火龙果•整理
uml.org.cn

硕士学位论文

基于 Hadoop 的分布式网络爬虫技术

DISTRIBUTED WEB CRAWLER TECHNOLOGY BASED ON HADOOP

郑博文

哈尔滨工业大学
2011 年 6 月

火龙果•整理
uml.org.cn

国内图书分类号：TP391.2　　学校代码：10213
国际图书分类号：681.37　　密级：公开

工学硕士学位论文

基于 Hadoop 的分布式网络爬虫技术

硕 士 研 究 生：郑博文
导　　师：赵铁军 教授
申 请 学 位：工学硕士
学　　科：计算机科学与技术
所 在 单 位：计算机科学与技术学院
答 辩 日 期：2011 年 6 月
授予学位单位：哈尔滨工业大学

火龙果•整理
uml.org.cn

Classified Index: TP391.2

U.D.C.: 681.37

Dissertation for the Master Degree in Engineering

DISTRIBUTED WEB CRAWLER TECHNOLOGY BASED ON HADOOP

Candidate: Zheng Bowen

Supervisor: Prof. Zhao Tiejun

Academic Degree Applied for: Master of Engineering

Speciality: Computer Science and Technology

Affiliation: School of Computer Science and Technology

Date of Defence: June, 2011

Degree-Conferring-Institution: Harbin Institute of Technology

火龙果•整理
uml.org.cn

摘　要

如今我们正生活在一个信息爆炸的年代，随着互联网行业迅猛发展，这些信息每年以指数型增长，同时对于随时随地获取信息的需求也与日俱增，这些需求驱动了云计算的发展。在这个大背景之下，Google、IBM、Apache 和 Amazon 等大型公司纷纷投入大量财力去发展云计算。其中 Apache 开发的 Hadoop 平台是一个对用户极为友好的开源云计算框架。本文所开发的分布式爬虫系统即是在此框架下设计和实现的。

本文的目的设计并实现一个基于 Hadoop 的分布式爬虫系统，完成大规模数据采集的任务。同时，该爬虫系统采集信息类型为 27 种语言的主流新闻网站。该爬虫的采集方式为全站式信息采集，即抓取 27 种语言种子对应网站上的全部信息。另外，27 种语言信息还要分别保存便于后面跨语言处理。

本文全部工作中研究部分包括云计算相关知识介绍、Hadoop 分布式平台相关知识介绍、网络爬虫原理和分布式爬虫发展现状调研。首先，对云计算的定义、原理和体系结构进行调研。然后，深入研究 Hadoop 平台的分布式文件系统(HDFS)和分布式计算模型(Map/Reduce)。接着讲述爬虫系统的原理，了解开发一个爬虫需要的流程。最后调研目前分布式爬虫系统的发展现状。

上面这些研究为本文提供了技术基础，本文在此基础上提出了基于 Hadoop 的分布式网络爬虫系统的设计方案，包括爬虫系统的基本流程设计、框架设计、功能模块划分和各模块的 Map/Reduce 设计。在概要设计的基础之上，本文做出了系统的详细设计，实现整个系统，包括数据存储结构的实现、爬虫总体数据结构和各个功能模块的实现。最后，对本文做出详细总结。

本文的意义在于实现了一个基于 Hadoop 的分布式爬虫系统，该系统采用 Map/Reduce 计算框架符合整个项目分布式框架。解决了单机爬虫效率低、可扩展性差等问题，提高了信息采集速度并扩大了信息采集的规模。为分布式跨语言信息获取和检索平台的索引模块和信息处理模块提供数据。

关键词： 分布式爬虫；Hadoop；HDFS；Map/Reduce

火龙果•整理
uml.org.cn

Abstract

Today we are living in a era of information explosion, with the rapid development of the Internet industry, this information is growing exponentially every year, for anytime, the demand for access to information is also increasing, these requirements drive the development of cloud computing. On this background, Google, IBM, Apache, and Amazon and other large companies have invested substantial financial resources to the development of the cloud. Apache Hadoop development platform is a very user-friendly open source cloud computing framework. The distributed crawler system developed in this paper that is in this framework, design and implementation.

The purpose of this paper is to design and implement a crawler system based on Hadoop distributed to complete the task of large-scale data collection. Meanwhile, the crawler system collects information for the mainstream news sites in 27 types of languages. The way of collecting information for crawler system is all website-based collection. In addition, the information in 27 languages was also saved separately for cross-language processing.

All of the work of research in this paper includes relevant knowledge described cloud computing, Hadoop distributed platform knowledge, principles of Web crawler and survey about development of a distributed crawler. First, research the definition of cloud computing, principles and architecture. Then, in-depth study Hadoop Distributed File System (HDFS) and the distributed computing model (Map / Reduce). Then the article described the principles of crawler system to understand the process of developing a crawler system. Finally, research the development of the current status of the distributed crawler system.

These studies above provided the technical foundation for this article, this paper puts forward a design for distributed web crawler based on Hadoop system, including the design for basic flow of the crawler, frame design, function module and the module's Map / Reduce design. Based on the outline design, this paper made the detailed design, and implements the entire system, Including implementation of data storage structure, overall data structure of crawler system

火龙果•整理
uml.org.cn

and implementation of the various functional modules. Finally, this makes a detailed summary of this article.

This article is about the implementation of a distributed crawler system based on Hadoop, the system uses Map / Reduce computing framework consistent with the overall project distributed framework. Solve low efficiency and poor scalability of the single crawler system, improved the speed of information gathering and expanded the scale of information collection. Meanwhile, the system provided data for index module and information processing module of "distributed cross-language information access and retrieval platform ".

Keywords: distributed crawler system; hadoop; hdfs; map/reduce;

火龙果•整理
uml.org.cn

目录

摘　要.....I
Abstract.....II
第 1 章 绪论.....1
1.1 课题来源.....1
1.2 课题研究背景及意义.....1
1.3 本文主要工作及内容.....4
1.4 本文的主要组织和结构.....5
第 2 章 相关技术研究.....8
2.1 云计算相关知识.....8
2.1.1 云计算定义.....8
2.1.2 云计算基本原理.....9
2.1.3 云计算的体系结构.....10
2.2 Hadoop 分布式平台相关知识.....11
2.2.1 分布式文件系统.....12
2.2.2 Map/Reduce 分布式计算模型.....15
2.3 网络爬虫原理.....16
2.3.1 网络爬虫系统的工作原理.....17
2.3.2 网络爬虫系统的基本结构.....17
2.3.3 分布式网络爬虫的工作原理.....18
2.4 分布式网络爬虫研究现状.....19
2.5 本章小结.....20
第 3 章 功能分析与结构设计.....21
3.1 分布式网络爬虫的设计需求.....21
3.2 系统布局.....22
3.3 爬虫系统基本结构.....24
3.3.1 爬虫基本流程设计.....24
3.3.2 爬虫系统的框架设计.....27
3.4 功能模块的 Map/Reduce 设计.....29
3.4.1 CrawlerDriver 模块设计.....29

火龙果•整理
uml.org.cn

3.4.2 ParserDriver 模块设计 31
3.4.3 OptimizerDriver 模块设计 33
3.4.4 MergeDriver 模块设计 35
3.4.5 HtmlToXMLDriver 模块设计 37
3.5 本章小结 39
第 4 章 分布式爬虫具体实现 40
4.1 存储结构的实现 40
4.1.1 待抓取 URL 库 40
4.1.2 原始网页库 40
4.1.3 链出 URL 库 41
4.1.4 XML 库 42
4.2 爬虫总体数据结构 42
4.3 功能模块的实现 44
4.3.1 CrawlerDriver 模块 44
4.3.2 ParserDriver 模块 46
4.3.3 OptimizerDriver 模块 48
4.3.4 MergeDriver 模块 50
4.3.5 HtmlToXMLDriver 模块 51
4.4 本章小结 53
第 5 章 性能分析与评价 54
5.1 系统运行展示 54
5.2 数据统计与分析 59
5.3 与非分布式爬虫的比较 60
5.4 本章小结 61
结 论 62
附录一 抓取数据统计表 63
参考文献 66
攻读学位期间发表的学术论文 69
哈尔滨工业大学学位论文原创性声明及使用授权说明 70
致 谢 71

火龙果•整理
uml.org.cn

第1章 绪论

1.1 课题来源

本课题来源于哈工大语言语音教育部-微软重点实验室机器智能与翻译研究室项目“分布式跨语言信息获取和检索平台”的子课题。该课题的主要目标是设计实现一个基于 Hadoop 的分布式爬虫系统，该爬虫针对 27 种语言主流媒体网站的信息进行采集，并分别保存，为后面的信息处理、创建索引和检索提供数据。

1.2 课题研究背景及意义

目前，互联网正处于飞速前进的发展过程中，进入 21 世纪以来其发展势头已远远超出人们想象。仅就我国而言，根据中国互联网络信息中心 2011 年 1 月发布的《第 27 次中国互联网络发展状况统计报告》最新数据显示[1]：“截至 2010 年 12 月底，我国网民规模突破 4.5 亿大关，达到 4.57 亿，较 2009 年底增加 7330 万人；互联网普及率攀升至 34.3%，较 2009 年提高 5.4 个百分点。我国手机网民规模达 3.03 亿，较 2009 年底增加 6930 万人。”

由于互联网规模的不断扩大，在互联网这个大平台上的信息正在以指数级增长，各种各样的信息资源被整合到一起，而且大量数据属于地理位置分散的异构数字化信息，所有信息形成了一个宏大的信息库。这个信息库包括极其巨大的海量数据，所以，如何快速、高效、安全地从浩瀚的信息海洋中找到网络用户感兴趣的信息已成为当前互联网发展的主要目标。为了解决这一问题，搜索引擎应运而生，它的出现大大提高了网络用户搜集和查找信息的能力。

然而随着互联网用户的增多，用户的需求也越来越多样化，搜索引擎的发展必须跟着用户的需求变得丰富、完善。当下，评价搜索引擎性能的好坏主要有三个指标[2,23]:“首先，原始网页数目的多少，这主要与搜索引擎爬虫的性能和规模有关，原始网页数目多、原始网页库规模大才会满足用户丰富多彩的需求，搜索引擎才会有市场；其次，搜索引擎的性能，主要体现在两方面，一是实时性，爬虫必须在较短的时间内更新原始网页库中最新数据。二是实效性，搜索引擎必须在较短的时间内返回用户提交 query 对应的结果集合；最后是搜

火龙果•整理
uml.org.cn

索的效果，搜索引擎需要能够采用比较优秀的检索算法和排序算法，将用户想要的结果返回。”

