# MapReduce 并行计算技术发展综述①

应 毅[1], 刘亚军[1,2]

[1](三江学院 计算机科学与工程学院, 南京 210012)
[2](东南大学 计算机科学与工程学院, 南京 210096)

**摘 要**: 经过几年的发展, 并行编程模型MapReduce产生了若干个改进框架, 它们都是针对传统MapReduce的不足进行的修正或重写. 本文阐述和分析了这些研究成果, 包括: 以 HaLoop 为代表的迭代计算框架、以 Twitter Storm 为代表的实时计算框架、以 Apache Hama 为代表的图计算框架以及以 Apache YARN 为代表的框架管理平台. 这些专用系统在大数据领域发挥着越来越重要的作用.

**关键词**: MapReduce; Hadoop; 并行计算; 大数据处理

## Survey of Developments of MapReduce Parallel Computing Technology

YING Yi[1], LIU Ya-Jun[1,2]

[1](College of Computer Science and Technology, Sanjiang University, Nanjing 210012, China)
[2](School of Computer Science and Engineering, Southeast University, Nanjing 210096, China)

**Abstract**: With the rapid development of recent years, some improved framework of MapReduce parallel programming model appeared. They are correction and recoding against lack of MRv1. This paper describes and analyzes this research achievements, including iterative computing framework as represented by HaLoop, real-time computing framework as represented by Twitter Storm, graph computing framework as represented by Apache Hama, computing resources negotiation platform as represented by Apache YARN. These special systems play a vital role in BigData fields.

**Key words**: MapReduce; Hadoop; parallel computing; big data processing

2006 年 8 月, Google 在搜索引擎大会上首次提出“Cloud Computing”的概念. 经过几年的高速发展, 工业界已经实现了多个云计算实例[1], 例如：Amazon 的 EC2、微软的 Azure、IBM 的“蓝云” 、阿里巴巴的“飞天”. 其中以Google的云计算平台最为著名. 从技术层面上看, Google 的云计算基础架构包括 3 个相互独立又紧密结合的系统: 分布式文件系统 GFS[2]; 并行编程模型和任务调度模型 MapReduce[3]; 能够处理结构化以及半结构化数据的大规模分布式数据库 Bigtable[4].

Apache Hadoop[5]是一种基于批处理技术的开源云计算平台[6], 由 Java 语言编写, 运行在 Linux 操作系统之上, 由 HDFS[7](Hadoop Distribute File System)和 Hadoop MapReduce 两个核心部件组成, 二者分别是 GFS 和 GMR(Google MapReduce)的开源实现. 短短几年间, Hadoop 已经成为大数据领域事实上的标准, 有自己完备的生态系统, 图 1 显示了 Hadoop 系统中的最主要产品, 但在根本上, Hadoop 的核心仍是 MapReduce.

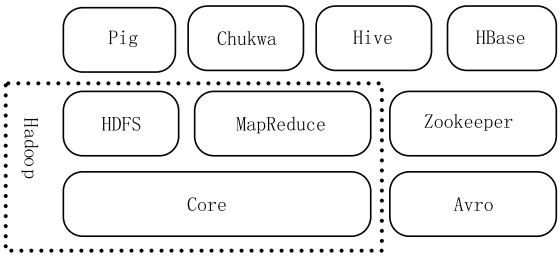


图 1 Hadoop 生态圈中的主要产品

① 基金项目:江苏省卓越工程师(软件类)计划试点专业(苏教高函[2012]17 号)
收稿时间:2013-09-06;收到修改稿时间:2013-10-12

本文首先介绍了并行编程框架 MapReduce, 同时简述了它的不足, 之后给出了 MapReduce 的几个改进框架, 它们都是近期业界著名的研究成果, 包括: 迭代计算框架、实时计算框架、图计算框架及框架管理平台. 文章最后进行了总结.

## 1 MapReduce

根据数据的来源, 大数据可以粗略地分成两大类: 一类来自物理世界, 另一类来自人类社会. 前者多半是科学实验数据或传感数据, 后者与人的活动有关系, 特别是与互联网有关[8]. 据“Digital Universe”项目统计 2010 年、2011 年全球新增数据总量分别为 1.2ZB 和 1.8ZB, 2012 年数据总量高达 2.8ZB[8,9]. 作为处理 BigData 的有力工具, 分布式系统和并行计算技术日趋重要.

