DOI:10.3969/j.issn.1009-6868.2016.02.004
网络出版地址:http://www.cnki.net/kcms/detail/34.1228.tn.20160223.1346.002.html

典型大数据计算框架分析
Typical Big Data Computing Frameworks

赵晟/ZHAO Sheng
姜进磊/JIANG Jinlei
(清华大学 计算机科学与技术系,北京 100084)
(Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

中图分类号:TP393 文献标志码:A 文章编号:1009-6868 (2016) 02-0014-005

摘要: 认为大数据计算技术已逐渐形成了批量计算和流计算两个技术发展方向。批量计算技术主要针对静态数据的离线计算,吞吐量好,但是不能保证实时性;流计算技术主要针对动态数据的在线实时计算,时效性好,但是难以获取数据全貌。从可扩展性、容错性、任务调度、资源利用率、时效性、输入输出(IO)等方面对现有的主流大数据计算框架进行了分析与总结,指出了未来的发展方向和研究热点。

关键词: 大数据分类;大数据计算;批量计算;流计算;计算框架

Abstract: Big data computing technologies have two typical processing modes: batch computing and stream computing. Batch computing is mainly used for high-throughput processing of static data and does not produce results in real time. Stream computing is used for processing dynamic data online in real time but has difficulty providing a full view of data. In this paper, we analyze some typical big data computing frameworks from the perspective of scalability, fault-tolerance, task scheduling, resource utilization, real time guarantee, and input/output (IO) overhead. We then points out some future trends and hot research topics.

Key words: big data; big data computing; batch computing; stream computing; computing framework

近年来,随着互联网进入Web 2.0时代以及物联网和云计算的迅猛发展,人类社会逐渐步入了大数据时代。根据维基百科的描述,所谓的大数据,是指所涉及的数据量规模巨大,无法通过人工在合理时间内达到截取、管理、处理、并整理成为人类所能解读的信息。大数据在带来发展机遇的同时,也带来了新的挑战,催生了新技术的发展和旧技术的革新。例如,不断增长的数据规模和数据的动态快速产生要求必须采用分布式计算框架才能实现与之相匹配的吞吐和实时性,而数据的持久化保存也离不开分布式存储。

图1展示了大数据应用的一般架构,其中的核心部分就是大数据计算框架和大数据存储。大数据存储提供可靠的数据存储服务,在此之上搭建高效、可扩展、可自动进行错误恢复的分布式大数据计算框架,计算依赖存储,两者共同构成数据处理的核心服务。由于文献[1]已经对大数据存储进行总结,详述了文件系统、数据库系统、索引技术,因此文中将重点对大数据计算框架进行分析。

1 大数据计算技术面临的问题与挑战

大数据计算技术采用分布式计算框架来完成大数据的处理和分析任务。作为分布式计算框架,不仅要提供高效的计算模型、简单的编程接口,还要考虑可扩展性和容错能力。作为大数据处理的框架,需要有高效可靠的输入输出(IO),满足数据实时处理的需求。当前大数据处理需要

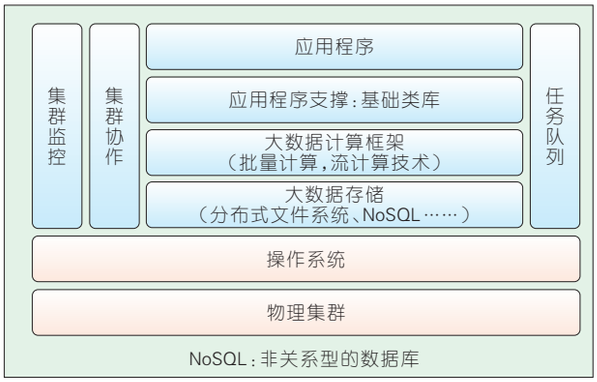


图1▶ 大数据应用的一般性架构

收稿时间:2016-01-10
网络出版时间:2016-02-23
基金项目:国家高技术研究发展("863")计划(2013AA01A213);国家自然科学基金 (61572280、61433008、U1435216、61373145)

解决如下问题和挑战，这些问题和挑战也是对大数据计算框架进行分析的重要指标。

（1）可扩展性：计算框架的可扩展性决定可计算规模，计算并发度等指标。现有计算框架通常采用主从模式的架构设计，便于集群的管理和任务调度，但主节点会成为系统的性能瓶颈，限制了可扩展性。另外，在现有弹性计算集群部署中，不断动态添加、删除计算节点，快速平衡负载等也对系统可扩展性提出挑战。