如何解决在海量数据的情况下时间与空间上的矛盾，被公认为是下一代信息检索技术研究的一个方向。面对巨大的数据、大量的检索请求和用户对于检索时间的苛刻要求，信息检索的效率成为一个亟待解决的问题，依靠单机不可能完成这样的任务。如果单纯地依靠提高硬件性能即使硬件发展足够快也无法满足信息增长的速度，所以必须依靠分布式信息检索技术才能解决。事实上，几乎所有使用的大型搜索系统都采用了分布式的体系结构来处理海量数据从而解决信息检索中的效率问题。

分布式搜索引擎是将很多台机器通过互联网络链接到一起，然后再软件层次使得多台机器可以协同解决大规模数据的处理、索引和检索问题。虽然它物理上把多台机器连接到了一起，但是这个集群在逻辑上仍然是一个整体。分布式搜索引擎有着得天独厚的优势，它可以很轻松地解决传统搜索引擎无法解决的问题，主要优势可以总结为以下五点[3,24]：

搜索引擎中的存储资源、处理器资源、内存资源和数据全部被检索服务器所共享，并且检索服务器分布在不同地地理位置，当用户通过客户端提交一个检索请求时，客户端自动地向距离最近的检索服务器提交请求，这样减轻了其他检索服务器的负担。

各个站点客户端的代理服务器相互合作更好地服务于用户的请求。

后台信息处理的分布式计算模型有较好的可扩展性，且与前端别叫独立，便于维护。

各种信息的索引分散地存在于不同机器上的索引数据库中，且这些分散的索引通过统一的分布式文件系统模型整合到一起，这样每一个索引数据库的容量很小，这使得查询速度会更快。

容错性好，一但某些检索器或者索引器发生故障，不影响整个搜索引擎。

进入 21 世纪以来，互联网泡沫破碎的阴影尚未走远，Web2.0 的概念便蓬勃发展起来，这个开创性概念的提出给处在衰退期的互联网打入一针强心剂。在这个方兴未艾的 Web2.0 时代，Facebook、RenRen、Twiter 等社区网站，抢走了很多传统门户网站的用户。大用户量以及高用户参与度，是这些社区网站的特点。因此，如何高效地服务于海量用户群体，使用户方便并快捷地体验网站所提供的服务，已经成为这些社区网站亟待解决的问题。

同时，依靠 Google 海量数据计算模型构建起来的 Google 计算集群，给 Google 提供了强大的信息处理能力而且使得 Google 在海量数据中检索的速度

火龙果•整理
uml.org.cn

大大提高。因此，如何高效地利用这种海量数据处理技术，使得更多的公司或者个人也能够拥有强大的计算与信息处理能力，成为那些拥有巨大数据资源的企业开始考虑的问题。

正是因为互联网行业中对海量数据处理的需求，云计算就应运而生。由于云计算的概念才刚刚提出，每个企业、个人对它都有不同的理解，因此它还没有一个明确的定义，但是我们纵观各大企业权威对云计算的定义，我们可以总结出云计算的一些特点。下面举几个例子可以代表这些特点，Wikipedia 关于云计算的定义为“云计算是网格计算下的一种新的(大约在 2007 年底出现)标签，它使用公用计算或其它方法来共享计算资源。云计算是依靠本机服务器或个人设备来处理用户应用程序之外的另一种选择”；News Blog 认为“云计算是一种将硬件与软件外包给因特网服务提供商的概念”； Forrester 则认为“云计算看起来十分像是一种典型的突破性技术”。

到目前为止，微软、Amazon、Google、EMC、IBM 和 HP 等众多 IT 业巨头都宣布要重点建设企业云计算框架。也有了很多成功的企业级云计算案例。

在所有企业级案例中最为人们所熟知的莫过于 2009 年 10 月 Google 和 IBM 联合开启的“云计算”计划[25]，包括 Carnegie Mellon University、Stanford University、University of California, Berkeley、University of Washington、MIT 在内的众多知名大学都参加了这一计划，同事国内的清华大学也在 2010 年 3 月份申请加入了这一计划。在这项计划中，高校和企业都是受益者，高校可以利用两个 IT 巨头的资源开发出各种创新的应用；而对于企业他们可以利用高校得天独厚的科研力量去探索和发展云计算。

与此同时，Amazon 也不甘落后他们推出自己的云计算框架 EC2[26]，这项 Amazon EC2 服务可以看成是一个开放的云计算系统，它可以为用户提供海量数据计算的能力，有效地降低数据处理的时间，同时比自己搭建一个云计算平台要方便的多。

当然这些 IT 巨头的成功案例中最成功的当属 Google，目前应用最广、最著名的云计算基础设施思想就是 Google 提出来的。Google 在一篇名为《Web Search For A Planet-The Google Cluster Architecture》[4]的文章中提出了他的云计算基础设施模式，在这篇文章中 Google 提出他们的云计算基础设施包括四个部分：分布式文件系统 GFS(Google File System)[5] ，Map/Reduce 编程模型[6]，分布式的锁机制 Chubby[7]和大规模分布式数据库 BigTable[8]。

正是由于云计算技术越来越被重视，实验室在信息检索和自然语言处理领域已有的研究基础上，设计实现一个分布式跨语言检索平台，该平台采用云计

火龙果•整理
uml.org.cn

算及云存储。其中数据获取、数据处理均采用分布式框架。本人的主要工作是在开源分布式框架 Hadoop 平台上搭建分布式爬虫。本文是结合所作工作阐述如何在 Hadoop 平台上设计和实现分布式爬虫。

1.3 本文主要工作及内容

Hadoop 由 Apache Software Foundation 公司于 2005 年秋天作为 Lucene 的子项目 Nutch 的一部分正式引入。它受到最先由 Google 开发的分布式计算模型(MapReduce) 和分布式文件系统(GFS)的启发，是谷歌所提出的这两部分的实现。2006 年 3 月份，MapReduce 和 Nutch Distributed File System (NDFS) 分别被纳入称为 Hadoop 的项目中。

Hadoop 框架中最核心的设计就是：Map/Reduce 和 HDFS。Map/Reduce 本身就是用于并行处理大数据集的软件框架。

Map/Reduce 的本质是高度抽象后的 Map 函数和 Reduce 函数。它可以理解为两个操作过程：Map 过程和 Reduce 过程，每一个过程可能有多个 Map 函数火车 Reduce 函数。Map 函数的载体接受一组数据，在执行 Map 函数之前，这组输入数据会被解析成键/值对列表，Map 函数计算的对象是一个键/值对。Reduce 函数接受 Map 函数的输出结果，在执行 Reduce 函数之前会将键相同的值聚到一起交给 Reduce 处理。

Hadoop 分布式文件系统(HDFS)是一个运行于 Hadoop 框架之上的文件系统，他的成功设计可以让这个文件系统运行于普通的个人电脑之上。它与现有的文件系统有许多相似之处，然而，它与其他的分布式文件系统的差别也非常显著。主要表现在：HDFS 的高度容错性、成本低廉性、拥有高吞吐量访问数据的接口、可通过 POSIX 流式访问和适合于开发海量数据的应用程序。HDFS 的前身为 Apache Nutch 的网络搜索引擎项目的基础设施。HDFS 现在是一个 Apache Hadoop 的子项目。

本文是在深入研究 Hadoop 的原理和应用基础上，结合实验室需求，设计和开发了基于 Hadoop 的分布式爬虫。并且目前已投入长期运行和使用。研究工作主要包括以下几点：

调研 ABICloud[31]、Hadoop[30]、Sector/Sphere[28]等目前应用较广泛的开源分布式框架。为分布式爬虫的设计实施提供技术支撑。

熟悉 Hadoop 的分布式文件系统(HDFS)。包括文件系统的结构、节点之间交互的通信协议、容错方式、数据访问方式。以方便爬虫系统的开发。

火龙果•整理
uml.org.cn

掌握 Hadoop 的 Map/Reduce 编程模型，包括 Map/Reduce 编程的整个流程、作业的配置、如何使用 Map/Reduce 方式读写 HDFS 上数据、具体的 Map 函数与 Reduce 函数的实现方式和数据分块的几种方式。

设计基于 Hadoop 分布式爬虫的各个模块，使得各个模块满足 Map/Reduce 编程模式，以实现爬取过程的分布式计算。每个模块对应一个 Map/Reduce 过程并完成一个功能。

开发分布式爬虫，设计实施方案确定分布式爬虫的整个流程及编码实现各个模块，完成各个模块的功能。

在伪分布式单节点环境下部署爬虫系统，测试爬虫的抓取效率、稳定性。同时，对 Hadoop 分布式配置环境进行优化，通过实验进行各种功能测试和性能测试，找出合适的 Map、Reduce 个数，从而更好地利用 Hadoop 进行分布式协同处理。

1.4 本文的主要组织和结构

本文首先介绍云计算的发展状况及云计算相关原理，然后简单介绍目前应用较广泛的第三方云计算框架 Hadoop，包括 Hadoop 分布式平台结构和如何在 Hadoop 平台开发自己的应用。另外，还会介绍分布式爬虫的研究现状及爬虫的基本原理。其次，在这些详细调研基础上，针对项目的需求提出基于 Hadoop 平台分布式爬虫的设计方案，完成整个系统的结构设计及流程控制，将爬虫系统按照功能分解成几个模块，每个模块对应一个 Map/Reduce 模式以适应 Hadoop 分布式计算的需要。再次，对分布式爬虫的数据存储方式和数据格式信息进行了设计，并完成了爬虫各分模块具体类的实现。最后，进行爬虫的各种测试，包括分模块测试、功能测试、性能测试，同时结合 Hadoop 集群配置的优化策略找出最合适的 Map、Reduce 个数，从而更好地利用 Hadoop 的分布式协同处理。在文章最后，详细分析本系统并进行评价，对本课题进行总结。

本文的内容结构共分为五章，介绍如下：

第一章，绪论，简单介绍本课题的来源、阐述本课题的研究背景及研究意义，另外，详细讲述本文的主要工作及研究内容并给出全文的组织结构。

第二章，相关技术研究，本章是对分布式爬虫相关技术进行详细介绍，主要包括四个部分：云计算技术现状、原理，Hadoop 分布式平台结构，爬虫的基本原理，分布式爬虫的研究现状。在云计算技术现状、原理部分，本文从机

火龙果•整理
uml.org.cn

体结构、服务和技术三个层次来阐述云计算技术的发展现状和原理。在Hadoop 分布式平台结构部分，本文对 Hadoop 的两个核心技术：分布式文件系统(HDFS)和分布式计算模型(Map/Reduce)进行详细介绍。在爬虫的基本原理部分，本文从爬虫种类、工作方式、搜索策略等方面来分析爬虫的基本原理。在分布式爬虫的研究现状部分，本文主要介绍 Google、Mercator、Internet Archive 等目前较著名的分布式爬虫。