MapReduce 是 Google 于 2004 年提出的能并发处理海量数据的并行编程模型[10,11], 同时也是一种高效的任务调度模型. MapReduce 的最大优势在于屏蔽底层实现细节, 有效降低并行编程难度, 提高编程效率. 开发人员只需将精力放在应用程序本身, 而关于集群的处理问题, 比如数据分块、资源分配调度、负载平衡、容错处理、节点通信等则交由平台来处理.

“Map” 、“Reduce”的概念和主要思想, 都是从函数式编程语言和矢量编程语言借鉴而来的. Map 函数负责分块数据的处理, Reduce 函数负责对分块数据处理的中间结果进行归约. MapReduce 其实就是 Divide/Conquer 的过程, 通过把问题 Divide, 使这些 Divide 后的 Map 运算高度并行, 再将 Map 后的结果进行 Reduce, 得到最终的结果. 如图 2 所示.

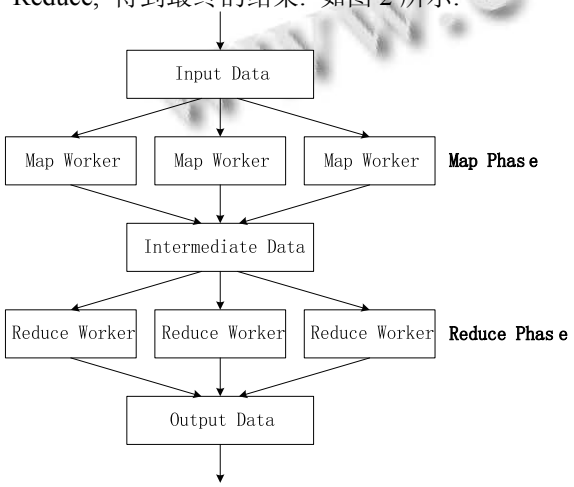


图 2 MapReduce 的工作原理

正是由于 MapReduce 有函数式和矢量编程语言的共性, 使得这种编程模式特别适合于海量数据的搜索、挖掘、分析和机器智能学习等. MapReduce 可以处理 TB 和 PB 量级的数据, 并在处理 TB 级别以上海量数据的业务上有着明显的优势.

MapReduce 也不是万能的, 它的不足主要有 3 点: ①MapReduce 主要针对松耦合型的数据处理应用, 对于不容易分解成众多相互独立子任务的紧耦合型计算任务, 处理效率很低; ②MapReduce 不能显式的支持迭代计算; ③MapReduce 是一种离线计算框架, 不适合进行流式计算和实时分析.

## 2 技术研究进展

针对第一代 MapReduce(MRv1)的缺点, 近期工业界和科研学术界提出了若干个 MRv1 的改进产品, 大体上可以分为: 迭代计算框架、实时计算框架、图计算框架和框架管理平台.

### 2.1 迭代计算框架

为了系统的健壮性, Hadoop 使用磁盘存储 MapReduce 计算的中间数据, 所以它不能显式的支持迭代编程. 但是很多大数据处理需要多次循环, 需求驱动之下, 产生了 HaLoop[12].

在 Hadoop 中, 迭代控制是在 Job 间的, 一个 Job 只能是一个 Map-Reduce 对, 迭代计算需要进行多次 Map-Reduce, 势必需要启动多个 Job, 并且在每个 Job 完成之后要进行迭代结束的判断, 判断时需要从 HDFS 中读取 Reduce 的结果, 又要启动额外的 Job. 磁盘 I/O 和反复启动 Job 严重降低了 Hadoop 的性能.

