第1卷 第1期 2019年10月
Vol.1 No.1 Oct.2019

ISSN 2096-742X
CN 10-1649/TP

文献DOI：
10.11871/jfdc.issn.
2096-742X.2019.
01.004

文献PID：
21.86101.2/jfdc.
2096-742X.2019.
01.004

页码：22-34

开放科学标识码
（OSID）

大数据基础理论与系统关键技术浅析

华强胜[1,2,3,4]，郑志高[1,2,3,4]，胡振宇[1,2,3,4]，钟芷漫[1,2,3,4]，林昌富[1,2,3,4]，
赵峰[1,2,3,4]，金海[1,2,3,4*]，石宣化[1,2,3,4]

1.大数据技术与系统国家地方联合工程研究中心，湖北 武汉 430074
2.服务计算技术与系统教育部重点实验室，湖北 武汉 430074
3.集群与网格计算湖北省重点实验室，湖北 武汉 430074
4.华中科技大学计算机科学与技术学院，湖北 武汉 430074

摘　要：【目的】本文主要就大数据基础理论及系统相关研究背景、技术架构和关键技术展开介绍，并结合技术发展趋势提出未来研究和技术发展方向。【方法】本文在简要介绍大数据处理基础理论的基础上，从面向数据并行的大数据处理技术、RDF (Resource Description Framework)图数据的查询与匹配、大数据分析技术三个方面简要介绍了大数据系统的关键技术。【结果】未来数据产生的速度将进一步提高，在这种应用背景下，如何在设备端进行快速的数据处理成为一种趋势。【结论】未来，我们将在继续关注大数据基础理论与系统关键技术的基础上，引入边缘计算、雾计算等场景，研究物联网环境下的大数据处理。

关键词：低复杂度算法；数据并行；QoS机制技术；图数据处理；语言模型

A Brief Review of Theory and System Technologies for Big Data

Hua Qiangsheng[1,2,3,4], Zheng Zhigao[1,2,3,4], Hu Zhenyu[1,2,3,4], Zhong Zhiman[1,2,3,4], Lin Changfu[1,2,3,4], Zhao Feng[1,2,3,4], Jin Hai[1,2,3,4*], Shi Xuanhua[1,2,3,4]

1.National Engineering Research Center for Big Data Technology and System, Wuhan, Hubei 430074, China
2.Services Computing Technology and System Lab, Wuhan, Hubei 430074, China
3.Cluster and Grid Computing Lab, Wuhan, Hubei 430074, China
4.School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China

Abstract: [**Objective**] The article mainly gives a brief review for big data theory and systems, including the research background, the technical architecture and the key technologies following by estimating future research directions. **[Method]** On the basis of the brief introduction of the big data processing theory, this paper introduces the key technologies for big data systems by the three aspects: the data parallel processing methods, the Resource Description Framework (RDF) graph data query and matching, and the big data analysis technologies. **[Results]** The speed of data generation will be accelerated further more in near future, thus how to quickly process the data on the edge side would lead a research trend. **[Conclusion]** In future, we will continue to focus on the basic theory and system technologies of big data. At the same time, we will also try to introduce some new research directions, such as edge computing, fog computing, and big data processing in the Internet of Things.

Keywords: low complexity algorithms; data parallelism technologies; QoS mechanism; graph data processing; language model

基金项目：国家重点研发计划云计算与大数据重点专项 (2018YFB1003203)；国家自然科学基金面上项目（61572216）
* 通讯作者（E-mail: hjin@hust.edu.cn）

引言

大数据具有多样性、快速性和大规模等特点，传统算法与系统面对大数据的复杂应用场景具有很大挑战性。大数据基础理论以及大数据的存储、检索、分析和挖掘等技术，为大数据的广泛高效应用提供技术支持。本文基于“大数据快速存储与计算环境”、“高通量计算与近似计算的理论与算法”等国家重点研发计划课题；“面向大数据的高时效并行计算机系统结构与技术”、“大规模图中图性质求解的低复杂度分布式算法研究”等国家自然科学基金项目以及一批面向电子政务、航空运输等应用处理平台项目，对不同领域大数据问题进行简要介绍，并从算法、系统等多个维度介绍一系列的典型系统、探讨多个典型大数据领域数据处理问题解决思路。

（1）在大数据理论方面，针对隐私保护数据聚合问题，大部分已有协议只能针对某一种或者几种统计函数，同时需要一个可信任的第三方。文献 [1] 设计一种协议可以计算任意的统计函数并且保护隐私，同时该协议不需要可信任第三方的参与且具有较低的通信复杂度。

（2）在大数据处理方面，通过对数据并行应用进行观察和分析，得出数据并行应用的运行过程具有阶段性的结论。根据每个阶段中任务的数据和数据分区的特点，文献 [2] 研发了基于阶段划分的程序分析优化器 SDPA (Stage-Divided Program Analysis)。同时，针对 I/O 请求具有突发性并且在不同服务器上分配不均以及由于应用的带宽访问模式的限制使得 I/O 带宽分配不合理导致系统整体性能下降的问题，文献 [2] 提出了一个通用的 I/O 带宽控制方案。

（3）在大数据管理方面，针对 SPARQL 查询语言复杂难懂以及无法在合理时间内处理复杂查询的特点，文献 [3] 提出以自然语言为访问接口，设计了基于异构图模式近似匹配方法研究的自然语言问答系统。同时还设计了共享内存环境下图查询并行处理系统 ParTriple[4]。

（4）在大数据分析方面，针对实际流处理应用中，上下流任务在数据处理和任务恢复两方面所具有的强依赖特性，给分布式流处理系统带来了处理延迟高、吞吐率受限、恢复延时长等严重问题，文献 [5] 研发了分布式流处理容错系统 Ares。另一方面，为了衡量查询语句与文档之间的语义相关性，提出基于实体的语言模型平滑方法。文献 [6] 还提出了符合文档语义主题下单词概率分布的实体语义语言模型，以及两层次的平滑方法。

1 低复杂度隐私保护数据聚合算法研究

隐私保护数据聚合问题是一直是应用密码学领域的研究热点。在实际生活中有很多应用，用户需要提供他们敏感的数据到数据中心，数据中心根据收到的用户数据计算相应的统计函数，如最大值、最小值和平均数等。对于这个问题，通常有两个核心的挑战需要解决，即通信开销与数据隐私。如何降低计算过程中通信开销和保护用户数据隐私是该领域研究学者一直不懈探索的目标 [1]。