（2）容错和自动恢复：大数据计算框架需要考虑底层存储系统的不可靠性，支持出现错误后自动恢复的能力。用户不需要增加额外的代码进行快照等中间结果的备份，只需要编写相应的功能函数，就可以在有输入的条件下得到预期的输出，中间运行时产生的错误对使用人员透明，由计算框架负责任务重做。

（3）任务调度模型：大数据计算平台中往往存在多租户共同使用，多任务共同执行的情况。既要保证各用户之间使用计算资源的公平性，又要保证整个系统合理利用资源，保持高吞吐率，还要保证调度算法足够简单高效，额外开销小。因此调度器设计需要综合大量真实的任务运行结果，从全局的角度进行设计。

（4）计算资源的利用率：计算资源的利用率代表机器能够实际创造的价值。数据中心运转时，能耗问题非常突出，设备和制冷系统都在消耗能源。由于不合理的架构设计，导致集群中非计算开销大，计算出现忙等待的现象时有发生。高效的计算框架需要和硬件环境共同作用达到更高的计算资源利用率。

（5）时效性：数据的价值往往存在时效性，随着时间的推移，新数据不断产生，旧数据的利用价值就会降低。离线批量处理往往导致运算的时间长，达不到实时的数据处理。流计算方案减少了响应的时间，但是不能够获得数据的全貌。因此增量计算的方法是当今的一个解决思路。

（6）高效可靠的IO：大数据计算中，IO开销主要分为两部分，序列化反序列化时数据在硬盘上读写的IO开销，不同节点间交换数据的网络IO开销。由于硬盘和网络的IO读写速率远远低于内存的读写速率，导致整个任务的执行效率降低，计算资源被浪费。在现有的计算机体系结构下，尽可能使用内存能够有效提高处理的速度，但是预取算法的合理性和内存的不可靠性都是需要考虑的问题。

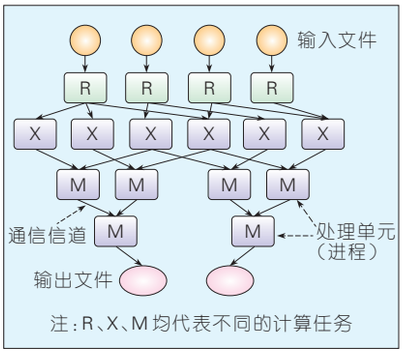
2 大数据批量计算技术

大数据批量计算技术应用于静态数据的离线计算和处理，框架设计初衷是为了解决大规模、非实时数据计算，更加关注整个计算框架的吞吐量。MapReduce低成本、高可靠性、高可扩展的特点降低了大数据计算分析的门槛，自Google提出以来，得到了广泛应用。在此基础上，人们设计出众多的批处理计算框架，从编程模型、存储介质等角度不断提高批处理的性能，使其适应更多的应用场景。

（1）MapReduce计算框架：MapReduce计算框架通过提供简单的编程接口，在大规模廉价的服务器上搭建起一个计算和IO处理能力强大的框架，并行度高，容错性好，其开源项目Hadoop已经形成完整的大数据分析生态系统，并在不断改进。可扩展性方面，通过引入新的资源管理框架YARN，减轻主节点的负载，集群规模提高，资源管理更加有效。任务调度方面，提出如公平调度[2]、能力调度[3]、延迟调度[4]等调度器，更加关注数据中心内资源使用的公平性、执行环境的异构性和高吞吐的目标。另外也采用启发式方法进行预测调度，能够实时跟踪节点负载变化，提供更优的执行序列和资源分配方案。容错性方面，MapReduce框架本身支持任务级容错，任务失败后会重新计算，但是对于Master节点的容错一直忽略，现有的解决方法采用备份的方式解决，通过共享存储同步数据，采用网络文件系统（NFS）或者Zookeeper的方式来支持共享存储。另外，MapReduce也已经添加了多平台支持，可以部署在图像处理单元（GPU）等高性能计算环境中。