HaLoop 仍然使用 Hadoop 的底层文件系统(HDFS)和任务排队策略(Task Queue), 但修改了 Master 节点的 JobTracker, 将每一个 Job 中的 1 个 Map-Reduce 对改成多个 Map-Reduce 对, 使得 Job 可以复用, 并增加了迭代控制功能(Loop Control), 使迭代控制在 Job 内部完成, 无需启动新的 Job 来计算. 迭代结束有两种判断方法: ①设置相邻两次迭代计算的差值门限(setFixedPointThreshold 接口); ②设置迭代的最大次数(setMaxNumOfIterations 接口).

HaLoop 用以下递归公式描述迭代计算. 其中 $R_0$ 是初始输入, L 是每次迭代不变的数据, $R_i$ 是第 i 次迭代产生的结果.

$$R_i+1 = R_0 \cup ( R_i \ \infty L )$$

迭代计算具有 Loop-Invariant Data 比 Loop-Variant Data 多得多的特性. HaLoop 修改了 Slave 节点的 Task Tracker, 将 L 数据进行缓存(Caching)和索引(Indexing), 每次迭代重用这些数据, 减少 I/O, 加快处理速度, 提高迭代效率.

为了充分利用 Slave 上的 Cache, HaLoop 使用 Loop-aware 任务调度机制修正了 Task Scheduler. 首次迭代时会将 Loop-Invariant Data 保存到 Slave 节点上, 以后每次调度 Task, 尽量放在这个节点上, 优点在于迭代过程中, 数据不必重复传输.

HaLoop 整体架构如图 3 所示.

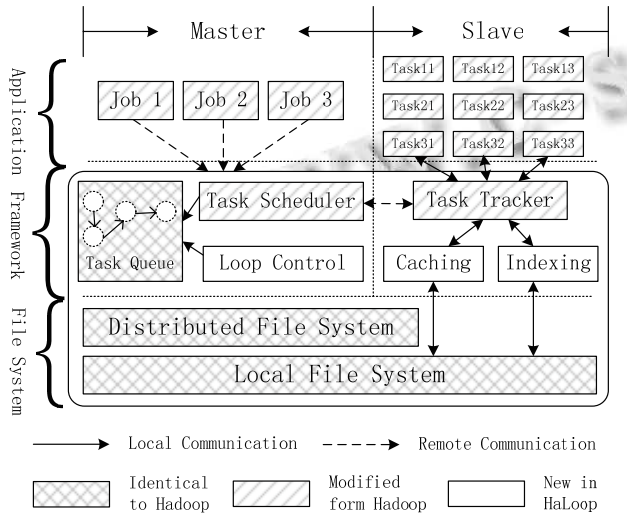


图 3 HaLoop 整体框架

其它开源的迭代计算框架还有: Twister[13]和 UCBerkeley Spark[14, 15].

Twister 系统对 Hadoop 进行了较大的改动, 被处理的数据全部驻留内存, 通过缓存数据和拓展 MapReduce API, 实现迭代过程中的数据重用和分发管理, 采用第 3 方消息通信管理插件来完成通信控制, 使用任务池来避免多次作业调度, 目前支持 NaradaBroking、ActiveMQ 等基于“发布——订阅”架构的通信插件[16].

### 2.2 实时计算框架

绝大多数分布式商用系统需要实时处理数据, MRv1 无法胜任, Google 在 2010 年公布的新技术 Dremel[17]即为一个海量数据实时查询系统, Apache Drill[18]是它的开源实现. Storm[19]是 BackType 开发的实时分布式计算系统, 后被 Twitter 开源.

Storm 集群主要有两类节点: 主节点上运行的 Nimbus 守护进程; 每一个工作节点上运行 Supervisor 守护进程. 图 4 描述了 Storm 集群的大致组网情况. 在 Storm 上运行一次应用被称为一个 Topology, 一个 Topology 会被分为多个子 Topology 由多个 Supervisor 协同完成. Topology 类似于 Hadoop 中的 Job, 不过在显式 Kill 之前, Topology 不会停止, 它会持续处理到达的流数据. Nimbus 负责分配代码、布置任务及故障检测, Supervisor 负责监听所在机器, 根据 Nimbus 的委派在必要时启动和关闭 Worker 进程. 每个 Worker 进程执行一个 Topology 的子集, 一个运行中的 Topology 由许多跨多台机器的 Worker 进程组成(图 3 中所示的 task).