第三章，功能分析与结构设计，本章针对实验室需求给出基于 Hadoop 分布式爬虫的设计方案，同时完成爬虫系统的结构设计。主要内容包括 5 个部分：分布式网络爬虫的设计需求、系统布局、爬虫系统基本结构、爬虫子模块的分布式结构和分布式爬虫的功能完善。在分布式网络爬虫的设计需求部分，本文主要结合实验室需求来阐述设计分布式爬虫的目标、要点。在爬虫系统布局部分，本文给出分布式跨语言检索平台的架构、平台的模块划分以及爬虫在整个平台中的地位。在爬虫系统基本结构部分，本文将介绍基于 Hadoop 爬虫的基本流程图、爬虫的框架设计和爬虫的模块划分。在爬虫子模块的分布式结构部分，本文将介绍各个子模块的 Map-Reduce 实现。在分布式爬虫功能完善部分，本文主要是针对爬虫系统设计和实现时遇到到一些技术细节问题做出相应的解决方案。

第四章，分布式爬虫具体实现，本章是在第三章设计的基础上对基于Hadoop 分布式爬虫的子模块进行了具体设计，包括数据结构、数据格式、具体代码的编写等等。主要内容包括 5 个部分：抓取模块(CrawlerDiver)的实现、分析模块(ParserDiver)的实现、优化模块(OptimizerDiver)的实现、合并模块(MergeDriver)的实现和转化 XML 模块(HtmlToXmlDiver)的实现。在抓取模块部分，本文主要介绍爬虫抓取功能的 Map-Reduce 实现，包括下载网页、多线程任务等。在分析模块部分，本文主要介绍爬虫提取链出链接部分的 Map-Reduce 实现，具体包括 Map/Reduce 模型的设计、HTML 解析等。在优化模块部分，本文主要介绍如何用 Map/Reduce 实现对分析模块提取出来的链出链接进行过滤去掉重复链接。在合并模块部分，本文主要介绍去重功能的 Map-Rduce 实现。在转化 XML 模块部分，本文主要介绍将 HTML 转化为 XML 部分的 Map-Rduce 实现。

第五章，性能分析与评价，对整个基于 Hadoop 的分布式爬虫系统性能进行分析和评价，并在给出系统的界面展示。本章主要包括四个方面：首先，系统的界面展示，这个界面主要是分布式文件系统的 WEB UI，通过它可以直观地看到集群中全部已抓取网页。其次，对已抓取的数据进行统计和分析。再

火龙果•整理
uml.org.cn

次，与非分布式爬虫系统的性能比较，突出分布式爬虫的优点。最后，对本系统的关键问题的解决办法做出探讨和剖析。

总结和展望，最后对全文进行总结，归纳出设计的要点，梳理全文的结构，整理本文设计的基于 Hadoop 分布式爬虫的优点和缺点。

火龙果•整理
uml.org.cn

第2章 相关技术研究

本章着重介绍全文所用到的相关技术，由浅入深，为分布式网络爬虫的搭建提供一个技术背景。由于云计算是分布式处理的发展，因此本章首先对云计算概念、架构等相关理论知识进行介绍。其次，本章将对本分布式爬虫系统的技术基础即 Hadoop 分布式处理平台做出深入剖析，主要包括分布式存储模型(HDFS)和分布式计算模型(Map/Reduce)。再次，本章将讨论网络爬虫的基本原理，以及爬虫的结构。最后，本章将介绍目前分布式网络爬虫的发展现状。通过本章将奠定全文的技术基础。

2.1 云计算相关知识

云计算（Cloud Computing），是一种新兴共享计算资源的计算方法，它提出将互联网云中海量的计算机联合起来协同提供计算服务。“很多因素推动了对这类环境的需求，其中包括连接设备、实时数据流、SOA 的采用以及搜索、开放协作、社会网络和移动商务等这样的 Web2.0 应用的急剧增长，另外，数字元器件性能的提升也使 IT 环境的规模大幅度提高，从而进一步加强了对一个由统一的云进行管理的需求[20]。”云计算的支持者认为云计算将是划时代开创性的计算模型，因为它让超级计算变为现实。企业或者个人无需再对硬件发展的瓶颈望洋兴叹，也不必再花费大量资金和财力购买超级服务器，只需在软件层次搭建一个云计算框架便可以完成海量数据的超级计算。

2.1.1 云计算定义

目前，云计算的概念刚刚兴起，每一个企业或者云计算的先驱探索者对云计算都有自己的理解，这体现了云计算巨大的包容性，也说明云计算的火热程度¬¬，因为每个企业都想在云计算中分一杯羹，所以都会在自己的应用角度去定义云计算。这样本文对云计算的定义也是取众家之长、避众家之短。下面是对云计算比较权威的定义。

维基百科（Wikipedia.com）认为“云计算是一种能够将动态伸缩的虚拟化资源通过互联网以服务的方式提供给用户的计算模式，用户不需要知道如何管

火龙果•整理
uml.org.cn

理那些支持云计算的基础设施[9]。”

Whatis.com 认为“云计算是一种通过网络连接来获取软件和服务的计算模式，云计算使得用户可以获得使用超级计算机的体验，用户通过笔记本电脑与手机上的瘦客户端接入云中获取需要的资源[10]。”

美国加州大学伯克利分校(UC Berkeley)在 2009 年 2 月发表的一篇关于云计算的报告《Above the Clouds: A Berkeley View of Cloud Computing》中认为“云计算既指在互联网上以服务形式提供的应用，也指在数据中心中提供这些服务的硬件和软件，而这些数据中心中的硬件和软件则被称为云[11]。”

商业周刊（BussinessWeek.com）在 2008 年 3 月发表的一篇名为《Google 及其云智慧》文章中指出，“Google 的云就是由网络连接起来的几十万甚至上百万台的廉价计算机，这些大规模的计算机集群每天都处理着来自于互联网上的海量检索数据和搜索业务请求[12]。”商业周刊在另一篇名为《解读亚马逊的云计算帝国》的文章中总结说，“从 Amazon 的角度看，云计算就是在一个大规模的系统环境中，不同的系统之间相互提供服务，软件都是以服务的方式运行，当所有这些系统相互协作，并在互联网上提供服务时，这些系统的总体就成为了云[13]。”

IBM 认为“云计算是一种共享的网络交付信息服务的模式，云服务的使用者看到的只有服务本身，而不用关心相关基础设施的具体实现。本书沿用 IBM 的定义，云计算是一种革新的 IT 运用模式。这种运用模式的主体是所有连接着互联网的实体，可以是人、设备和程序。这种运用方式的客体就是 IT 本身，包括我们现在接触到的，以及会在不远的将来出现的各种信息服务。而这种运用方式的核心原则是：硬件和软件都是资源并被封装为服务，用户可以通过互联网按需地访问和使用[14]。”

2.1.2 云计算基本原理

云计算是对网格计算（Grid Computing）、并行处理（Parallel Computing）和分布式处理（Distributed Computing）的改进处理，其前身是利用并行计算解决大型问题的网格计算和将计算资源作为可计量的服务提供的公用计算，在互联网宽带技术和虚拟化技术高速发展后萌生出云计算[21]。

通过上面云计算概念的理解，我们可以给出云计算的基本原理：利用互联网中相互连接的而地理上分布式存在的计算机集群为用户提供计算、存储、软硬件等服务。这使得用户可以充分地利用集群中每一台计算机去计算和存储他

火龙果•整理
uml.org.cn

们的数据。云计算就是将普通的个人电脑或者服务器连接起来组成一个集群，相当于获得这些计算机计算能力和存储能力的加和，但是所花销的成本更加低廉。云计算采用按需分配资源的构想，使得硬件与软件得到充分地利用。云计算概念的出现降低了普通人使用云计算的门槛，普通开发人员不再需要了解集群底层的实现就可以很轻松地开发他们的应用，尽情地享受集群给他们带来的高性能计算。同时，用户不需要知晓集群中每一台机器在哪里，你所用 API 的底层实现，只需透明地享用云中的全部资源。

云计算这个全新概念的实现最关键的点在于虚拟化技术，它将地理上分散式存在的机器在软件层次上构成一个超大虚拟计算池，让池中所有机器的内存、CPU 和硬盘组合在一起协同工作。

说了这么多通俗点讲，云计算就是将所有的硬件设备、应用软件和软件服务放到广域网或者互联网。其中，广域网我们可以理解为私有云，互联网我们可以理解为公有云。

2.1.3 云计算的体系结构

在描述了云计算的基本原理之后，接下来讲解云计算的体系结构。在体系结构中我们可以看到一般的云计算体系分为几个模块，各模块的功能，模块之间的联系，资源如何实现共享以及模块之间是如何在虚拟化技术的基础上实现交互的。图 2-1 为一般的云计算体系结构。

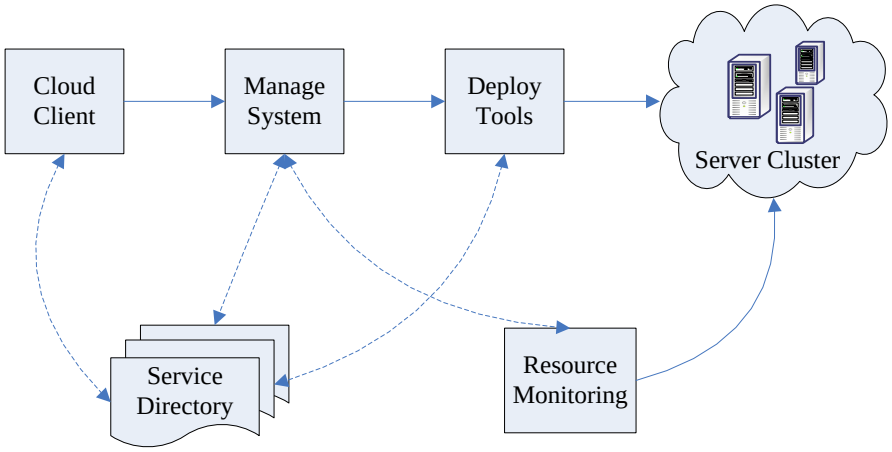

图 2-1 云计算平台的体系结构

火龙果•整理
uml.org.cn

(1)Cloud Client：是用户进入云的入口，用户可以通过它与云中服务器集群进行交互，它提供一个友好的客户端界面帮助用户完成注册、登陆、申请服务、管理资源和部署集群等功能。云客户端达到的目的是让用户感觉不到云的存在，操控云应该像操纵本地电脑一样。

(2)Service Directory：用户完成云注册后会获得云的使用权，这时云的服务端会为用户提供一个服务目录，用户可以通过客户端去云中申请、取消或者更改他们在云中的服务。

(3)Manage System：这个系统是为云的管理者设计的，管理者也是通过客户端管理云，区别是他与用户的云权限不同。管理者通过管理系统为用户授权、给用户分配志资源和部署云集群。