（2）Dryad计算框架：Dryad是构建微软云计算基础设施的核心技术。编程模型相比于MapReduce更具一般性——用有向无环图（DAG）描述任务的执行，其中用户指定的程序是DAG图的节点，数据传输的通道是边，可通过文件、共享内存或者传输控制协议（TCP）通道来传递数据，任务相当于图的生成器，可以合成任何图，甚至在执行的过程中这些图也可以发生变化，以响应计算过程中发生的事件。图2给出了整个任务的处理流程。Dryad在容错方面支持良好，底层的数据存储支持数据备份；在任务调度方面，Dryad的适用性更广，不仅适用于云计算，在多核和多处理器以及异构集群上同样有良好的性能；在扩展性方面，可伸缩于各种规模的集群计算平台，从单机多核计算机到由多台计算机组成的集群，甚至拥有数千台计算机的数据中心。Microsoft借助Dryad，在大数据处理方面也形成了完整的软件栈，部署了分布式存储系统Cosmos[5]，提供DryadLINQ编程语言，使普通程序员可以轻易进行大规模的分布式计算。

（3）Spark计算框架：Spark是一种高效通用的分布式计算框架，采用基于DAG图的编程模型，提供了丰富



▲图2 Dyrad计算框架的任务处理流程

的编程接口。不同于MapReduce只能通过串联多个任务实现复杂应用，Spark可以在DAG图中划分不同的阶段，完成复杂应用的定义。在计算效率方面，Spark将结果以及重复使用的数据缓存在内存中，减少了磁盘IO带来的开销，更适用于机器学习等需要迭代计算的算法；在容错性方面，Spark表现突出，数据以弹性分布式数据集（RDD）[6]的形式存在，依靠Lineage的支持（记录RDD的演变），能够以操作本地集合的方式来操作分布式数据集。当RDD的部分分区数据丢失时，它可以通过Lineage获取足够的信息来重新运算和恢复丢失的数据分区。通过记录跟踪所有RDD的转换流程，可以保证Spark计算框架的容错性。资源管理及任务调度方面，Spark借助Mesos或者YARN来进行集群资源的管理，部署在集群中使用。Spark发展至今，已经形成了完整的软件栈，在Spark的上层，已经能够支持可在分布式内存中进行快速数据分析的Shark[7]、流计算Spark Streaming、机器学习算法库Mllib、面向图计算的GraphX等。

（4）GraphLab计算框架：图计算框架GraphLab的提出是为了解决大规模机器学习问题。相比于信息传递接口（MPI），GraphLab提供了更简单的编程接口，抽象的图模型使用户不必关注进程间的通信。相比于MapReduce计算框架，GraphLab更适合处理各数据之间依赖程度强、数据与数据之间需要频繁计算和信息交互的场景。GraphLab提出的图计算理论和方法不仅解决了集群中图处理的扩展问题，也解决了单机系统中大规模的图计算问题，可形成完整的面向机器学习的并行计算框架。但是，对于大规模自然图的处理，GraphLab仍然存在负载极不均衡、可扩展性差等缺点，因而GraphLab团队进一步提出了PowerGraph[8]。PowerGraph并行的核心思想是根据边的规模对顶点进行分割并部署在不同的机器上，由于不需要将同一个节点所对应的所有边的信息载入单机的内存中，因而消除了单机内存的约束。在系统的容错性方面，PowerGraph采用检查点技术，未来也考虑使用节点的副本冗余来提高容错性，能够在提高计算效率的同时完成快速恢复。

3 大数据流计算技术

大数据批量计算技术关注数据处理的吞吐量，而大数据流计算技术更关注数据处理的实时性，能够更加快速地为决策提供支持。大数据的流计算技术是由复杂事件处理（CEP）发展而来的，现在流计算的典型框架包括Storm、S4、Samza、Spark Streaming等。

（1）Storm计算框架：Storm提供了可靠的流数据处理，可以用于实时分析、在线机器学习、分布式远程过程调用（RPC），数据抽取、转换、加载（ETL）等。Storm运行用户自定义的拓扑，不同于MapReduce作业，用户拓扑永远运行，只要有数据进入就可以进行相应的处理。Storm采用主从架构，如图3所示，主节点中部署Nimbus，主要负责接收客户端提交的拓扑，进行资源管理和任务分配。从节点上运行监控器，负责从节点上工作进程也就是应用逻辑的运行。可扩展性方面，Storm借助于Zookeeper很好地解决了可扩展性的问题，集群非常容易进行横向扩展，便于统一的配置、管理和监控。容错性方面，使用ZeroMQ传送消息，消除了中间的排队过程，使得消息能够直接在任务之间流动，其注重容错和管理，实现了有保障的消息处理，保证每一个元组都会通过整个拓扑进行处理，未处理的元组，它会自动重放，再次进行。Storm的缺点是：集群存在负载不均衡的情况；任务部署不够灵活，不同的拓扑之间不能相互通信，结果不能共用。