Nimbus 和 Supervisor 之间的协调工作通过 ZooKeeper 完成. Nimbus 和 Supervisor 是无状态的, 所有状态维持在 ZooKeeper 或本地磁盘. 当 Nimbus/Supervisor 进程异常重启后, 将会很快恢复状态并继续工作, 这种设计使 Storm 极其稳定. 借助 ZooKeeper, 也降低了主节点和工作节点之间的相互依赖.

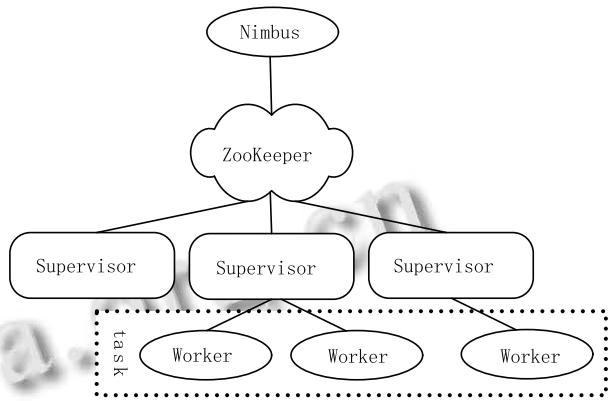


图 4 Storm 集群

其它实时计算框架包括 Stinger Initiative[20]、Yahoo!开源的 Apache S4[21]、Cloudera Impala[22]和 LinkedIn 开源的 Apache Samza[23].

### 2.3 图计算框架

BSP(Bulk Synchronous Parallel model)模型[24,25]是 2010 年图灵奖得主 Leslie Valiant 在 1990 年提出的一种基于消息通信的并行执行模型. Google 的 Pregel[26]、Yahoo!开发并开源的 Giraph[27]以及 Apache Hama[28], 都是基于 BSP 模型开发的. BSP 模型不仅是一种体系结构, 也是一种并行程序设计方法. 其设计准则是 Bulk 同步, 关键在于超步(Super Step)的概念. 一个

BSP程序由一系列串行的超步组成, 在一个超步中, 所有的进程并行执行局部计算, 一个超步可分为三个阶段: ①本地计算阶段(每个处理器只对存储本地内存中的数据进行计算); ②进程通信阶段(对非本地数据进行操作); ③Barrier 同步阶段(等待所有通信行为结束).

Google Pregel 是一个用于解决大规模图数据问题的分布式图计算框架, 能够完成复杂的图处理任务, 例如: 图遍历(BFS)、最短路径(SSSP)、PageRank[29]计算、社交网络分析和图挖掘等. Pregel 采用简单图模型来组织存储和处理大规模图数据, 使用异步通信方式和消息合并机制来减少网络通信量和存储量, 提供显式的同步控制框架和“检查点加消息记录”的容错管理方案, 是目前较为完善的专门针对大规模图处理应用的系统.

Yahoo! Giraph 是在 Hadoop 平台上运行的一个大规模图算法库, 在原有 MapReduce 的基础上, 只启动 Map 任务, 在 Map 任务中嵌套 BSP 模型(参考 Pregel 的设计), 实现多次循环迭代, 以支持大规模图处理应用.

Apache Hama 是 Google Pregel 的模仿, 一个建立在 Hadoop 上的分布式并行处理系统, 适合需要多次迭代的图处理. Hama 和 Giraph 均采用基于 HTTP 协议的 RPC 通信机制. 与 Pregel 类似, Hama 也提供显式的同步控制框架. Hama 集群环境由计算引擎 BSPMaster/GroomServer、分布式锁服务 ZooKeeper、存储系统 HDFS/HBase 三部分组成. 图 5 主要概述了 Hama 的整体系统架构, 并描述了系统模块之间的通讯与交互. HDFS/HBase 负责持久化存储数据, BSPMaster 负责进行对 GroomServer 进行任务调配, GroomServer 负责进行对 BSPPeer 进行程序调用, ZooKeeper 负责对 GroomServer 进行失效转发.