(4)Deploy Tools：提供给云管理者所用，帮助管理者部署集群和配置资源。同时，生成用户的服务列表。

(5) Resource Monitoring：辅助管理系统实时监测云中各种资源的使用状况，资源使用出现问题时及时调整以使得云中负载得到均衡。

(6)Server Cluster：由管理系统管理的物理服务器集群，这些服务器相互连接、交互信息构成一个云网络，是云中真正负责计算和存储的关键部分。他们通过中间的管理系统每天处理数以万计的用户云服务请求。

上面这六个模块就是云计算的体系结构，目前包括 Hadoop 在内的众多云平台均采用这个体系结构。

2.2 Hadoop 分布式平台相关知识

“Hadoop 是 Apache 公司的一个开源项目，这个项目一个可靠的、可扩展的分布式框架。Hadoop 的设计思想完全吻合 Google 开发的 Map/Reduce 和 Google File System。[27]”在 Hadoop 的分布式框架中，计算模型依然叫做 Map/Reduce，但是存储结构叫做 Hadoop Distributed File System。这两个部分可以说是 Hadoop 的核心。

从上面的介绍可以看出，Hadoop 的框架结构主要由 HDFS(Hadoop Distributed File System)和 Map/Reduce 计算模型两部分组成：

(1)分布式文件系统(HDFS) 是一个在多台物理机器之上协同组织和管理文件的系统。在 Hadoop 上用户管理文件和管理一台机器上的文件一样。

(2)Map/Reduce 计算模型是一种编程模型，在 Hadooop 框架下使用 Map/Reduce 编程模型可以让多台机器协同计算一个任务。

火龙果•整理
uml.org.cn

基于 Hadoop 的分布式平台框架如图 2-2：

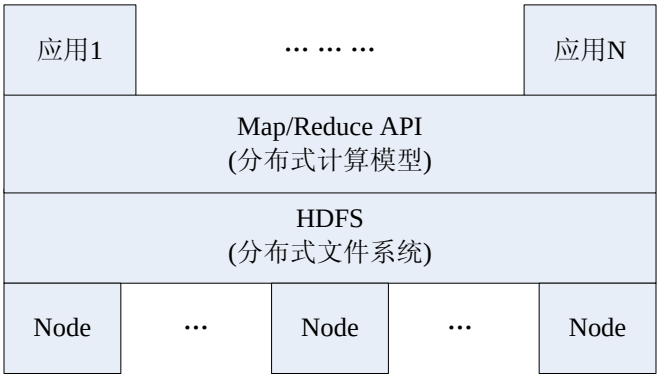

图 2-2 Hadoop 分布式平台框架图

从图中可以清晰地看出 Hadoop 分布式平台的框架，在最底层的是一个个物理的计算机节点。在其上有一层分布式文件系统(HDFS)，这个 HDFS 将每个节点上的计算机整合到一起。并且这个 HDFS 对用户完全是透明的，即让用户觉得分布式文件系统像是在一台机器上的文件系统。在文件系统之上就是分布式计算框架 Map/Reduce，它可以将一个任务分解成若干个子任务分配到各个节点上去执行从而实现分布式编程，在这一层 Hadoop 为用户提供了一套分布式编程的 API，用户不用关心海量数据如何被分布到不同的机器上，不需要考虑机器失效的处理，不需要考虑各节点间如何协同操作共同完成任务，其简化了程序员的负担，以让不具备分布式系统经验的程序员，能够轻松地进行分布式编程。

看完了 Hadoop 平台的框架图，下面将讨论 Hadoop 分布式平台的两个重要组成部分：Hadoop 分布式文件系统(HDFS)和 Hadoop 分布式计算模型(Map/Reduce)。

2.2.1 分布式文件系统

“分布式文件系统（Distributed File System）是指文件系统管理的物理存储资源不一定直接连接在本地节点上，而是通过计算机网络与节点相连。现在的分布式文件系统一般还是保持与最基本的本地文件系统几乎相同的访问接口和对象模型， 这主要是为了向用户提供向后的兼容性，也保持原来的简单的对象模型和访问接口。但这并不说明文件系统设计和实现的难度没有增加。恰恰由于对用户透明地改变了提供结构，满足用户的需求，以掩盖分布式文件操作的复杂性，大大增加了分布式文件系统的实现难度”[15]。

火龙果•整理
uml.org.cn

基于这个概念，Hadoop 开发了一套类似谷歌分布式文件系统(Google Distributed File System,GFS)的分布式文件系统(HDFS)。HDFS 是 Hadoop 分布式框架的存储模块。HDFS 可以执行的操作有创建、删除、移动或重命名文件等，这些操作类似于传统的单机文件系统。HDFS 与其他的分布式文件系统相比有如下特性[29]：

(1)可以存储超大文件，例如几百 TB 的文件。

(2)流式数据访问。HDFS 上文件访问的思想是一次写入、多次读取模型。目前 HDFS 还不支持插入操作。

(3)廉价的硬件设备。Hadoop 并不需要运行在昂贵的并且高可靠性的硬件上。因为 Hadoop 有一个比较完备的容错措施，不会让用户感觉到明显的中断。

HDFS 设计的架构是经典的 master/slave 结构。HDFS 分布式文件系统包括一个 NameNode 节点，这个节点作为 master 服务器管理文件系统的命名空间，以及控制客户端对分布式文件系统的访问。另外，HDFS 还包括许多个 DataNode 节点，通常每台计算机就是一个 DataNode 节点。他们作为 slave 负责管理本地机器上的存储。HDFS 向外提供一个统一的文件系统命名空间，并且允许用户以文件的形式存储他们的数据。从内部结构上来看，一个文件被分割成一个或者多个块(block)，这些块被分散地存储在多个 DataNode 节点上。NameNode 执行着诸如打开、关闭、重命名文件和目录等操作，还管理着块(blocks)与 DataNodes 之间的映射。DataNode 负责应对文件系统上用户对文件的读和写的请求。此外，DataNode 还负责在 NameNode 的统一调度下块(block)的创建、删除和复制。HDFS 结构如图 2-3。

图 2-3 中展示了整个 HDFS 中三个重要角色:NameNode, DataNode 和 Client。

(1)NameNode 是 HDFS 分布式文件系统的中心枢纽，它维护着整个文件系统中所有文件的目录树，以及树上所有的文件和目录的原信息。这些信息被永久地存储在本地的磁盘上。NameNode 还记录着每个文件的每个块所在的具体位置。

(2)DataNode 是 HDFS 分布式文件系统上真正存储数据的节点。当 Client 和 NameNode 进行读或者写操作时，它们负责存储和定位块，并且定时地像 NameNode 报告它们存储块的列表。

火龙果•整理
uml.org.cn

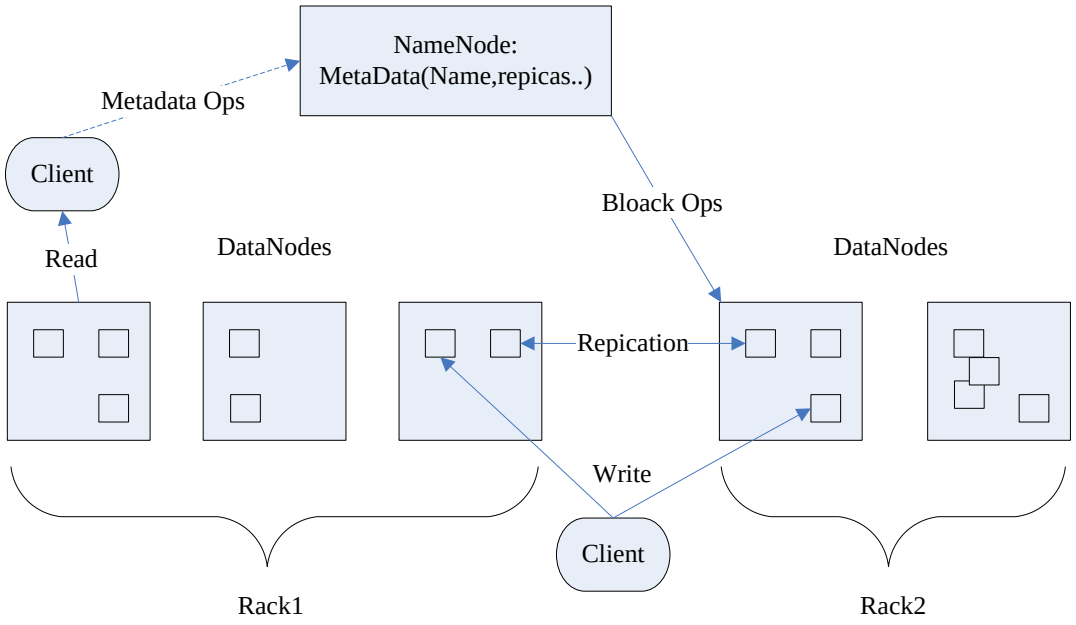

图 2-3 HDFS 结构结构图

(3)Client 代替用户通过 NameNode 和 DataNode 之间的交互来访问整个文件系统。Client 提供一个类似 POSIX 的文件系统接口，这样用户就不需要 NameNode 和 DataNode 是如何工作的。

Hadoop 分布式文件系统(HDFS)的设计和实现与其他分布式文件系统相比有如下几个特性[22]：

首先是透明性，具体包括访问的透明性、位置的透明性、移动的透明性、性能的透明性和伸缩的透明性。

其次是并发控制，客户端对于文件的读写不应该影响其他客户端对同一个文件的读写。

第三，文件复制功能，一个文件可以表示为其内容在不同位置的多个拷贝。这样可以提高容错的能力，如果有一个块坏掉了可以从其他的节点获得这个块。

第四，硬件和操作系统的异构性。HDFS 可以在不同的操作系统和计算机上实现同样的客户端和服务端程序。

最后，容错能力，在 HDFS 分布式文件系统中，尽量保证文件服务在客户端或者服务端出现问题的时候能正常使用是非常重要的。

火龙果•整理
uml.org.cn

2.2.2 Map/Reduce 分布式计算模型

Map/Reduce 是 Hadoop 分布式框架的编程模型，也是一个简易的软件框架，用户只需使用这套软件框架提供的 API 去编制自己的应用程序就可以让集群中所有机器去完成你的计算任务。用户并不需要了解 Map/Reduce 计算的底层实现。采用这样的编程模型用户可以轻松地处理海量数据。

Map/Reduce 任务在启动后先将数据的数据集分割成小的数据块，切分原则一般是 64M 为一个数据块，这些数据块被分别送到 Map 函数的执行者那里，然后被完全并行地处理。紧接着，这个 Map/Reduce 框架会排序 Map 的输出结果，这些排好序的结果作为输入交给 Reduce 任务。通常这个任务的输入和输出都是存放在分布式文件系统(HDFS)中。通常，这个框架会管理所有的计划任务，监控他们并重启那些失败的任务。