1.1 研究问题

为解决上述问题，文献 [1] 采用 One Aggregator 模型，即在这个模型中，有一个不可信的聚合中心和 n 个用户 [1]。通过双向的通信链路，聚合中心与每一个用户可以互相发送消息。每个用户定期会产生隐私的数据。根据用户的隐私数据，聚合中心主要负责计算一些统计函数。文献 [1] 假设聚合中心与用户都是半诚实的 (Semi-honest)，即他们忠诚的执行协议，同时在运行过程中想推测其他用户的隐私数据 (有好奇心)。他们彼此之间可以相互同谋，以便获取其余参与方的隐私数据。假设聚合中心和 k 个用户同谋，那么文献 [1] 的研究问题是在无可信任第三方情况下，聚合中心计算任意统计函数，同时保护用户隐私和实现较低通信开销。

1.2 研究策略

根据以上问题描述，可以看出为了使聚合中心

可以计算任意统计函数，一个可行的方法就是直接收集所有用户数据。然而用户数据是隐私的，显然用户直接发送隐私数据到聚合中心的方式不可行。为此文献 [1] 设计一个协议，在无可信任第三方情况下，聚合中心获取所有用户数据，同时保护隐私 (即达到 (*n*-*k*)- 源匿名 ：对于未同谋用户的数据，聚合中心不知道所收到的数据与用户之间对应的关系) 和实现较低的通信开销。

为了保护用户数据隐私，文献 [1] 采用 DH (Diffie-Hellman) 密钥交换协议，使得每个用户产生共享的密钥集。根据所得密钥集，每个用户生成相应的伪随机函数。利用生成的伪随机函数每个用户对数据进行加密。

为了降低通信开销，文献 [1] 设计随机抽样与划分技术使得每个用户获得唯一序列号。具体来说，每个用户在区间 [1, *n*^5] 随机抽样一个随机数，并将此区间划分为 *n* 等长的子区间。当某个子区间所含随机值数量是 1 时，聚合中心保存此子区间；当某个子区间所含随机值数量超过 1 但是小于 $a=\frac{\log n}{\log\log n}$时，聚合中心将此区间均匀划分 a^2 份；当某个子区间所含随机值数量大于 $a=\frac{\log n}{\log\log n}$ 时，聚合中心将此区间均匀划分 $\log^3 n$ 份。聚合中心与用户递归此过程，直到聚合中心保存的子区间的数量等于 *n*。然后，聚合中心发送这 *n* 个子区间到所有用户，每个用户根据这 *n* 个子区间获得自己的唯一序列号。最后根据唯一序列号，聚合中心获取所有的用户隐私数据。通过严格的理论证明，对于隐私保护问题，文献 [1] 设计的协议可以实现 (*n*-*k*)- 源匿名，同时该协议通信复杂度为 $O(n^2(\frac{\log n}{\log\log n})^2)$。

2　面向数据并行的大数据处理技术

2.1　SDPA：数据并行的应用程序分析优化器

在大数据处理系统中，数据并行应用的性能变得越来越重要，这些应用的性能和代码逻辑与运行时的资源利用有关。程序分析技术是一种常用的优化应用的方法，而且这项技术已经被运用到数据并行应用中，但是目前还没有一个能够高效分析整体数据并行应用的方法。

通过对数据并行应用进行观察和分析，我们得到数据并行应用的特点，即数据并行应用的运行过程具有阶段性，并且每个数据并行应用的作业可以划分为多个阶段执行。在每个阶段中会运行多个独立的任务，而任务的数目和数据分区的数目相同，每个任务会计算一个分区的数据，且每个任务对分区中的数据进行同样的操作。根据这个特点，文献 [2] 提出了基于阶段划分的程序分析 (Stage-Divided Program Analysis)，称之为 SDPA，来简化和加速数据并行应用的程序分析 [2]。对于每个阶段，我们对其重新划分为一个或多个运行周期。对于每个周期内的用户定义函数，将其抽取出来并融合为一个方便分析的函数。这样避免了过程间分析且规避开不必要分析的复杂数据处理系统框架代码。

2.1.1 问题分析及解决思路

性能对于数据并行应用程序至关重要，而通过优化代码来提高应用程序的性能是一种好方法。程序分析是一种常用的方法，它分析程序的行为，并获得程序的正确性，安全性和活跃性等属性来优化程序。这些大数据处理系统的编程接口支持数据并行运算符，使得编写大数据应用程序变得很容易，且只需要少量代码就能完成编写工作。但是，这些应用程序的每个运算符都会调用大量的框架代码。因此一个简短的应用程序可能会引用很多类和方法，并且分析这些并行数据应用程序会花费大量时间。

2.1.2 SDPA 的设计与实现

数据并行应用有一个特点，就是在运行阶段会被划分为多个 Stage，而每个 Stage 的任务相对独立，即所有的任务都对不同的数据分区进行相同的计算并且具有相同的 shuffle 依赖。基于这些发现，文献 [2] 提出了 SDPA 来简化加速数据并行应用的程序分

析。SDPA 的处理流程如图 1 所示。

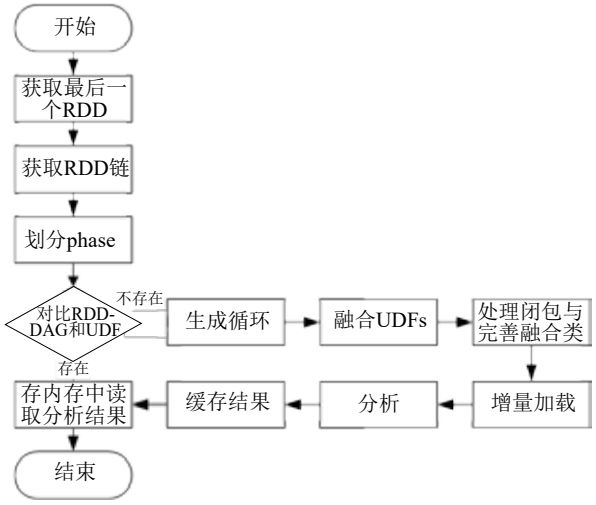

图 1　SDPA 处理流程图

Fig. 1 Flow chart of SDPA

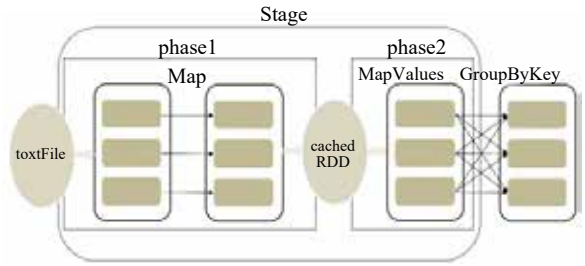

图 2 带有 Cache 的 Phase 划分

Fig. 2 Phase partition with Cache

（1）**Phase** 划分

在 Spark 中，每个 Job 可以划分为一个或多个 Phase。每个 Phase 可以分为三个部分：从数据源读取数据对象，例如缓存 RDD (Resilient Distributed Datasets) 或 Shuffle 缓冲区；对每个数据对象执行一系列的用户自定义函数进行计算；计算结果写入新的数据收集器。Phase 的划分如图 2 所示，按照 Phase 的定义，Phase 与 Spark 划分的 Stage 区别在于，Stage 中间可能有缓存操作，缓存操作会对 Stage 进行隔断，因此一个 Stage 会被划分为一个或多个 Phase。