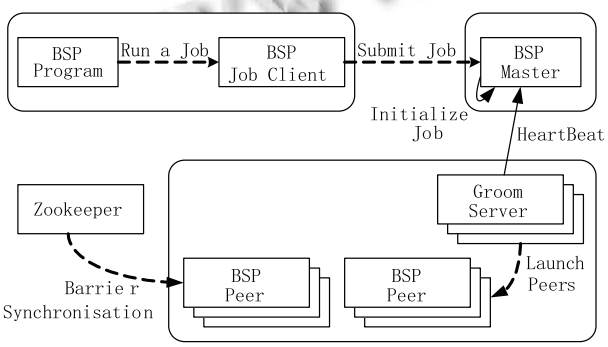


图 5　Apache Hama 系统架构

一个 GroomServer 包含一个 BSPPeer 实例, 二者运行在同一个物理节点上, GroomServer 通过向 BSPMaster 发送“心跳”汇报自身状态(包括本节点处理能力、可用内存等信息). BSPMaster 控制集群环境中的超步: BSPMaster 在每一次操作(input、output、computation、saving 等)终止时, 向所有计算节点发送相同的指令, 然后等待每一个计算节点的回应心跳, 之后进行下一阶段的任务分配. BSPMaster 还具有状态维护、计算任务分割、任务调度、失效转发等功能.

2.4 MRv2

以 GMR 和 HMR(Hadoop MapReduce)为代表的传统 MapReduce 是一种离线计算框架, 类似于 HPC(High Performance Computing)中的批处理技术, 适合数据密集型计算, 对迭代计算、流式计算等计算框架不能很好的支持. 为此, Apache 启动了 YARN[30](Apache Hadoop NextGen MapReduce, MRv2)项目的开发. 与 MRv1 相比, YARN 不再是一个单纯的离线计算环境, 而是一个通用的计算框架管理平台(YARN 是“Yet Another Resource Negotiator”的缩写), 不仅支持 MapReduce, 也支持其它流行的计算框架, 如: 迭代计算、流式计算、内存计算、在线计算、实时处理、MPI 等. 使用者可以将各种计算框架移植到 YARN 上, 由 YARN 进行资源的统一管理和分配, 使各个框架共享一个集群, 大大降低运维成本和硬件成本.

Apache YARN 由 Resource Manager(RM)和 Node Manager(NM)组成. 相较于 MRv1, 其最大特点是将 JobTracker 的两个主要功能“作业调度/监控”和“资源管理”进行分拆, 总体上采用双层调度架构, 如图 6 所示.

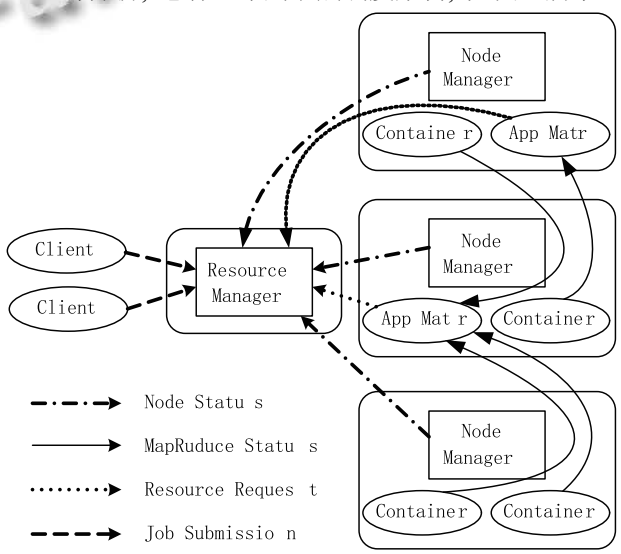


图 6　Apache YARN 整体架构

RM 由 Scheduler 和 Applications Manager(ASM)组成. Scheduler 和 NM 协调完成资源管理的功能, NM 负责单个节点的资源管理和监控, 定期将资源使用情况汇报给 RM. 运行在 YARN 上的任何一个作业(如 Hadoop Job、Spark Job)都是一个 Application Master(AM), Scheduler 根据容量、队列等限制条件, 将系统中的资源分配给各个正在运行的 AM. YARN 自带了多个资源调度器, 如 FIFO、Capacity Scheduler、Fair Scheduler 等.