（2）S4计算框架：S4是Yahoo发布的开源流计算平台，它是通用的、可扩展性良好、具有分区容错能力、支持插件的分布式流计算平台。S4采用分散对称的架构，没有中心节点，计算框架更加易于部署和维护。在计算过程中，每个计算节点都在本地内存中进行处理，避免了IO给计算带来的巨大瓶颈。S4的核心思想是将整个任务处理分为多个流事件，抽象成为一个DAG图，每个事件对应DAG图中的一条有向边，并用（K，A）的形式表示，其中K和A分别表示对应事件的键和属性。这种表示类似MapReduce中的键/价值设计，适合多个处理进行连接。S4的结构如图4所示，它采用Zookeeper来管理集群，提高了集群的可扩展性。在通信层，提供备用节点，如果有节点失败，处理框架会自动切换到备用节点，但是在内存中的数据会发生丢失。主从备份虽然在一定程度上提高了容错能力，但是相对较弱。同时通信层还使用一个插件式的架构来选择网络协议，通过TCP和用户数据报协议

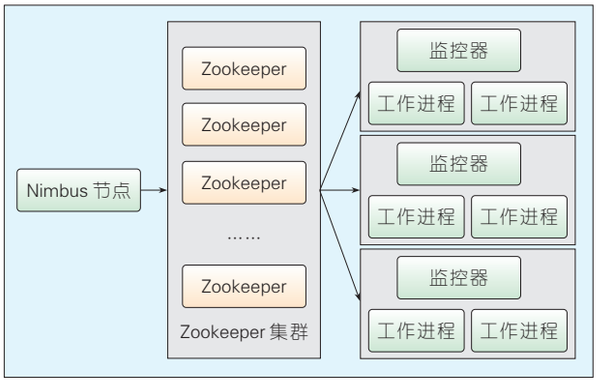
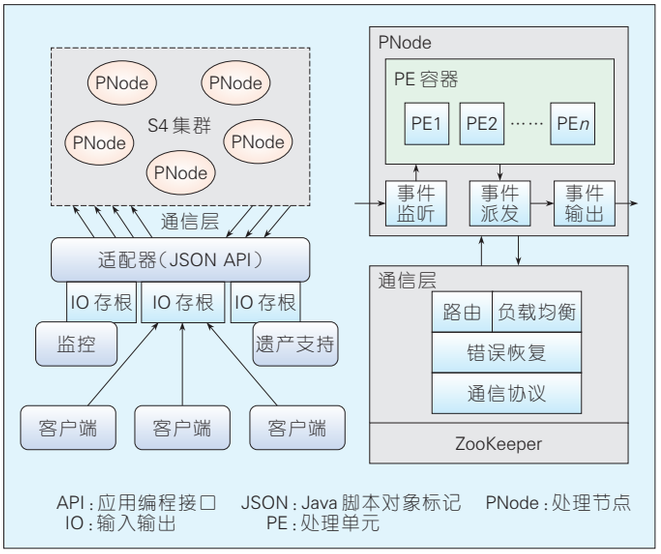


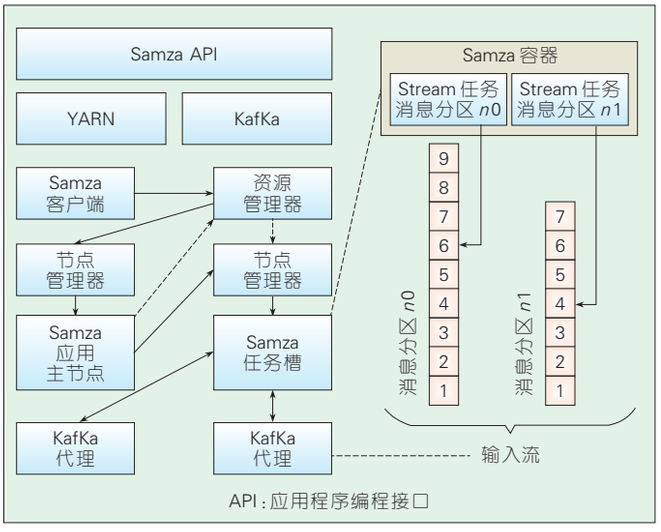
图3▶ Storm计算框架的架构



◀图4
S4 计算框架的架构

(UDP)之间的权衡来提高网络 IO 的速率。S4 框架的主要缺点是:持久化相对简单,数据存在丢失的风险;节点失败切换到备份节点之后,任务都需要重做;缺乏自动负载均衡的相关能力。