通常情况下，计算节点和存储节点都是一样的，那就是说 MapReduce 和 HDFS 在 Hadoop 上运行的节点集合相同。这种配置允许整个框架在已经存在数据的节点集合上去高效地计划任务，当然，这也导致了集群上高网络带宽的利用率。

Map/Reduce 框架有一个主节点作为任务调度者和多个从节点作为任务执行者，其中任务执行者在每台机器上存在一个。那个主节点负责规划和指定整个任务的 Map 和 Reduce 子任务在从节点上执行计划，并且监控他们，适时重启那些失败的子任务。而从节点则直接执行主节点规划的子任务。

用户基于 Map/Reduce 开发的应用程序至少应该定制输入输出文件的路径，并且提供执行 map 函数和 reduce 函数的类，这些类实现了 Mapper 和 Reducer 接口。上面这些和其他参数构成了 Hadoop 作业的配置。在执行任务时，客户端先向任务调度者提交它的任务和配置信息，即可执行文件 jar 包，然后任务调度者将任务和配置信息分配给从节点，让从节点开始执行相应的 Map/Reduce 任务。同时，任务执行者还会监控任务执行者，然后将状态和诊断信息返回给客户端。

下面我们来讨论数据在一个 Map/Reduce 作业中的是如何流动的。如图 2-4 展示的是 Map/Reduce 数据流图。

火龙果•整理
uml.org.cn

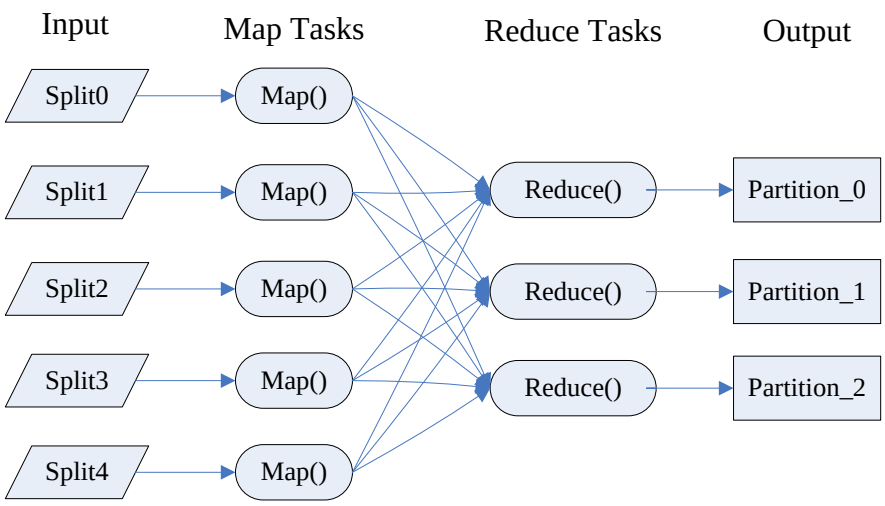

图 2-4 Map/Reduce 数据流图

在 Map/Reduce 任务开始时，输入数据会被分割成若干个分片(Split)，默认为每 64M 为一个分片。每一个分片会被送到一个 Map 函数去处理，处理的中间结果会将给 Reduce 函数处理。每个 Reduce 任务会创建一个分区(partition)。分区可以通过用户自定义的 partitioner 来控制，默认的是使用 hash 函数来形成“木桶”键值，这种方法效率比较高。

2.3 网络爬虫原理

Web 网络爬虫系统的功能是下载网页数据，为搜索引擎系统提供数据来源。很多大型的网络搜索引擎系统都被称为基于 Web 数据采集的搜索引擎系统，比如 Google。由此可见 Web 网络爬虫系统在搜索引擎中的重要性。

网页中除了包含供用户阅读的文字信息外，还包含一些链接信息。Web 网络爬虫系统正是通过网页中的连接信息不断获得网络上的网页。正是因为这样采集过程像一个爬虫或者蜘蛛在网络上漫游，所以它才被称为网络爬虫系统或者网络蜘蛛系统，在英文中称为 Spider。

本章主要介绍了翻译评价语料库的建设。在对语言学知识对翻译自动评价贡献的研究中，语料库的建设至关重要。因此，我们对语料库的建设过程进行了大量的统计分析，分析其合理性。最后还对语料人工标注的一致性进行了定量的分析。

火龙果•整理
uml.org.cn

2.3.1 网络爬虫系统的工作原理

Web 网络爬虫系统一般会选择一些比较重要的、出度(网页中链出链接数)较大的网站的 URL 作为种子 URL 集合。网络爬虫系统将这些种子集合作为初始 URL，开始数据的抓取。由于网页中含有链接信息，通过已有网页的 URL 会得到一些新的 URL，可以把网页之间的指向视为一个森林，每个种子 URL 对应的网页是森林中的一棵树的根节点。这样，Web 网络爬虫系统就可以根据先广搜索算法或者先深搜索算法遍历所有的网页。由于先深搜索算法可能会使爬虫系统陷入一个网站内部，不利于搜索比较靠近网站首页的网页信息，因此一般采用先广搜索算法采集网页。Web 网络爬虫系统首先将种子 URL 放入下载队列，然后简单地从队首取出一个 URL 下载其对应的网页。得到网页的内容将其存储后，再经过解析网页中的链接信息可以得到一些新的 URL，将这些 URL 加入下载队列。然后再取出一个 URL，对其对应的网页进行下载，然后再解析，如此反复进行，知道遍历了整个网络或者满足某种条件后才会停止下来。

2.3.2 网络爬虫系统的基本结构

通过上面 Web 网络爬虫系统基本原理的介绍，我们可以将一般的爬虫系统基本结构分为 6 个模块，这 6 个模块组成的爬虫系统基本结构图如图 2-5：

(1)配置模块。该模块允许用户通过配置文件来配置爬虫系统。比如，爬虫系统下载网页的深度(层数)、多线程抓取时的线程数、抓取同一网站两个网页的间隔时间和限制待抓取 URL 的正则表达式等等。

(2)已访问 URL 识别模块。由于一个网页的 URL 可能会被多次解析出来，所以为了防止同一网页被多次重复下载爬虫必须要有这个模块来过滤掉已抓取的网页。

(3)robots 协议模块。当网络爬虫系统第一次对某个网站进行网页采集的时候，要首先抓取 robots.txt，然后获知指定不该访问的目录。或者根据会根据网页的 Meta 信息判断哪些是服务器定义不能索引和访问的，然后只访问能够索引的页面。

(4)网页抓取模块。网页抓取模块主要完成对网页的抓取工作。通过 URL 建立与服务器的连接，然后获得网页内容。

火龙果•整理
uml.org.cn

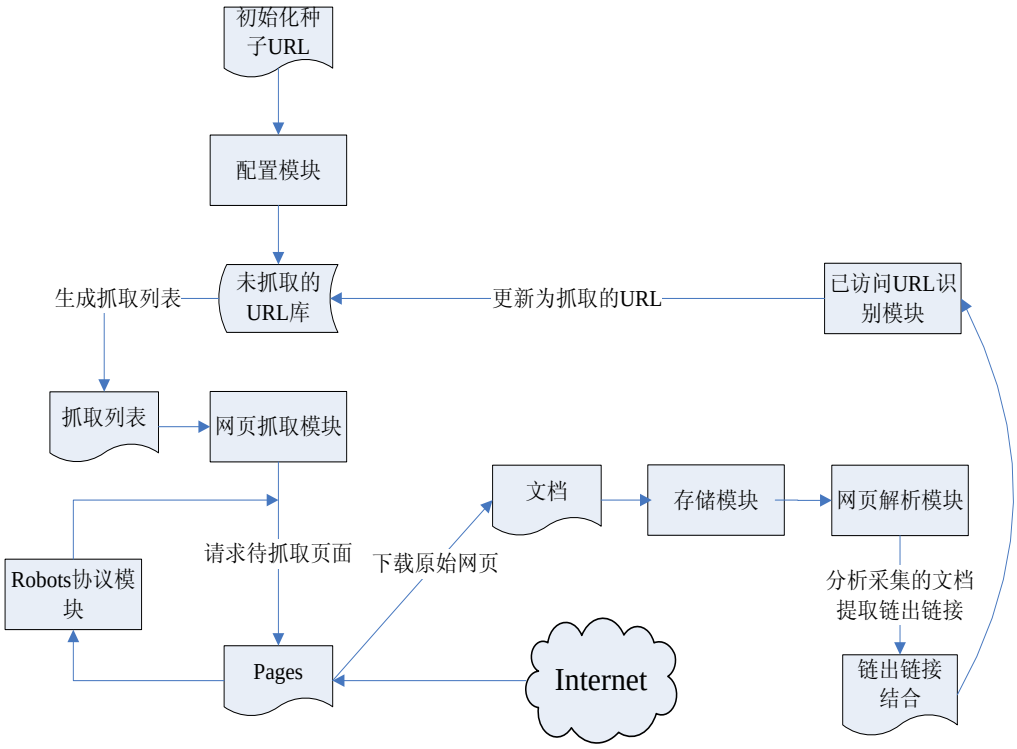

图 2-5 Web 爬虫系统的基本结构图

(5)网页解析模块。从已下载的网页中提取链出链接，然后把这些提取出的 URL 放入下载队列。

(6)存储网页模块。这个模块的作用是将已经下载网页经过一定的组织存储在本地服务器上或者分布式文件系统中。以备搜索引擎后续模块的处理。

上面这个基本结构是 Web 网络爬虫系统必须具备的。在应用时，由于不同的爬虫系统对各个模块的组合方式不同，因此构成了不同的系统结构。

2.3.3 分布式网络爬虫的工作原理

本节的前两个小节描述的是设计一个集中式爬虫系统所必须考虑的两个问题，但是，不论分布式爬虫系统还是集中式爬虫系统都需要考虑这两个核心工作原理与核心基本结构。因为，分布式网络爬虫可以看做是多个集中式网络爬虫系统组合而成。结合上面给出的集中式爬虫的核心工作原理和核心基本结构，下面本节来阐述分布式网络爬虫的工作原理。

分布式爬虫系统是运行于机器集群之上的，集群中每一个节点都是一个集

火龙果•整理
uml.org.cn

中式爬虫，其工作原理与集中式爬虫系统的工作原理相同。这些集中式爬虫在分布式爬虫系统中是由一个主节点控制来协同工作的。由于分布式爬虫系统要求多个节点协同工作，这样多个节点需要相互通信来交互信息，所以搭建分布式爬虫系统的关键是网络通信。因为，分布式爬虫系统可以利用多个节点抓取网页，所以，分布式爬虫系统的效率远远高于集中式爬虫系统。

