.2018年4月4日访问.

[43] KeepLearningBigData. NewSQL. https://blog.csdn.net/xubo245/article/details/47604583.2018年4月6日访问.

[44] Corbett, James C, et al. "Spanner: Google's globally-distributed database." *Usenix Conference on Operating Systems Design and Implementation* 2012:251-264.

[45] Hindman, Benjamin, et al. "Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center." *Proceedings of the 8th USENIX conference on Networked systems design and implementation* USENIX Association, 2011:429-483.

[46] Vavilapalli, Vinod Kumar, et al. "Apache Hadoop YARN: yet another resource negotiator." *Symposium on Cloud Computing* ACM, 2013:1-16.

[47] 百度百科. Zookeeper [EB/OL]. https://www.baidu.com/link?url=TCj0qlkeWOR5pXVE8rRbgud_E06CShBfqHGugrejwq9i7sJodYtJ781YLRaZ33eezrVj_Q7s49_NVOkM4vnQNFU5MKRKAiR-xmuODfjIbxi&wd=&eqid=878d009700090455000000035ad1f438.2018年4月8日访问.

[48] Wang, Lidong. "Big Data and Visualization: Methods, Challenges and Technology Progress." *Canadian Journal of Electrical & Computer Engineering* 34.3(2015):3-6.

[49] Brakewood, Beth, and R. A. Poldrack. "The ethics of secondary data analysis: Considering the application of Belmont principles to the sharing of neuroimaging data." *Neuroimage* 82.1(2013):671-676.

[50] 阿涅斯瓦兰吴京润, and 黄经业. "颠覆大数据分析 : 基于 Storm、Spark 等 Hadoop 替代技术的实时应用." *电子工业出版社*, 2015.

[51] 雷军等. "基于开源生态系统的大数据平台研究." *计算机研究与发展* 54.1(2017):80-93.

[52] 知乎. 三篇文章了解 TiDB 技术内幕——说调度 [EB/OL]. https://zhuanlan.zhihu.com/p/27275483.2018 年 4 月 20 日访问.

[53] 知乎. 三篇文章了解 TiDB 技术内幕——说存储 [EB/OL]. https://zhuanlan.zhihu.com/p/26967545.2018 年 4 月 20 日访问.

[54] 知乎. 三篇文章了解 TiDB 技术内幕——说计算 [EB/OL]. https://zhuanlan.zhihu.com/p/27108657.2018 年 4 月 20 日访问.

[55] 阿里云 E-MapReduce. E-MapReduce 支持计算与存储分离,成本下降 1 倍 [EB/OL]. https://yq.aliyun.com/articles/59136.2018 年 4 月 21 日访问.