(3) Samza 计算框架:Samza 是 Linkedin 开源的分布式流处理框架,其架构如图 5 所示,由 Kafka[9]提供底层数据流,由 YARN 提供资源管理、任务分配等功能。图 5 也给出了 Samza 的作业处理流程,即 Samza 客户端负责将任务提交给 YARN 的资源管理器,后者分配相应的资源完成任务的执行。在每个容器中运行的流任务相对于 Kafka 是消息订阅者,负责拉取消息并执行相应的逻辑。在可扩展性方面,底层的 Kafka 通过 Zookeeper 实现了动态的集群水平扩展,可提供高吞吐、可水平扩展的消息队列,YARN 为 Samza 提供了分布式的环境和执行容器,因此也很容易扩展;在容错性方面,如果服务器出现故障,Samza 和 YARN 将一起进行任务的迁移、重启和重新执行,YARN 还能提供任务调度、执行状态监控等功能;在数据可靠性方面,Samza 按照 Kafka 中的消息分区进行处理,分区内保证消息有序,分区间并发执行,Kafka 将消息持久化到硬盘保证数据



◀图5
Samza 计算框架的架构及其作业处理流程

安全。另外,Samza 还提供了对流数据状态管理的支持。在需要记录历史数据的场景里,数据实时流动导致状态管理难以实现,为此,Samza 提供了一个内建的键/价值数据库用来存储历史数据。

(4) Spark Streaming 计算框架:Spark 是当前迭代式计算的典型代表,在前面的批量计算中已经介绍了 Spark 在大数据计算、数据抽象和数据恢复等方面的成果。如今 Spark 也在向实时计算领域发展,2013 年发表于顶级会议 SOSP 上的论文介绍了 Spark 在流计算中取得的最新成果。Spark Streaming 是建立在 Spark 上的应用框架,利用 Spark 的底层框架作为其执行基础,并在其上构建了 DStream 的行为抽象。利用 DStream 所提供的应用程序编程接口(API),用户可以在数据流上实时进行 count、join、aggregate 等操作。Spark Streaming 的原理是将流数据分成小的时间片断,以类似批量处理的方式来处理这小部分数据。DStream 同时也是 Spark Streaming 容错性的一个重要保障。

4 框架比较

随着数据的爆炸式增长,大数据计算平台在数据分析和处理中扮演着越来越重要的角色。本文分析了现有大数据处理面临的挑战和问题,详细分析了批处理和流处理相关计算框架的特点和重点解决的问题,结合存储、应用介绍了它们的核心创新点和软件栈,这些框架的总结和对比如表 1 所示。

5 结束语

应用推进了技术的发展和革新,目前业界在不断提高大数据计算框架的吞吐量、实时性、可扩展性等特性以应对日益增长的数据量和数据处理需求,大数据计算框架依然是现在以及未来一段时间内的研究热点。未来的发展趋势是:随着商业智

▼表1 典型大数据计算框架的对比

计算框架	计算效率（实时性）	容错性	特点	适用场景
MapReduce	低	任务出错重做	编程接口简单，计算模型受限	文本处理、log分析、机器学习
Spark	高	RDD的Lineage保证	内存计算，通用性好，更适合迭代式任务	迭代式离线分析任务、机器学习
Dryad	较高	任务出错重做	针对Join进行了优化，允许动态优化调度逻辑（修改DAG拓扑）	机器学习、微软技术栈
GraphLab	较高	检查点技术	机器学习图计算专用框架	机器学习、大图计算
Storm	高	Worker重启或分配到新机器，任务重做	通用性好，消息传递可靠，支持热部署，主节点可靠性差	通用的实时数据分析处理
S4	高	部分容错，检查点技术	通用性较好，通信在TCP和UDP之间权衡，持久化方式简单	实时广告推荐、容忍数据丢失
Samza	高	任务出错重做	可扩展性好，兼容流处理和批处理	在线和离线任务相结合的场景
Spark Streaming	低	RDD和预写日志（Write Ahead Logs）	通用性好，容错性好，通过设置短时间片实现实时，应用较为局限	历史数据和实时数据相结合的分析

DAG：有向无环图　RDD：弹性分布式数据集　TCP：传输控制协议　UDP：用户数据报协议

能和计算广告等领域的发展，更强调实时性的流计算框架将得到更加广泛的关注；能够缩短批量计算处理时间的Percolator[10]、Nectar[11]、Incoop[12]等增量计算模式将获得进一步的发展和应用；批量计算和流计算模式将进一步融合以减少框架维护开销。而实际上，现在的Spark计算框架除了支持离线批处理任务以外，已经能够通过Spark Streaming支持在线的实时分析。