分布式爬虫系统的体系结构有很多种，工作方式和存储方式也很多。但是，典型的分布式爬虫系统都采取主从方式的体系结构。即有一个主节点控制所有从节点执行抓取任务，这个主节点负责分配 URL，保证集群中所有节点的负载均衡。另外，关于存储方式，比较流行的是将抓取的网页保存在分布式文件系统上，这样管理多个节点上的数据更加方便。本文采用的就是这样的存储方式，使用的分布式文件系统是 Hadoop 的 HDFS 系统。

2.4 分布式网络爬虫研究现状

目前，最成功的分布式 Web 网络爬虫系统主要应用在搜索引擎公司(如：Google)和其他商业性较强的公司里。然而，这些公司并没有公布他们分布式爬虫的技术细节并且云计算的应用尚处在萌芽阶段。现在比较著名的分布式网络爬虫有 Mercator、UbiCrawler、WebFountain 和 Google Crawler。

Mercator[16]分布式爬虫项目是一个例外，因为它的整个设计方案全部公布于众了。这个分布式网络爬虫是由 Compaq 完成的，目前 AltaVista 搜索引擎正在使用这个爬虫系统。而且 Google 在 1998 年最初也是用的这个爬虫系统[17]。

UbiCrawler[18]分布式爬虫项目是一个高性能、大规模的分布式网络爬虫，这个爬虫主要特点是平台独立性即可以跨平台、大规模分布式性能良好、高容错性、高效的分配函数(基于一致性哈希函数)和可自行调整爬行任务优先级。另外该爬虫是由 Java 编写完成。

WebFountain[19]分布式爬虫项目是由 IBM 开发完成的。WebFountain 与 Mercator 类似采用分布式模块化，但是使用 C++语言编写的。它的特点是一个管理员机器控制一系列的蚂蚁机器。经过多次下载页面后，页面的变化率可以推测出来。这时，一个非线性的方法必须用于求解方程以获得一个最大的新鲜度的访问策略。

Google Crawler 爬虫系统使用成千上万台小型机和微机进行合作，完成分布式抓取工作的。它使用一个 URLServer 分配 URL 给网络爬虫让他们去爬取。网络爬虫爬取的网页再传送到一个 StoreSever 上去，由 StoreSever 将网页

火龙果•整理
uml.org.cn

进行压缩并将其存储起来。

2.5 本章小结

本章主要是对本文所要用到的相关技术作以阐述，为后面的系统设计和实现提供技术支持。

阐述内容大体可概括为云计算相关知识概述、Hadoop 分布式平台相关知识概述、网络爬虫的基本原理介绍和分布式网络爬虫现状的调研这四个方面。

火龙果•整理
uml.org.cn

第3章 功能分析与结构设计

通过上一章相关技术的介绍和研究，我们可在此基础上结合实验室需求给出基于 Hadoop 分布式网络爬虫的设计方案，同时完成爬虫系统的结构设计。首先，本章给出分布式网络爬虫的设计需求，包括爬虫的目标、要求还有些实验室项目需求。其次，本章给出基于 Hadoop 分布式系统框架的跨语言检索系统布局，对爬虫系统有个宏观的认识，确定爬虫系统在跨语言检索系统中的位置。再次，本章给出基于 Hadoop 爬虫的基本流程图、爬虫的框架设计和爬虫的模块划分。最后，给出爬虫系统各个子模块的分布式结构，以及它们的 Map/Reduce 实现方式。

3.1 分布式网络爬虫的设计需求

这个分布式网络爬虫系统是实验室项目“分布式跨语言信息获取和检索平台”的子课题。该项目大致可以分为两部分：信息获取部分和信息检索部分。而本文完成的是信息获取部分。我们先来介绍“分布式跨语言信息获取和检索平台”这个项目：首先，该项目简单来理解就是针对一种语言的查询词，检索出多种语言书写的信息，目前，我们完成的是 27 种语言的跨语言检索。这 27 种语言分别是中文、英文、日文、韩文、德文、乌尔都语、法语、俄语、西班牙语、葡萄牙文、阿拉伯文、印地语、意大利文、印尼语、马来语、缅甸语、波斯语、越南语、蒙古语、哈萨克语、吉尔吉斯语、尼泊尔语、菲律宾、普什图、老挝、塔吉克、藏语。其次，该平台检索范围专门针对这 27 种语言的新闻信息。再次，该项目还要求不论是爬虫、索引还是检索都要采用分布式框架。

结合实验室项目“分布式跨语言信息获取和检索平台”的需求，我们可以总结出分布式网络爬虫的具体实现要求，其需求细节如下：

(1)爬取信息类型：由于这个跨语言检索平台检索范围是针对 27 种语言的新闻信息，所以在数据获取方面，爬虫系统爬取的信息类型是 27 种语言对应的新闻网站上面的信息。

(2)分布式框架：由于该项目全部采用分布式架构，所以爬虫也应符合这一架构。本爬虫系统选择 Hadoop 分布式框架。Hadoop 是一个目前较为流行的分

火龙果•整理
uml.org.cn

布式框架。

(3)灵活性：主要体现在配置简单灵活。由于每一种语言算是一个特定的任务，所以本爬虫系统对每一种语言提供了一个配置文件用来配置语言和爬虫性能等相关信息。那么要求爬虫的配置最简化，以便能够适应每一种语言的特定任务。

(4)低消耗和高效率：首先本系统要求较低的硬件代价和高效率的抓取性能。本系统所采用的 Hadoop 分布式框架正好能够满足这样的要求。因为，Hadoop 集群的搭建不需要高代价的服务器，只需要普通的 PC 机即可；另外，在 Hadoop 框架下开发的爬虫是分布式爬取，抓取效率比较高。

(5)健壮性：一个成熟的系统必须拥有优秀的健壮性和容错能力。这里举几个例子来说。应该能够容忍用户在使用本爬虫系统时的配置错误、应该能够容忍网络环境不好和受到网路干扰时与目标服务器连接超时或数据丢失的错误、应该应对频繁抓取某主机网页时该主机封爬虫系统 IP 地址的情况，另外还有爬虫应当具备持续运行几个星期甚至几个月的能力，等等。

(6)可扩展性：对于一个分布式网络爬虫系统，它的可扩展性及其重要。应该能够较为容易地增加或者删改集群节点。关于这方面，本系统所采用的 Hadoop 分布式框架可完美地解决这一点。众所周知 Hadoop 的超级优势就是无比强大的可扩展性，没有计算节点规模的限制！

(7)爬行策略控制：对于一个分布式网络爬虫大规模爬取网页时会经常遇到很多问题，比如：频繁抓取一个主机网页时，主机会封掉爬虫系统的 IP 地址；再比如由于分布式多台机器同时对一个主机进行抓取时，主机会认为是洪水攻击而封掉爬虫系统 IP 地址，等等。这些问题在爬虫系统的设计过程中均需考虑在内。

(8)负载均衡：应尽量保持集群中每个节点爬取的网页数目相等，以充分发挥爬虫系统分布式架构的优势。关于这个要求，我们可以通过调整 Hadoop 的可调参数使得每个节点分配的任务量尽量均衡。

3.2 系统布局

由于这个分布式爬虫系统是跨语言检索系统的子课题，所以，我们有必要了解下跨语言检索系统的布局。通过这一小节的介绍，我们可以对整个系统有个宏观的认识，了解整个系统的模块划分从而确定爬虫系统在整个系统中的地位，以便更好地确定需求。另外，还明确了爬虫系统的目标及具体工作，为和

火龙果•整理
uml.org.cn

后面分布式索引模块交互提供了技术保障。

整个分布式跨语言信息获取和检索平台是以目前非常流行的开源分布式框架 Hadoop 为技术支撑。前面已经介绍 Hadoop 是一个云计算平台，包括分布式文件系统(HDFS)和分布式计算模型(Map/Reduce)两部分，用户在此平台上可以很容易开发分布式程序而不必关心其底层实现。这个分布式跨语言信息获取和检索平台系统按照功能划分可分为 5 个模块，而每一个模块都对应着 Hadoop 的一个或者多个 Map/Reduce 任务。这 5 个模块分别是：分布式采集模块(即爬虫)、分布式分析模块、分布式索引模块、分布式检索模块和用户查询模块。本文的重点是分布式采集模块的设计和实现。图 3-1 是分布式跨语言信息获取和检索平台按照功能进行的模块划分。

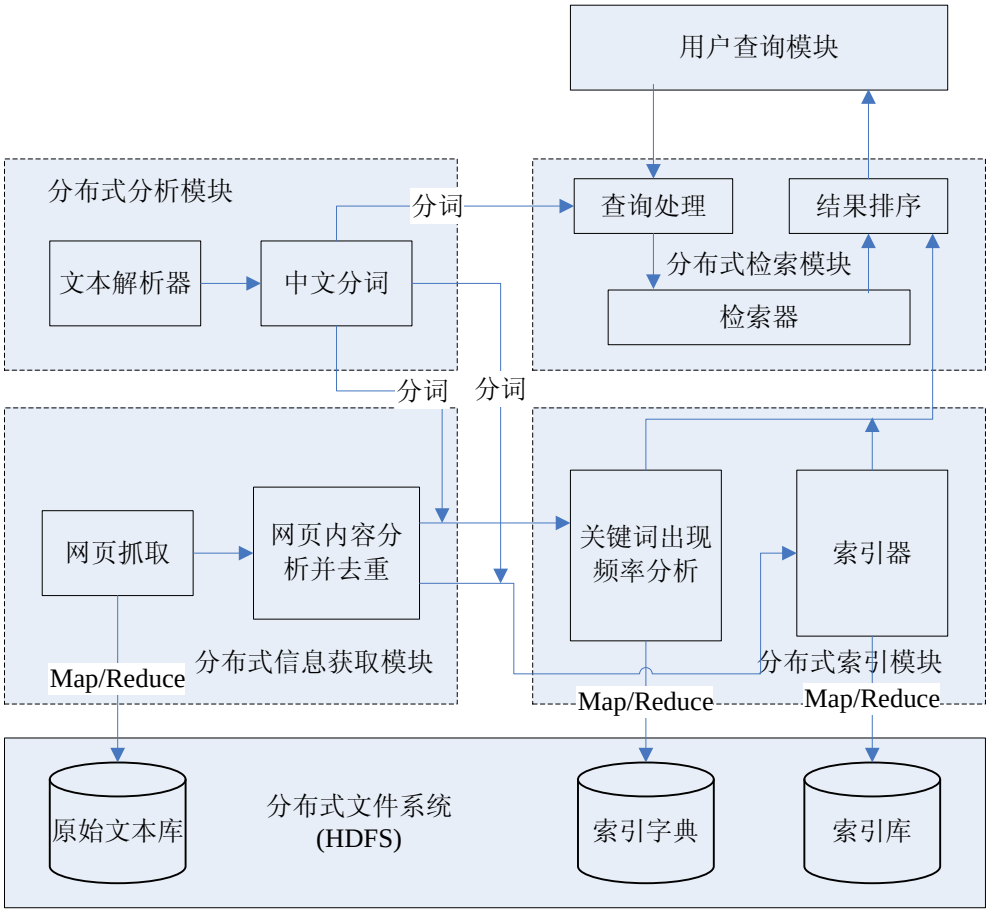

图 3-1 跨语言分布式信息采集和检索平台的功能模块划分

上图显示了 5 个功能模块之间的交互关系。首先，分布式信息获取模块负责抓取网页的工作，这部分由若干个 Map/Reduce 过程共同协作完成，而本文的重点也放在此处。抓取下来的网页经过初步的预处理被保存在分布式文件系