（2）用户自定义函数融合

每个 Phase，都是对数据源中的数据对象应用一系列用户定义函数的过程。因此，将每个 Phase 中各个算子的用户定义函数抽取出来，然后将这些函数融合成为一个大的函数，可以将这些核心的函数变成一个函数。这样可以跳出算子的束缚，可以更加容易的分析一系列的算子代码。Phase 中对象的操作如图 3 所示。

融合过程使用融合工具 Soot 来实现，然后操作 Soot 生成的中间语言 Jimple 代码。对每个 Phase 的迭代融合机制可概括为以下四个步骤：

- 获取当前 Phase 的最后一个 RDD
- 获得完整的 RDD 链
- 通过 Java 的反射机制抽取出每个 RDD 的 UDF
- 将每个 UDF 的输入输出相连接

融合过程中，需要针对每个算子的语义进行处理。将一个 UDF 的输出作为接下来相邻 UDF 的输入，这样就可以将一系列的 UDF 融合成为一个大的函数，然后使用 Soot 声明一个类，将生成的循环放入这个类中。在后面的程序分析过程中，就可以直接分析这个融合生成的类。

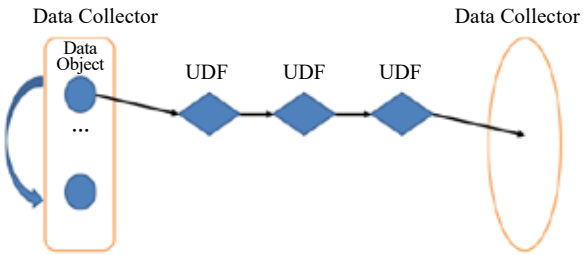

图 3　Phase 中对象操作示意图

Fig. 3 A sketch for data object operation in Phase

（3）代码处理与分析

在 SDPA 中，我们使用 Soot-SPARK 来对融合之后的代码进行程序分析。Soot-SPARK 是一个用 Java 实现指针分析的框架，它支持基于子集的和基于等价的分析以及两者之间的任何分析。SPARK 是作为 Soot 框架的一部分提供的，可以在 soot.jimple.spark.* 包中找到。SPARK 的一部分提供指针分析，并且可以使用选项来控制分析的多个方面。由于使用 Soot-SPARK 分析代码需要一个主类与入口方法，而在对用户定义函数进行融合之后只是得到一

个大的函数，因此需要对融合之后的函数进行处理。SDPA 将融合之后的函数放到一个 Phase-Function-Class 里面，然后添加入口的 apply 方法，然后将这个类和前面的 Phase-Function-Class 链接起来。这就可以使用 Soot-SPARK 进行程序分析。

（4）闭包处理与分析过程加速

在 SDPA 中，闭包处理通过以 Soot 对 Jimple 代码分析为基础。针对不同的 RDD，每个 RDD 都会对应一个算子，对每个算子来说，它生成的 RDD 都是固定类型的，譬如 Map 算子，生成的是 MapParitionRDD。因此，对于这些 RDD，首先进行强制类型转换，得到精确类型的 RDD。得到精确类型之后，通过观察 Jimple 代码，遍历其字段，通过判断字段的 name 是否包含“$outer”来判断这些是否是闭包变量。针对这些闭包变量，使用 Soot 将其注入到最后融合的 Phase-Function-Class 中。

此外，在完成迭代器融合之后需要对融合生成的 Phase-Function-Class 进行程序分析。首先需要对 Phase-Function-Class 需要的类和方法进行加载，而 Soot-SPARK 的加载方式是全量加载，也就是说会加载很多不必要的方法，因此为了加快分析过程，SDPA 实现了增量加载。针对迭代应用，SDPA 实现了 cache code，以方便复用分析结果。

2.2 海量存储系统中的软件定义 QoS 机制技术

随着数据信息的快速发展，对大容量存储系统的需求日益增长，海量存储系统逐渐成为研究热点。海量存储系统由客户端、管理节点、存储节点等三种节点组成，结构图如图 4 所示。管理节点存储命名空间和系统的元数据，如文件名、其对应的对象号、以及该对象保存在哪个数据节点上等信息；数据节点保存文件数据（对象），对象以对象号标识应用服务器通过定制协议或 HTTP 访问存储系统。海量存储系统的核心是分布式文件系统，文件系统允许从一个计算机网络中的多台主机上访问文件，这使得多个用户可以通过多台主机共享文件及存储资源。

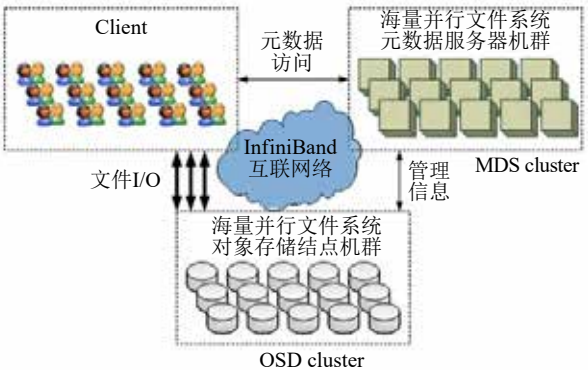

图 4　海量存储系统

Fig. 4　Mass storage system

海量存储系统的发展也使得应用越来越倾向于数据密集型，导致高度密集的海量数据 I/O 吞吐需求日益增长。同时半导体技术的飞速发展，使得固态硬盘（Solid State Drives）得到迅速普及和广泛应用。但是尽管新的高速存储设备的出现和部署使得 HPC 系统的整体 I/O 性能稳步提高，但是相比于计算性能的增长，I/O 资源仍处于不足的状态。这使得多应用之间对于存储资源的竞争和干扰极大程度上影响应用的性能，譬如一个规模庞大的作业因为等待缓慢的 I/O 而被阻塞，又或者一个小的作业在运行时产生了大量 I/O 阻塞了其他优先级较高的应用。因此可以得出 I/O 的分配不合理不仅是存储资源的低利用率，也是对计算资源的浪费。所以，使得每个应用占据与它的计算资源匹配的 I/O 带宽对于 HPC 存储系统来说很有必要。