YARN 对资源进行抽象, 称之为 Container, 它将某个节点上的 CPU、内存、磁盘等资源封装在一起. AM 只有获得至少一个 Container 后才能启动任务, AM 本身也必须运行在一个 Container 中. ASM 负责管理系统中所有应用程序的 AM; 获取第一个 Container 用于启动 AM; 监控 AM 的运行状态; 在 AM 失败时对其重启. 每个应用程序均会有一个 AM, 与 NM 合作, 在合适的容器中运行对应的 Task, 并监控这些 Task 执行.

与 YARN 相似的开源项目还有 Apache Mesos[31] 和 Facebook 开发后贡献给 Hadoop 的 Corona[32].

## 3 各模型特点及具体应用

GMR 的提出最早只是为了解决与搜索相关的问题, 但随着研究的不断深入, 由上节所述可以看出, 对 MRv1 进行了多个方向的改进, 它们相互影响有各具特点. 表 1 显示了 MRv1 的改进分类及特点.

表 1 基于 MRv1 的改进模型对比

| 分 类 | 名 称 | 特 点 |
|---|---|---|
| 迭代计算框架 | HaLoop Twister UCBerkeley Spark | 循环控制、数据缓存、减少磁盘 I/O |
| 实时计算框架 | Apache Drill Cloudera Impala Stinger Initiative Apache S4 Twitter Storm Apache Samza | 保证响应时间的事务功能、消息精确处理、动态流数据处理、记录级容错 |
| 图计算框架 | Yahoo! Giraph Apache Hama | 以 BSP 模型为理论基础、占用较低资源的消息通信机制、同步控制框架 |
| 计算框架管理平台 | Apache YARN Apache Mesos Hadoop Corona | 通用计算资源管理与分配, 不局限于特定计算框架 |

大多数 MRv1 的改进模型都起源于某个特殊的应用领域, 表 2 对几个具有代表性的模型应用进行了比较. 通过对比不难看出, 迭代计算框架和实时计算框架主要解决分布式商用计算中的在线服务和实时处理问题(如视频优化、分布式 RPC、日志分析); 图计算框架仍旧主要针对互联网应用中的基于大数据的后台 Batch 计算(如 PageRank、排序); 计算框架管理平台主要对各种计算模型进行资源管理(如管理 MapReduce、Storm、Spark、Giraph 等).

各模型也由于各自的开发成熟程度不同, 在具体使用上有较大差别. 迭代计算框架 UCBerkeley Spark、实时计算框架 Twitter Storm、图计算框架 Yahoo! Giraph、计算框架管理平台 Apache Mesos 和 Apache YARN 由于发展时间较长、技术实力雄厚在业界应用较为广泛.

表 2 典型模型具体应用对比

| 模型名称 | 应用领域 | 应用企业 |
|---|---|---|
| UCBerkeley Spark | 机器学习、数据挖掘、实时视频优化、可视化模式分析 | UC Berkeley AMPLab、百度、Yahoo!、Sohu、淘宝、Adatao、Freeman Lab、Conviva 等 |
| Twitter Storm | 实时分析、在线机器学习、连续计算、分布式 RPC、信息流处理、ETL(数据抽取、转换和加载) | Twitter、Yahoo!、百度、淘宝、支付宝、Groupon、乐元素、Admaster 等 |
| Apache Samza | 服务器日志分析、示踪数据处理、流数据实时摄取 | LinkedIn |
| Yahoo! Giraph | PageRank、图索引 | Yahoo! TomTom Maps |
| Apache Hama | 矩阵计算、面向图计算、PageRank、排序计算、BFS | NHNCorp |
| Apache YARN | Storm 管理、Spark 管理、HBase 管理、Giraph 管理、Kafka 管理、BSP 管理、Tez(DAG 计算框架)管理 | Yahoo!、Samza、Hortonworks |
| Apache Mesos | Spark 集群管理 | UC Berkeley AMPLab、Twitter、Airbnb、CloudPhysics、Conviva、UCSF、Categorize 等 |

## 4 结语

作为 HPC 在工业界的演进结果, MapReduce 计算技术在大数据领域发挥着越来越重要的作用. 在商业需求驱动之下, 其并行计算基因必定将与分布式商用计算紧密结合, 在不同的应用领域, 产生出更具有针对性的专用模型. 毫无疑问, 这些专用模型的研究是一个充满前途和挑战的领域.