火龙果•整理
uml.org.cn

统(HDFS)中，构成原始文本库。其次，分布式分析模块负责对原始文本库中的网页进行分析，主要是通过文本解析器提供的分词功能来完成的。将分词处理后的结果递交给分布式索引模块，同时分析模块还会对用户提交的查询进行分析。再次，分布式索引模块负责关键词出现频率分析和创建倒排索引。关键词分析之后生成索引词典，索引器创建倒排索引之后构成索引库保存在分布式文件系统(HDFS)中，创建索引这部分也是由若干个 Map/Reduce 过程组成。另外，分布式检索模块负责去索引库中查询索引完成检索将结果数据集反馈给用户。最后，用户查询模块负责用户和搜索引擎之间的交互。用户先向分布式检索模块提交查询，检索模块将查询后的结果集合按照某种规则排好序返回给用户。

通过上面的介绍，我们了解了分布式跨语言信息获取和检索系统的功能模块划分，也明确了本文的工作重点和详细需求，即分布式信息获取模块的设计与实现。本章会重点介绍分布式爬虫系统的结构设计。

3.3 爬虫系统基本结构

确定一个分布式系统基本结构的重要依据就是节点之间的拓扑结构。根据分布式集群中各个节点之间的关系可以将拓扑结构分为四类：中心化拓扑结构、全分布式非结构化拓扑结构、全分布式结构化拓扑结构和半分布式拓扑结构。本爬虫系统采用的是中心化拓扑结构，因为基于 Hadoop 的分布式结构，一个主控节点负责任务的调度和分配，其他节点负责执行任务。

3.3.1 爬虫基本流程设计

在介绍爬虫系统基本流程之前先说明一下爬虫系统的基本需求：对于 27 种语言中的每一种语言都有一个 URL 种子集合，然后每一种语言都对应着一个配置文件用来配置这个任务的相关信息，接着把爬虫抓取深度(即抓取层数)作为参数，完成网页抓取。针对这个需求，我们给出爬虫的基本流程，参见图 3-2。

火龙果•整理
uml.org.cn

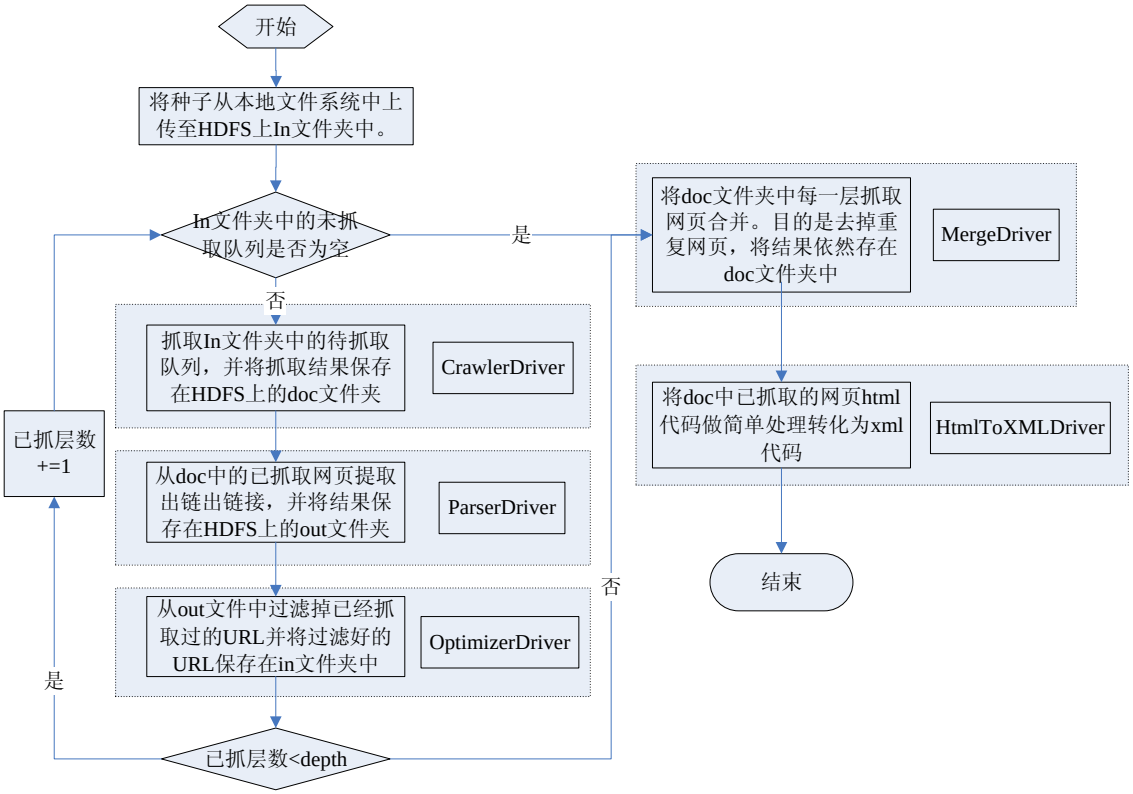

图 3-2 爬虫系统基本流程图

下面详细说明爬虫系统的基本流程图：

(1)先为每一种语言搜集一个 URL 种子集合作为下载数据的入口链接，本系统搜集的是每一种语言对应的主流媒体 URL 种子集合。然后，将一种语言对应的种子文件从本地文件系统上传至 Hadoop 集群分布式文件系统 HDFS 的 in 文件中，in 这个文件夹始终存放当前层要抓取的 URL。同时，设置已抓层数为 0。

(2)判断 in 文件夹中待抓取队列是否为空。若是，跳转到(7)；否则，执行(3)。

(3)抓取 in 文件夹中的待抓取队列。这个抓取过程是由 CrawlerDriver 模块完成的，该模块是一个基于 Hadoop 的 Map/Reduce 过程。后面我们将详细介绍 CrawlerDriver 模块的 Map/Reduce 详细实现。最后将抓取下来的网页存放在 HDFS 的 doc 文件夹中。这个 doc 文件夹存放着每一层未经加工过的网页。

(4)解析已抓取的网页，从 doc 文件夹中已抓取的网页中提取出链出链接。这个解析过程是由 ParserDriver 模块完成的，该模块也是一个 Map/Reduce 计算过程。具体抽取实现是在 Map 阶段通过 HTML 解析完成的。后面我们将详细

火龙果•整理
uml.org.cn

介绍 ParserDriver 模块的 Map/Reduce 详细实现。这个过程结束后会将抽取出来的链出链接保存在 HDFS 上的 out 文件夹。这个 out 文件夹始终存放当前层解析出来的链出链接。

(5)优化解析出来的链出链接，过滤已抓取的 URL，即从 out 文件夹中已解析出来的链出链接 URL 里过滤掉已抓过的 URL。这个优化过程时由 OptimizerDriver 模块完成的，该模块还是一个 Map/Reduce 过程。后面我们会详细介绍如何基于 Hadoop 完成 OptimizerDriver 模块的 Map/Reduce 实现。优化后会将过滤优化好的 URL 集合保存在 in 文件夹中等待下一轮的抓取。

(6)判断已抓层数是否小于 depth。如果小于，“已抓层数”自加 1，返回(2)；否则进入(7)。

(7)合并去重，将每层抓取的网页进行合并同时去掉重复抓取的网页。这个工作是由 MergeDriver 模块完成的，同样，这个模块也是一个基于 Hadoop 开发的 Map/Reduce 过程。后面我们会详细介绍如何基于 Hadoop 完成 OptimizerDriver 模块的 Map/Reduce 实现。合并后将结果依然保存在分布式文件系统 HDFS 上的 doc 文件夹中。

(8)对抓取的网页做简单的预处理。即将 html 代码转化为 xml。这个工作是由 HtmlToXmlDriver 模块完成的，同样，这个模块也是一个基于 Hadoop 开发的 Map/Reduce 过程。将处理好的 xml 文件存放在 HDFS 的 xml 文件夹中。

(9)结束。

从上面这个流程图中我们可以看出，整个爬虫系统可以分为 5 个部分，每个部分是一个独立的完成对应功能的模块，每个模块对应着一个 Map/Reduce 过程。下面介绍下这 5 个模块的功能：

(1)CrawlerDriver 模块：并行下载待抓取队列，把 in 文件夹中的文本文件作为待抓取的 URL 种子集合，该文本文件在第一轮抓取时是用户给定的初始种子，从第二轮开始就是上一轮提取出来的链出链接。该模块是基于 Hadoop 开发的一个 Map/Reduce 过程，Map 和 Reduce 分别完成了不同的功能，具体下载是在 Reduce 阶段完成的，并且采用多线程下载，下载部分是采用 Java 的网络编程完成的。下载下来的网页保存在 HDFS 上的 doc 文件夹中。

(2)ParserDriver 模块：并行分析已下载网页，提取链出链接。根据 doc 文件夹中已下载的网页分析出每一个网页中向外指向的链接即链出链接。该模块同样是基于 Hadoop 开发的 Map/Reduce 过程，但是只需要一个 Map 阶段即可完成目标。在 Map 阶段主要工作是利用 HTML 解析器解析出链出链接，另外，还通过规则限制链出 URL 的类型，防止抽取出的链接链到其他网站上。

火龙果•整理
uml.org.cn

最后将这些链出链接保存在 HDFS 上的 out 文件夹中。

(3)OptimizerDriver 模块：并行优化链出链接，过滤掉重复链接。根据 out 文件夹中已提取的链出链接，进行优化，剩下为抓取的 URL 交给下一层处理。由于网站层与层之间链接的关系是一个图的结构，所以该模块的工作可以理解成寻找环路的问题，将构成环路的 URL 过滤掉。这个模块也是一个基于 Hadoop 开发的 Map/Reduce 过程。将优化好的 URL 存放在 HDFS 上的 in 文件夹中。

(4)MergeDriver 模块：并行合并各层抓取的网页。根据 doc 文件夹中每一层抓取的网页，进行合并，去掉层与层之间可能重复的网页。这部分也是一个基于 Hadoop 开发的 Map/Reduce 过程。最后，依然将结果存放在 doc 文件夹中。