2.2.1 问题分析及解决思路

在 HPC 平台上实现 I/O 控制面临很多挑战：第一，HPC 应用可以轻易达到上千的计算节点的规模，每秒产生百万级别的 I/O 请求。这使得全局上监控应用的 I/O 行为十分困难。同时搜集到应用所有线程的 I/O 行为会造成监测规模过于庞大的问题。第二，I/O 请求具有突发性并且在不同服务器上分配不均。这些特性要求 I/O 控制做到细粒度（应对 I/O 突发性）和全局优化（应对 I/O 分配不均）。在这两点上实现具有较大难度，同时还要求服务器去学习应用的 I/O

变化规律也十分困难。第三，保障应用的带宽并不是简单的设置一个硬性的标准，同时需要使得整个系统的 I/O 性能达到最优，例如，当应用的带宽被自身的访问模式所限制无法达到理想带宽时，余下的带宽就应当分配给其他应用使用以达到整体性能最优。

为此，文献 [7] 提出了一个通用的 I/O 带宽控制方案，基于 OrangeFs 并行文件系统实现，其核心思想是在 HPC 存储系统中加入 Data Plane 和 Control Plane 组件去控制应用的 I/O 行为，同时使用令牌桶技术达到多应用间的公平性和整体性能最优，架构如图 5 所示 [7]。

2.2.2 SDQoS 设计与实现

(1) 数据模块和控制模块

每一个服务端维持一个 Data Plane，是监测和控制应用 I/O 行为的核心组件，同时这一系列操作会在实际 I/O 行为发生之前进行处理。该组件的实现在不同的文件系统上会有不同的实现方式，例如在 lustre 文件系统上，Data Plane 可以实现在 OST 层，在 pvfs 文件系统中，文献 [7] 实现在服务端。

Data Plane 首先与 Control Plane 进行连接，获取到应用的 I/O 需求，并为每个应用维护一个单独的队列。然后，每个到来的 I/O 请求从令牌桶中获得令牌，根据获得令牌的情况被服务。

Control Plane 在整个系统运行过程中是全局且唯一的，它并不是单独运行在某个节点上，而是逻辑上存在的一个分布式调度组件，用于接收应用的特性并基于这些特性计算出应用应该被分配的带宽（也可以直接指定带宽）。需要指出的是，这些被计算出的带宽并不是应用强制需要达到的带宽。由于实际 I/O 过程的不均衡以及服务器能提供带宽的物理限制，应用实际得到的带宽也会进行调整。

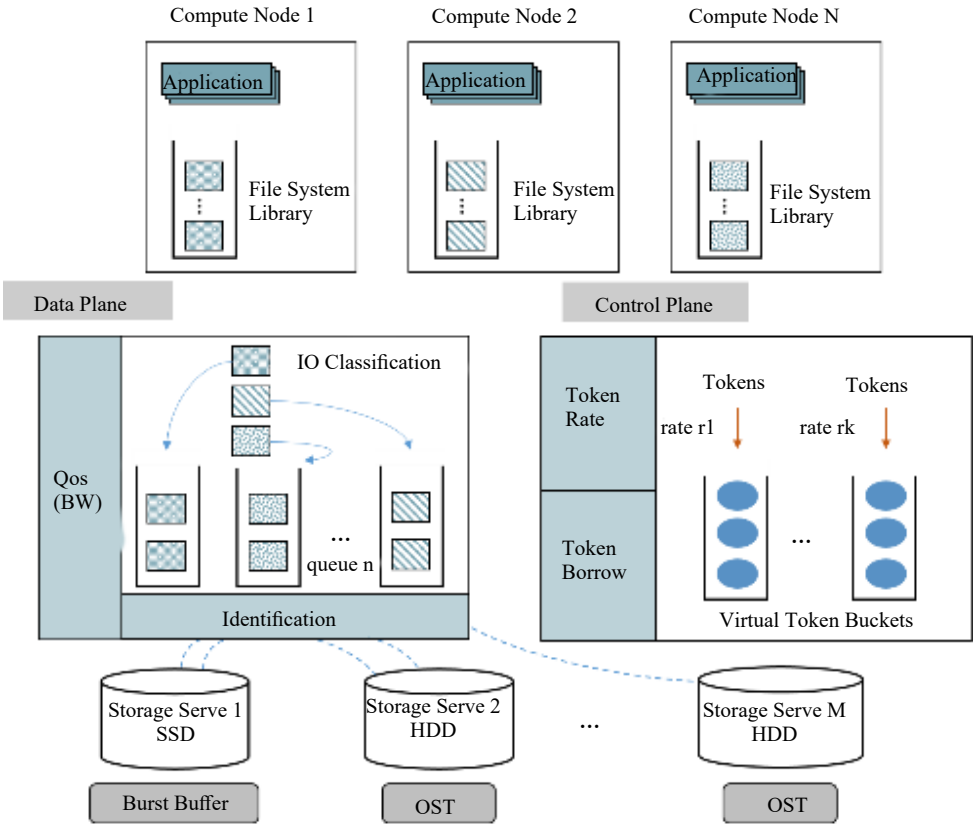

图 5　SDQoS 框架

Fig. 5.　SDQoS framework

(2) 本地 I/O 控制

在 SDQoS 中，文献 [7] 利用令牌桶去限制单个应用的 I/O 带宽。

令牌桶策略在网络中被广泛用于管理和整形网络数据包，它能够应对突发流量同时保障了常规流量。因为在 HPC 中，I/O 同样存在突发现象，所以该令牌桶策略在 HPC 环境下的 I/O 场景中同样适用。

令牌桶策略在 HPC 中的使用和在网络应用中的使用有很大的区别。在网络中，超出限制的数据包可以被丢弃重发，而在 HPC 中，I/O 不可以被丢弃，需要被缓存起来，这样就会造成延迟和阻塞。并且，有时需要允许应用得到超过它得到的令牌数的 I/O 资源以达到整个系统 I/O 性能最优。

在具体实现上，我们为每个应用都分配了单独的令牌桶。令牌桶根据应用的需求设定令牌产生速率以及令牌桶容量。其中，令牌产生速率代表了应用稳定情况下的带宽，令牌桶容量代表了应用在突发情况下所能分配的最大带宽。

3 RDF 图数据的查询与匹配

由于图数据模型具有描述能力强，适用范围广等优点，越来越多的数据使用图数据模型发布或存储 [8-10]。资源描述框架（Resource Description Framework，RDF）数据是图数据中的一种。由于 RDF 数据模型有着诸多优点，譬如方便简洁，模块化程度高等，越来越多的机构或组织加入了 RDF 用户社区。