除了计算框架本身的改进，消除磁盘IO和网络IO对计算效率的影响也是重要的研究方向之一。这方面，内存计算已经取得了不错的应用效果。例如，PACMan[13]用内存缓存输入数据，从而加速了MapReduce的执行；文中提到的Spark也是尽可能将所有的数据放在内存中，从而减少磁盘随机读写的IO开销，提高任务的执行效率。由于内存资源始终是宝贵的，难以满足大数据的存储需求，因此在很多情况下将仍然充当缓存的角色，需要进一步探究更为高效的缓存管理算法。另一方面，非易失性存储（NVM）器件的发展将对计算机系统的存储架构带来革命性的变化，如何充分利用新存储介质及其存储架构的特性提升计算效率也将是未来大数据计算框架研究的重要话题。

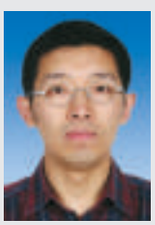
总之，应用的推动和技术的进步将会产生新的问题。作为大数据应用的核心，对于挖掘数据价值起着重要作用的计算框架将会面临更多的挑战，亟待解决。

参考文献

[1] 孟小峰, 慈祥. 大数据管理: 概念, 技术与挑战[J]. 计算机研究与发展, 2013, 50(1): 146–169
[2] ZAGARIA M, BORTHAKUR D, SARMA J S, et al. Job Scheduling for Multi–User MapReduce Clusters [R]. USA: EECS Department, University of California, 2009
[3] Hadoop.[EB/OL].[2013–08–24]. http://hadoop.apache.org/docs/r1.2.1/capacity_scheduler.html# Overview
[4] ZAHARIA M, KONWINSKI A, JOSEPH A D, et al. Improving MapReduce Performance in Heterogeneous Environments[C]// 8th USENIX Symposium on Operation Systems Design and Implementation(OSDI). USA: ASM, 2008：7
[5] CHAIKEN R, JENKINS B, LARSON PÅ, et al. SCOPE: Easy and Efficient Parallel Processing of Massive Data Sets [J]. Proceedings of the VLDB Endowment, 2008, 1(2): 1265–1276
[6] ZAHARIA M, CHOWDHURY M, DAS T, et al. Resilient Distributed Datasets: A Fault–Tolerant Abstraction for in–Memory Cluster Computing[C] //Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. USA: USENIX Association, 2012：2–2
[7] XIN R S, ROSEN J, ZAHARIA M, et al. Shark: SQL and Rich Analytics at Scale[C]// Proceedings of the 2013 ACM SIGMOD International Conference on Management of data. USA: ACM Press,2013: 13–24
[8] GONZALEZ J E, LOW Y, GU H, et al. PowerGraph: Distributed Graph–Parallel Computation on Natural Graphs[C]// Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI). USA: USENIX Association, 2012: 17–30
[9] KREPS J, NARKHEDE N, RAO J. Kafka: A Distributed Messaging System for Log Processing[C]//Proceedings of the 6th International Workshop on Networking Meets Databases (NetDB). USA: ACM Press, 2011
[10] PENG D, DABEK F. Large–Scale Incremental Processing Using Distributed Transactions and Notifications[C]// Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI). USA: USENIX Association, 2010: 1–15
[11] GUNDA P K, RAVINDRANATH L, THEKKATH C A, et al. Nectar: Automatic Management of Data and Computation in Datacenters[C]// Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI). USA: USENIX Association, 2010:75–88
[12] BHTOTIA P, WIDER A, RODRIGUES R, et al. Incoop: MapReduce for incremental computations[C]//Proceedings of the 2nd ACM Symposium on Cloud Computing. USA: ACM, 2011：7
[13] ANANTHANARAYANAN G, GHODSI A, WANG A, et al. PACMan: Coordinated Memory Caching for Parallel Jobs[C]//9th USENIX Symposium on Networked Systems Design and Implementation(NSDI). USA: USENIX Association, 2012

作者简介

赵晟，清华大学计算机科学与技术系硕士研究生；研究方向为分布式存储与计算。



姜进磊，清华大学计算机科学与技术系副教授；研究方向为分布式计算与系统、云计算、大数据和虚拟化等；先后主持和参加国家自然科学基金、“863”计划、“973”计划等项目10余项；获国家技术发明奖二等奖1项；已发表论文50多篇，其中被SCI/EI检索40余篇。