## 参考文献


1 陈康,郑纬民.云计算:系统实例与研究现状.软件学报,2009,20(5):1337−1348.

2 Ghemawat S, Gobioff H, Leung ST. The Google file system. ACM SIGOPS Operating Systems Review. ACM. 2003, 37(5): 29–43.

3 Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. Communications of the ACM, 2008, 51(1): 107–113.

4 Chang F, Dean J, Ghemawat S, et a1. Bigtable: A distributed storage system for structured data. Proc. of the 7th USENIX Symp. on Operating Systems Design and Implementation. 2006. 205–218.

5 Tom White 著.周敏奇,王晓玲,金澈清等译.Hadoop 权威指南(第二版).北京:清华大学出版社,2011.

6 周洪波.云计算:技术、应用、标准和商业模式.北京:电子工业出版社,2011.

7 Shvachko K, Kuang H, Radia S, et al. The Hadoop distributed file system. Mass Storage Systems and Technologies (MSST). 2010 IEEE 26th Symposium on. IEEE. 2010. 1–10.

8 李国杰.大数据研究的科学价值.中国计算机学会通讯,2012,8(9):8-15.

9 IDC 发布最新《数字宇宙研究报告》. http://www.ecas.cn/xxkw/kbcd/201115_93655/ml/xxhjsyjcss/201212/t20121229_3730152.html.

10 李建江,崔健,王聃,等.MapReduce 并行编程模型研究综述.电子学报,2011,(11):2635−2642.

11 幸莉仙,黄慧连. MapReduce 框架下的朴素贝叶斯算法并行化研究.计算机系统应用,2013,22(2):108−111.

12 Bu Y, Howe B, Balazinska M, et al. HaLoop: Efficient iterative data processing on large clusters. Proc. of the VLDB Endowment, 2010, 3(1-2): 285–296.

13 Ekanayake J, Li H, Zhang B, et al. Twister: A runtime for iterative mapreduce. Proc. of the 19th ACM International Symposium on High Performance Distributed Computing. ACM. 2010. 810–818.

14 Zaharia M, Chowdhury M, Das T, et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. Proc. of the 9th USENIX Conference on Networked Systems Design and Implementation. USENIX Association. 2012. 2–2.

15 Zaharia M, Chowdhury M, Franklin M J, et al. Spark: cluster computing with working sets. Proc. of the 2nd USENIX Conference on Hot topics in Cloud Computing. 2010. 10–10.

16 于戈,谷峪,鲍玉斌,等.云计算环境下的大规模图数据处理技术.计算机学报,2011,(10):1753−1767.

17 Melnik S, Gubarev A, Long JJ, et al. Dremel: interactive analysis of web-scale datasets. Proc. of the VLDB Endowment, 2010, 3(1-2): 330–339.

18 Drill Home Page. http://incubator.apache.org/drill/.

19 Storm Home Page. http://storm-project.net/.

20 Stinger Initiative Home Page. http://hortonworks.com/stinger/.

21 Neumeyer L, Robbins B, Nair A, et al. S4: Distributed stream computing platform. Data Mining Workshops (ICDMW). 2010 IEEE International Conference on. IEEE. 2010. 170–177.

22 Cloudera Impala Home Page. https://github.com/cloudera/impala.

23 Samza Home Page. http://samza.incubator.apache.org/.

24 Valiant LG. A bridging model for parallel computation. communications of the ACM, 1990, 33(8): 103−111.

25 Pace MF. BSP vs MapReduce. Procedia Computer Science, 2012, 9: 246–255.

26 Malewicz G, Austern MH, Bik AJC, et al. Pregel: a system for large-scale graph processing. Proc. of the 2010 international conference on Management of data. ACM. 2010. 135−146.