SPARQL 语言是一种常用的图查询语言。SPARQL 查询语句通常构成一个查询图，表示在对应 RDF 图数据上的子图匹配查询。SPARQL 是知识图的固有访问接口，但复杂难懂，文献 [3] 提出以自然语言为访问接口，设计了基于异构图模式近似匹配方法的自然语言问答系统。此外，SPARQL 查询引擎无法在合理时间内处理复杂查询，文献 [4] 提出了共享内存环境下图查询并行处理系统 ParTriple。

3.1 异构图模式近似匹配方法

3.1.1 研究问题

随着互联网的飞速发展，大量的知识库可供公众访问，例如 DBpedia，YAGO，Freebase。不同于传统的信息检索中使用的文档库，知识库以 RDF 形式，即三元组（主语，谓语，宾语）的形式整合细粒度的信息，因此它在整合信息、加强智能搜索方面担任着非常重要的角色 [4]。

然而，由于知识库采用普通用户不熟悉的结构化的方式来存储信息，所以访问使用这些大规模知识库中的信息成为当前困扰普通用户的问题 [11-12]。另外，因为知识库以图的结构来组织信息，所以，问题的本质也就是图模式的近似匹配问题。但是为了连接用户和知识图，需要有一个访问接口来方便用户使用。

虽然知识图提供了固有的访问接口，例如 SPARQL，但是复杂的语法和知识图结构使得它不能直接为普通用户所用。因此，为知识图设计一个高效且方便用户使用的访问接口成为人们亟待解决的问题。由于以自然语言为访问接口，方便用户直接从知识库中获取信息，所以文献 [3] 提出了基于异构图模式近似匹配方法，并在此基础上实现了自然语言问答系统 Svega[3]。

3.1.2 基于语义向量的图模式匹配与相似度计算

给定一个查询图 $Q=(V, E, RE)$，其中 $V=v1, v2, \cdots, vn$，节点集合 V 中的每一个节点 vi 在知识库中都有一个匹配的候选结合 Svi，则当且仅当满足以下两个条件时，知识库中的一个子图 sub-KG 才是查询图 Q 的匹配子图：

（1）对于查询图 Q 中的任何一个节点 vi，一定有一个节点 $ui \in Svi$，并且该节点被包含在子图 sub-KG 中；

（2）如果查询图 Q 中的两个节点 vi 和 vj 之间存在一条边，则在子图 sub-KG 中一定存在一条路径 $P(ui, uj)$，并且节点 ui 和 uj 分别是查询图中节点 vi 和 vj 对应的匹配节点。

针对异构图模式近似匹配问题，文献 [3] 提出了

一种以图模式近似匹配技术来访问知识图的方法。为了更加精准的挖掘出查询目的，文献 [3] 提出了一种实体驱动的策略来构建查询图。首先基于查询图中节点特征识别出自然语言问句中的实体，然后根据问句中单词之间的语法关系分析挖掘出实体之间的谓语关系，从而构建出查询图。就图模式匹配过程而言，其本质就是消歧处理，所以该系统首先基于查询图的结构特征进行消歧，提出了基于节点匹配来从知识库中得到查询图的异构子图，同时挖掘匹配的路径，过滤歧义信息，然后基于查询图的语义特征进行消歧，提出了合成词向量的方法来表示查询图中边和匹配子图中路径的语义信息，从而简化了相似度的计算，同时可以根据语义相似度来过滤歧义信息。

3.2 共享内存环境下图查询并行处理系统 ParTriple

3.2.1 研究问题

随着 RDF 数据的爆炸式增长，现有的一些 SPARQL 查询处理系统已无法在合理时间内处理复杂查询。为此，SPARQL 查询引擎应当使用并行处理技术来实现对复杂查询的高效处理。然而，当前的一些 SPARQL 查询并行处理技术有着查询优化效率低下且生成的查询计划不利于并行，系统并行程度较低等不足。在这种背景下，将 SPARQL 图查询处理与并行技术相结合，开发 SPARQL 图查询并行处理系统，即共享内存环境下的图查询并行处理系统，显得尤为必要。

开发图查询并行处理系统主要有两方面的挑战。一方面，查询优化对查询处理的性能有很大影响，其生成的查询计划不可避免地影响查询执行的并行处理。另一方面，由于现代计算机上配备的处理核心数目越来越多，如何在单机上最大限度地利用这些计算能力来进行 SPARQL 查询处理本身也有很多难点，譬如并行性开发不够，资源竞争过多等等。为了应对这些挑战，文献 [4] 在高可扩展的 RDF 数据存储查询系统 TripleBit 的基础上设计并实现了 ParTriple 系统 [4]。

3.2.2 并行处理系统 ParTriple

ParTriple 旨在为多核计算机环境下的 SPARQL 图查询提供并行处理。文献 [3] 提出了一种基于自底向上树的查询计划生成方法。另一方面，文献 [4] 提出了一种两级并行模型来执行查询计划，以充分利用多核架构的计算能力，提高查询处理性能。

ParTriple 系统主要由数据导入模块、数据结构模块、查询处理模块三部分组成。ParTriple 通过数据导入模块，对 RDF 数据进行解析、字典映射，并进行排序。在此期间，数据结构模块与数据导入模块进行交互，建立数据字典，存储三元组数据，并建立相应的索引。查询处理模块用于处理用户提交的 SPARQL 查询语句。可分成 SPARQL 语句解析、查询优化器、任务管理、并行操作符等子模块。

首先，该系统提出了一种高效、易于并行的查询计划生成算法。这种算法生成的查询计划可以使用流水线处理技术，且不同的流水线之间不需要相互等待。进一步地，这种查询计划可以更有效地削减中间结果，提高查询处理性能。接着，我们提出了一种数据块级与模式级的两级并行执行框架，将查询执行分解成基于数据存储分块的子任务与基于中间结果分块的子任务，从操作符间并行与操作符内并行两个层面开发了系统的并行性。最后，我们开发了三种不同的并行操作符，分别针对数据扫描，连接处理，数据洗牌等操作，进一步提升了系统查询处理的性能。

4 大数据分析技术

随着信息技术的迅猛发展，越来越多的应用会产生海量的数据集。这些应用需要对大数据进行快速计算、分析与处理，给大数据计算理论框架带来了高吞吐、低延时、高可扩展、高容错等迫切需求。近年来，以 Hadoop、Spark 和 Storm 为代表的大数据计算理论框架受到业界的广泛关注并投入使用。

Hadoop 是一种专用于批处理的计算框架，适用于需要分析海量的离线数据集的应用场景。Hadoop 实现了 MapReduce 的思想，通过数据切片计算来

处理海量的离线数据集，具有高可扩展性。由于Hadoop需要将所有中间输出和结果持久化到磁盘，频繁的磁盘IO操作会影响计算的速度。Spark是一种基于内存计算的批处理计算框架，适用于需要反复操作特定数据集的应用场景。Spark通过RDD（Resilient Distributed Dataset，弹性分布式数据集）将计算的中间输出和结果存储在内存中，能够显著加速Hadoop作业的执行速度。Storm是一种专用于流处理的计算框架，适用于需要对海量连续流数据进行实时处理的应用场景。Storm通过将应用抽象成DAG（Directed Acyclic Graph，有向无环图），并利用分布式RPC（Remote Procedure Call，远程过程调用）框架来支撑无边界流数据的实时处理，具有非常低的延时。由于DAG中上下流任务的依赖关系会导致在任务恢复期间发生任务级联等待的现象，影响处理性能，文献[5]通过挖掘上下流任务的依赖关系，设计了一个低延时和高容错的分布式流处理系统。

4.1 分布式流处理容错系统 Ares

为了满足实时流处理应用的低延迟和高容错的需求，分布式流处理系统需要保障系统能够以低延迟的方式处理海量的实时数据流，同时当系统中的节点发生故障时，系统能够在较短的时间内恢复正常。现有的分布式流处理系统通常通过优化任务调度来实现海量实时数据流的低延迟处理。典型的分布式流处理系统任务调度策略通过将流处理应用拓扑中的上下流任务进行绑定并分配到同一节点的方式，降低上下流任务之间的数据传输时间，从而保障低延迟处理性能[13-15]。然而，该任务调度策略忽视了上下流任务之间的依赖关系对系统容错性能的影响。即当系统中的节点发生故障时，由于失效节点上存在大量的上下流任务，这些任务之间的依赖关系会导致在任务恢复期间发生任务级联等待的现象，从而使得系统无法在较短的时间内恢复正常[16-17]。

该研究指出实现一个低延迟和高容错的分布式流处理系统的关键在于能否在任务调度过程中充分挖掘流处理应用拓扑中上下流任务之间的依赖关系。为此，文献[5]研发了分布式流处理容错系统Ares[5]。Ares在系统任务调度过程中兼顾上下流任务

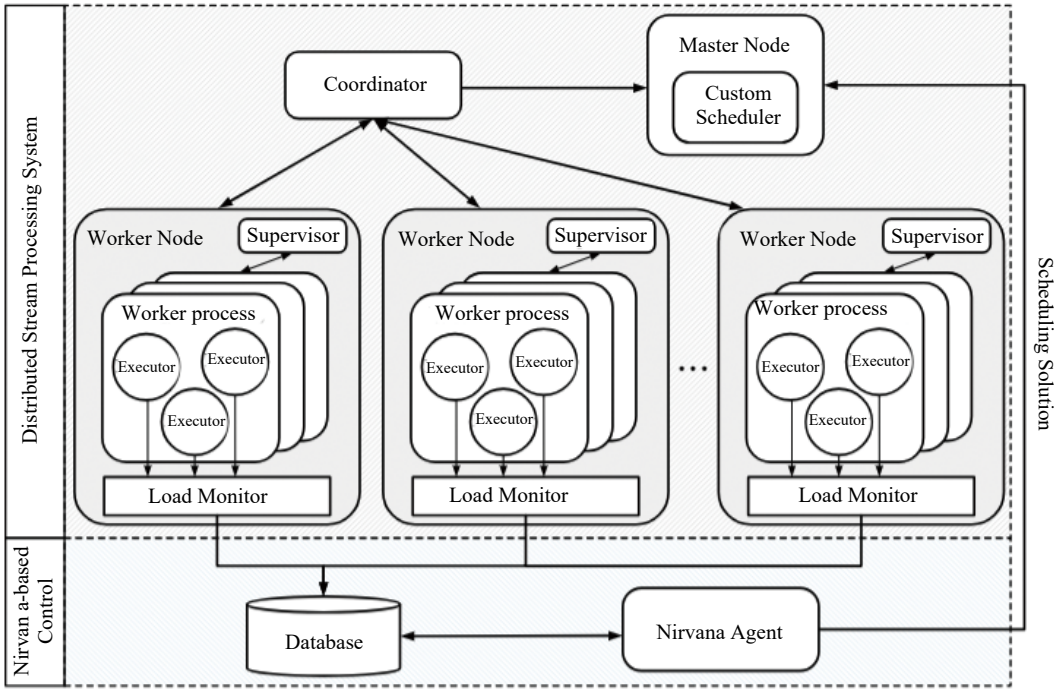

图6 Ares系统架构

Fig. 6 Ares framework

之间的依赖关系对处理性能和容错性能两者的影响，并提出了一种容错调度问题来同时优化处理延迟和恢复时间。为了最大化系统效用，Ares 利用单一任务的最佳放置策略往往取决于该任务的上下流任务的放置策略的特性，设计了一种基于最佳应对动态算法 Nirvana 来自动优化任务放置策略。相对于目前最新流处理系统，Ares 将系统总体吞吐率提高了 3.6 倍，将平均处理延迟降低了 50.2%，将平均恢复时间降低了 52.5%。

图 6 显示了 Ares 的系统架构。Ares 的架构在传统分布式流处理系统架构的基础上，增加了一个负载监视器和一个 Nirvana 控制器。负载监视器负责实时监控所有计算节点上的任务的资源使用情况。Nirvana 控制器根据当前系统的资源使用情况，通过 Nirvana 算法计算得到满足低延迟和高容错需求的最佳任务调度方案。

4.2 面向文档过滤的语言模型平滑方法

丰富的网络资源中蕴含着海量的数据信息，帮助用户从中快速、准确的找到所需的信息是一项极具价值的任务。但是海量的数据规模以及自然语言表达带来的语义歧义性和多样性，给信息系检索带来了巨大的挑战。文献 [6] 以实体为桥梁，以维基百科中的实体信息为内容，提出了符合文档语义主题下单词概率分布的实体语义语言模型，并且提出两层次的平滑方法，结合文档无关的全局语料库信息源和文档主题相关的实体语义语言模型信息源来对原始文档语言模型进行平滑，使得平滑后的语言模型能够很好的衡量查询语句和文档之间的语义相关性，提高了检索系统的性能。

4.2.1 实体语义语言模型的构造

文档中的实体被链接到维基百科中。但是由于实体链接工具不是完美的，存在错误的可能，不能保证百分百的正确率。为了处理这种不确定性，文献 [6] 提出了硬合并（Hard-Fused Method）和软合并（Soft-Fused Method）两种方法。硬合并方法只考虑文档中置信度比较高的实体，同时忽略实体的频率信息。软合并方法将置信度作为实体的权重，同时考虑实体的频率信息。