27 Giraph Home Page. http://giraph.apache.org/.

28 Hama Home Page. http://hama.apache.org/.


(下转第 11 页)

批对象应不一样. 此外有些节点能修改项目立项信息, 而有些节点只有查看权限, 如填报人和依托项目负责人有编辑权限, 而其他审批节点的审批人则只有查看权限.

(5) 审批意见填写与否的可配置性. 在每个审批节点, 审批人在审批通过或审批拒绝时是否应该强制填写处理意见, 须通过配置实现. 默认情况下, 拒绝时必须填写处理意见. 同意时则可以不填写处理意见.

(6) 流程变更的适应性. 为了增强系统的稳定性, 应对流程定义中增加、删除节点时对已审批和审批中的项目的影响, 流程应由流程实例和流程定义“拼凑”起来, 走过的节点将不受流程变更的影响.

(7) 流程的可见性. 为了提高系统的易用性, 让审批人能查看整个流程进度和流程所走的历史轨迹, 在审批人的审批界面将显示整个流程信息, 包括流程的历史轨迹和后续尚未走完的流程信息.

以上需求使得流程实现时必须考虑到流程的可配置性、流程变更的适应性、流程的可见性. 基于这三大原则后实现的项目立项审批流程如图 7. 审批人在


图 7 项目立项审批实现效果(审批流程部分)

前台界面可方便地查阅流程进度和流程历史轨迹, 其中项目填报人在第一次提交流程后, 依托项目审批人以“预算金额不对”退回. 项目填报人修改预算金额后再次提交通过, 直至流转到科研处项目室, 此节点具有跳转、终止流程的权限, 即审批人可以选择审批的下一节点. 每次处理系统都将记录审批人的处理时间, 以便用户查阅、跟踪流程进度.

## 4 总结

本文描述了一个柔性审批流程模型的设计和实现过程. 该审批模型具有流程可配置性、流程变更的适应性和流程可见性的特点, 它能够弥补传统的审批流程灵活性的不足, 有效地支持人工调整流程, 在实现电子审批的同时, 充分发挥了审批人的主观能动性. 该自定义审批模型能应对各种“异常”的审批流程, 这些“异常”情况, 需要通过人机交互来动态决定审批流程的实际路径[8]. 该审批模型已经应用于某机构科研管理信息系统中, 取得了不错的应用效果.

## 参考文献

1 张霞,陆剑江.面向角色的动态审批流程的研究.计算机应用与软件,2011,28(1):193–195.

2 胡奇.jBPM4 工作流应用开发指南.北京:电子工业出版社,2010,306.

3 王正方,南琳,王作鹏,孙兆华.企业信息系统中的业务信息审批模型.计算机工程,2008,34(9):257–259.

4 刘晓冰,吕强,薛冬娟,邱立鹏.基于特征映射的自定义审批系统的设计与实现.计算机集成制造系统,2008,14(5):970–976.

5 周建涛,史美林,叶新铭.柔性工作流技术研究的现状与趋势.计算机集成制造系统,2005,11(11):1501–1510.

6 韩月娟,赵雷,吕强,杨季文.一个基于 SOFM 的柔性化审批应用方案的设计.计算机工程,2006, 32(21):275–276

7 蒋建彬.基于 SOA 的审批流程管理系统的研究和实现[学位论文].成都:电子科技大学,2010.

8 何青.一个基于 TBAC 的审批业务工作流模型.山东大学学报(工学版),2006,36(4):121–124.

(上接第 6 页)

29 Brin S, Page L. The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems, 1998, 30(1): 107–117.

30 Vavilapalli VK, Murthy AC, Douglas C, et al. Apache hadoop YARN: Yet another resource negotiator. Proc. of the Fourth ACM Symposium on Cloud Computing. ACM, 2013.

31 Hindman B, Konwinski A, Zaharia M, et al. Mesos: A platform for fine-grained resource sharing in the Data Center. Proc. of the 8th USENIX Conference on Networked Systems Design and Implementation. USENIX Association. 2011. 22–22.

32 Corona Home Page. https://github.com/facebook/hadoop-20/tree/master/src/contrib/corona/.