硬合并方法仅考虑链接置信度大于特定阈值的实体，并且忽略实体在文档中出现的次数。实体链接工具的精度有限，不可避免的会产生错误。直接忽略低置信度的实体可以预防引入实体链接工具的错误，只使用那些质量高的、信得过的链接实体可以提高模型的健壮性。硬合并方法同时将所有高置信度的实体当作一样来对待，分配同样的权重，即忽略它们在文档中的频率信息。

另外一种构建实体语义语言模型的方法是软合并方法，这种方法考虑实体链接结果中的置信度等级和实体在文档中频率信息。文档中实体的重要程度是不一样的。重要实体的知识库信息与文档的语义主题更加相关。用 SF-ESLM (Soft Fused-Entity Semantic Language Model) 来表示软合并方法生成的实体语义语言模型：

$$P_{SF-ESLM}(w|d)=\frac{1}{\sum_{m'\in M(d)}p(m')}\sum_{m\in M(d)}p(m)*P_{MLE}(w|Wiki(e(m)))$$

其中 $P_{SF\text{-}ESLM}(w|d)$ 表示单词 w 在文档 d 软合并方法得到实体语义语言模型中的概率大小，$M(d)$ 表示文档 d 中所有实体提及名字组成的集合。

4.2.2 两层次的平滑方法

基于实体的语言模型平滑框架从文档语义主题的角度出发，利用实体作为桥梁，构建实体语义语言模型。在这个方法中，文档中的实体被实体链接工具链接到外部实体知识库中，如维基百科，然后使用软合并方法和硬合并方法将实体在知识库中的信息合并在一起组成实体语义语言模型。文献 [6] 最后提出了两层次的平滑方法，将上述语言模型结合在一起，可以很好的区分单词与文档主题的相关性。

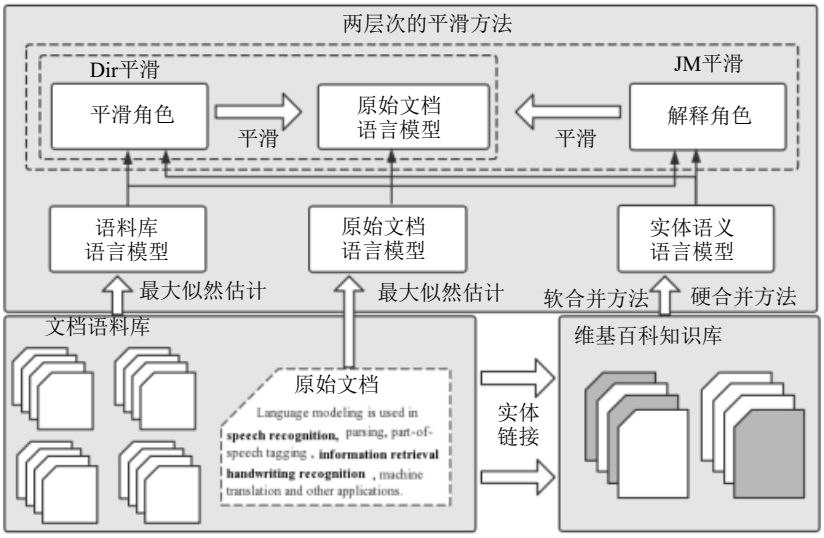

图 7　基于实体的语义平滑方法示意图

Fig. 7 A sketch for entity-based language model smoothing.

如图 7，整个框架包含原始文档语言模型、实体语义语言模型和语料库语言模型三个语言模型。原始文档语言模型通过统计原始文档中词频信息来计算得到。实体语义语言模型通过文档中实体指向的维基百科主页信息来计算得到。全局语料库语言模型通过统计语料库中的全局词频信息来计算得到。实体语义语言模型是整个框架的核心。它引入与文档主题相关的语义信息，使得平滑后的语义模型更加接近文档主题下的单词概率分布。

5　总结与未来工作

本文简要介绍了大数据基础理论以及大数据的存储、检索、分析和挖掘等关键技术，并介绍了低复杂度隐私保护数据聚合算法、面向数据并行的大数据处理技术、RDF 图数据查询与匹配以及大数据分析技术等几个方面的一些代表性工作。

未来，我们将在已有工作的基础上继续关注大数据基础理论以及大数据系统的前沿技术发展趋势。在大数据基础理论方面，我们将在结合新型体系结构的基础上，进一步研究大数据代数系统、数学结构以及大数据度量指标等方面的基础科学问题，从大数据的内在结构、本质特征出发，为大数据的分析、计算提供基础理论支撑。在大数据分析方面，我们将继续探索大数据分析技术的一些共性问题，例如新型数据挖掘算法、高维数据分析技术、基于深度学习的大数据处理技术以及多源异构复杂数据的治理和分析技术。在大数据计算技术方面，我们将继续关注新型体系结构上的高能效大数据计算技术、以性能为导向的新型大数据计算框架，同时我们将在继续关注大数据基础理论与系统关键技术的基础上，引入边缘计算、雾计算等场景，研究物联网环境下的大数据计算技术。最后，在研究大数据分析、计算等关键技术的同时，我们将在国家重点发展战略的指导下，关注与国计民生息息相关重点大数据应用系统，例如基于大数据挖掘的大健康分析系统、基于大数据处理的国防安全战略防御系统等。

利益冲突声明

所有作者声明不存在利益冲突关系。

参考文献

[1] Xuhui Gong, Qiang-Sheng Hua, Lixiang Qian, Dongxiao Yu, Hai Jin. Communication-Efficient and Privacy-Preserving Data Aggregation without Trusted Authority. In *Proceedings of 2018 IEEE Conference on Computer Communications (INFOCOM 2018)*, Honolulu, United States, 2018, pp. 1250-1258.

[2] Fei Wang, Xuanhua Shi, Dongxiao Yu, Zhixiang Ke, Hai Jin, Song Wu. SDPA: An Optimizer for Program Analysis of Data-Parallel Applications. In *Proceedings of the 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS 2018)*, Exeter, United Kingdom, 2018, pp. 14-21.

[3] Gaofeng Li, Pingpeng Yuan, and Hai Jin. Svega: Answering Natural Language Question over Knowledge Base with Semantic Matching. In *Proceedings of The 30th International Conference on Software Engineering and Knowledge Engineering (SEKE 2018)*, California, United States, 2018, pp. 615-616.

[4] Wang, Tao, Pingpeng Yuan, Xiaofei Liao and Hai Jin. Parallel Processing SPARQL Theta Join on Large Scale RDF Graphs. In *Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM).* Abu Dhabi, United Arab Emirates, 2018, pp. 1-6.

[5] Changfu Lin, Jingjing Zhan, Hanhua Chen, Jie Tan, Hai Jin. Ares: A High Performance and Fault-tolerant Distributed Stream Processing System. In *Proceedings of the 26th IEEE International Conference on Network Protocols (ICNP 2018)*, Cambridge, United Kingdom, 2018, pp. 176-186.

[6] Feng Zhao, Zeliang Tian and Hai Jin. Entity-Based Language Model Smoothing Approach for Smart Search. *IEEE Access*, vol. 6, pp. 9991-10002, 2018.

[7] Yusheng Hua, Xuanhua Shi, Hai Jin, Wei Liu, Yan Jiang, Yong Chen, Ligang He. Software-defined QoS for I/O in exascale computing. *CCF Transactions on High Performance Computing*, Vol. 1, No. 1, pp 49–59.

[8] Hao Long, Pingpeng Yuan and Hai Jin. Distributed PathGraph: A Cluster Centric Framework for Distributed Processing Graph. In *Proceedings of the 10th IEEE Conference on Service-Oriented Computing and Applications (SOCA 2017)*, Kanazawa, Japan, 2017, pp. 34-41. 2017.

[9] Xuanhua Shi, Xuan Luo, Junling Liang, Peng Zhao, Sheng Di, Bingsheng He, Hai Jin. Frog: Asynchronous Graph Processing on GPU with Hybrid Coloring Model. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 30, no. 1, pp. 29-42, 1 Jan. 2018.

[10] Xuanhua Shi, Zhigao Zheng, Yongluan Zhou, Hai Jin, Ligang He, Bo Liu, Qiang-Sheng Hua. Graph Processing on GPUs: A Survey. *ACM Computing Surveys*. Vol.50, No.6, pp.1-35, 2018.

[11] Pingpeng Yuan, Pu Liu, Buwen Wu, Hai Jin, Wenya Zhang, Ling Liu. TripleBit: a Fast and Compact System for Large Scale RDF Data. PVLDB, 6(7): 517-528, 2013.

[12] Mazzeo Giuseppe, Carlo Zaniolo. Answering Controlled Natural Language Questions on RDF Knowledge Bases. *In Proceedings of the 19th International Conference on Extending Database Technology (EDBT 2016)*, March 15-18, 2016, Bordeaux, France. pp. 608-611.

[13] Fangjia Xing, Liming Gui, Hanhua Chen, Changfu Lin, Hai Jin. Efficient Event Stream Dissemination in Online Social Networks Based on Community Detection. In *Proceedings of the 2018 IEEE International Conference on Communications (ICC)*, Kansas City, United States, 2018, pp. 1-6.

[14] Sijie Wu, Hanhua Chen, Changfu Lin, Hai Jin. Shadow: Exploiting the Power of Choice for Efficient Shuffling in MapReduce. In *Proceedings of the 23rd IEEE International Conference on Parallel and Distributed Systems (ICPADS 2017)*, Shenzhen, China, 2017, pp. 553-560.

[15] Fan Zhang, Hanhua Chen, Hai Jin. Piggyback Game: Efficient Event Stream Dissemination in Online Social Network Systems. *IEEE Transactions on Parallel and*

Distributed Systems(TPDS), vol. 30, no. 3, pp. 692-709.

[16] Hanhua Chen, Fan Zhang, Hai Jin. Popularity-aware Differentiated Distributed Stream Processing on Skewed Streams, In *Proceedings of the 25th IEEE International Conference on Network Protocols (ICNP 2017)*, Toronto, Canada, 2017, pp. 1-10.

[17] Hanhua Chen, Liangyi Liao, Hai Jin, Jie Wu. The Dynamic Cuckoo Filter. In *Proceedings of the 25th IEEE International Conference on Network Protocols (ICNP 2017)*, Toronto, Canada, 2017, pp. 1-10.

收稿日期：2019年8月30日

华强胜，1979年生，博士，华中科技大学计算机学院副教授，博士生导师。主要研究方向为分布式计算理论与算法，算法与体系结构协同设计。

本文承担工作为：大数据基础理论分析与讨论以及全文统筹。

Hua Qiangsheng, PhD, was born in 1979. He is an associate professor in School of Computer Science and Technology, Huazhong University of Science and Technology. His research interests include distributed computing theory and algorithms, co-design of algorithms and computer architecture.

Role in this paper: Responsible for writing the big data theory part and the paper organization.

E-mail: qshua@hust.edu.cn

金海，1966年生，博士，华中科技大学计算机学院教授，博士生导师，CCF 会士，IEEE 会士。主要研究方向为并行与分布式计算。

本文承担工作为：全文统筹以及大数据关键技术分析与讨论。

Jin Hai, PhD, was born in 1966. He is a professor and PhD supervisor in School of Computer Science and Technology, Huazhong University of Science and Technology. He is a CCF Fellow and an IEEE Fellow. His research interests are on parallel and distributed computing.

Role in this paper: Responsible for coordinating the paper writing and attending technical discussion on technologies for big data.

E-mail：hjin@hust.edu.cn

郑志高主要承担了文章的文献调研与回顾，全文整理与撰写工作。胡振宇主要承担了文章中面向数据并行的大数据处理技术分析与讨论。钟芷漫主要承担了文章中RDF图数据查询与匹配关键技术分析与讨论。林昌富主要承担了文章中分布式流处理系统关键技术分析与讨论。赵峰主要承担了文章中面向文档过滤的语言模型分析与讨论。

石宣化主要承担了文章中大数据分析与处理关键技术分析与讨论。

Zheng Zhigao mainly undertakes the literature research and review of the article, and collates and writes the full text. Hu Zhenyu mainly undertakes analysis and discussion of big data processing technology oriented to data parallelism in the article. Zhong Yuman mainly undertakes the analysis and discussion of the key technologies of RDF graph data query and matching in the article. Lin Changfu mainly undertakes the analysis and discussion of key technologies of distributed stream processing systems in the article. Zhao Feng mainly undertakes the analysis and discussion of the language model for document filtering in the article.Shi Xuanhua mainly undertakes the analysis and discussion of key technologies in big data analysis and processing in the article.

引文格式：华强胜,郑志高,胡振宇,钟芷漫,林昌富,赵峰,金海,石宣化.大数据基础理论与系统关键技术浅析[J]. 数据与计算发展前沿,2019,1(1):22-34. DOI:10.11871/jfdc.issn.2096-742X.2019.01.004.PID: 21.86101.2/jfdc.2096-742X.2019.01.004.