(5)HtmlToXMLDriver 模块：并行地将 HTML 转化为 XML。根据 doc 文件夹中抓取的网页，进行转化完成预处理。这部分是通过 DOM 树完成的。同样也是一个 Map/Reduce 过程。将转化后的 xml 保存在 HDFS 上的 xml 文件夹中。

这样，这 5 个功能模块就构成了一个基于 Hadoop 的分布式爬虫系统。从生成待抓取队列开始循环执行 CrawlerDriver、ParserDriver 和 OptimizerDriver 以完成各层网页抓取，跳出循环后，执行 MergeDriver 和 HtmlToXMLDriver 预处理工作。其中，循环次数是通过预设定的参数“爬取层数 depth”和“待抓取队列是否为空”来控制的。

3.3.2 爬虫系统的框架设计

基于上面解释的本爬虫系统基本流程图，下面这个分布式爬虫系统的框架设计。如图 3-3 所示。

在图 3-3 中有四个存储结构：待抓取 URL 库、原始网页库、链出 URL 库和 xml 库。这四个存储结构都是存在于 Hadoop 的分布式文件系统以 HDFS 为载体。下面详细说明这四个存储结构：

(1)待抓取 URL 库：存放当前层需要抓取的 URL 集合，实际上就是一个记录着待抓取 URL 的文本文件，其中 URL 之间以“\n”为分隔符。在第一层抓取之前，这个文本文件是用户提交的 URL 种子集合作为爬虫进入互联网的入口。

火龙果•整理
uml.org.cn

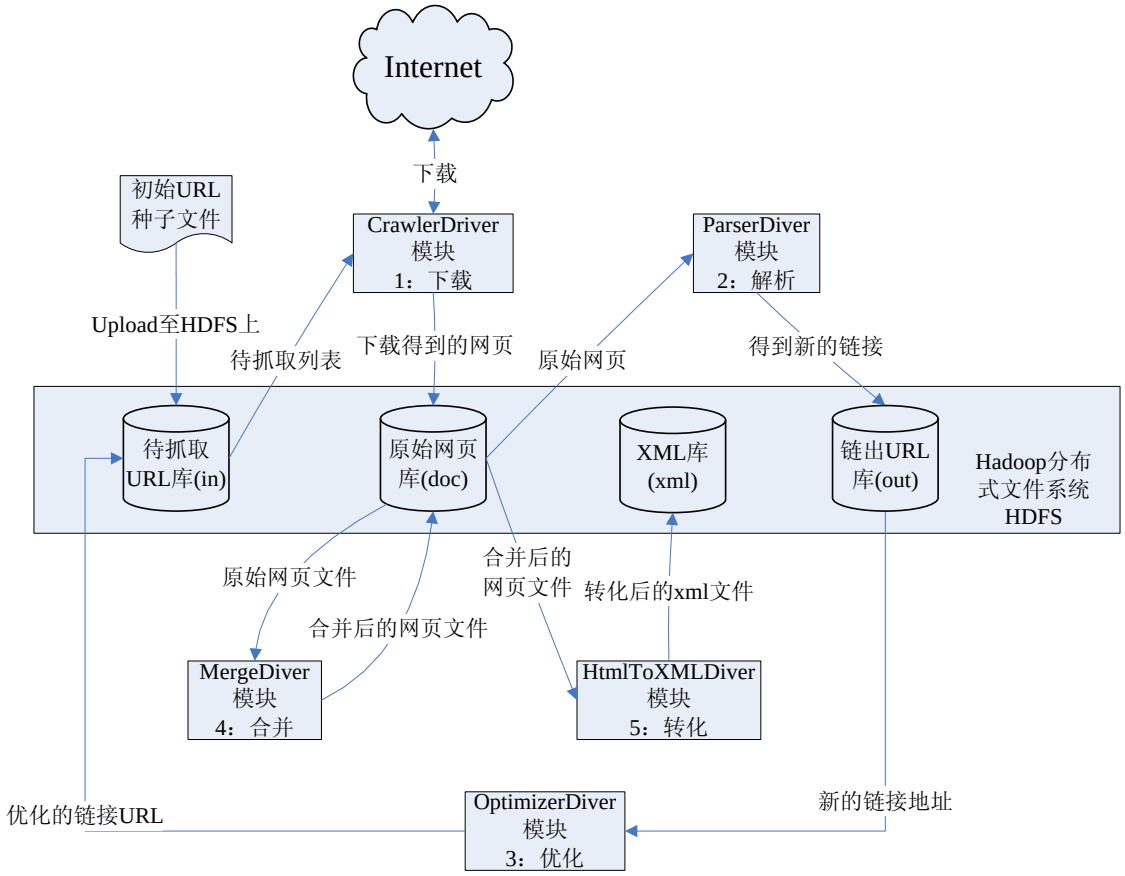

图 3-3 爬虫系统基本框架

(2)原始网页库：存放每一层抓取下来的原始网页。这里的网页是未经过任何处理的 HTML 信息，其存放形式是 key 值为 URL，value 值为 URL 对应的网页 HTML 信息。

(3)链出 URL 库：存放每一层解析出来的链出链接，其存放形式是 key 值为 URL，value 值为 URL 对应网页包含的链出链接集合。

(4)xml 库：存放所有层抓取下来的网页经过转化的 XML 信息。这里的转化相当于对 HTML 信息的预处理。其存放形式是 key 值为 URL，value 值为 URL 对应的网页的 XML 信息。

上图中 5 个功能模块分别完成不同的功能，且他们都是多台机器并行完成他们的工作，而这四个存储结构分别存储着各个功能模块生成的结果。

火龙果•整理
uml.org.cn

3.4 功能模块的 Map/Reduce 设计

同上面给出的爬虫系统的流程设计和框架设计，我们可以看出整个分布式爬虫系统包含 5 个功能模块，每一个模块都是一个基于 Hadoop 开发的 Map/Reduce 过程。下面我们将针对 CrawlerDriver、ParserDriver、OptimizerDriver、MergeDriver 和 HtmlToXMLDriver 这 5 个模块的输入、输出特点和功能需求，进行基于 Hadoop 的 Map/Reduce 设计。

3.4.1 CrawlerDriver 模块设计

该模块的功能是根据待抓取 URL 库中的种子 URL 生成待抓取队列，然后将这个队列中的 URL 分不到执行节点上完成网页抓取，并将抓取结果保存在原始网页库中。

图 3-4 表示 CrawlerDriver 模块的 Map/Reduce 时序图：

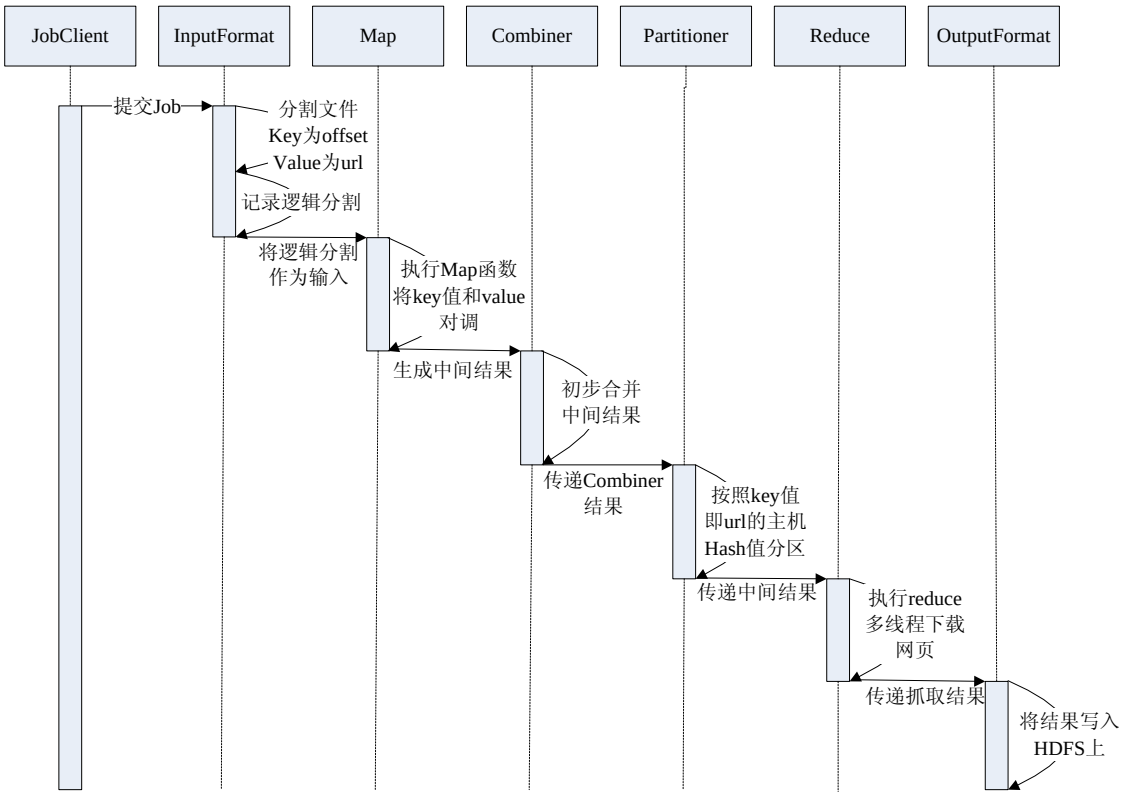

图 3-4 CrawlerDriver 模块的 Map/Reduce 时序图

火龙果•整理
uml.org.cn

依据上面这个时序图，我们可以知道 CrawlerDriver 模块的具体 Map/Reduce 执行过程：

(1)在 Hadoop 分布式集群中的客户端节点创建一个 Map/Reduce 任务，并将这个任务提交给集群。

(2)执行 InputFormat 阶段。InputFormat 是执行 Map 之前的预处理，CrawlerDriver 模块的 InputFormat 阶段主要工作如下：

1)将输入的待抓取 URL 库中的数据文件切割成小数据分片 InputSplits，一般是每 64M 为一个 Split，其中每一个 InputSplit 交给一个 Mapper 处理，这里的这个 Mapper 是一个独立执行 Map 函数的进程。

2)通过 RecordReader 接口的实现，将一个已切分的 InputSplit 解析成<key,value>对即一条记录(Record)，在 CrawlerDriver 模块中解析后的<key,value>对表示为 key 值是每一行的起始偏移量(offset)，value 值是行字符串即 URL 字符串。

3)记录逻辑分割。第一条所说的分割其实不是真正意义上的分割，而是逻辑上的分割。也就是找到输入文件的分割点，然后记录下这些分割点并把这些分割点传递给 Mapper，Mapper 再到 HDFS 上根据分割点读取自己分配得到的 Split。

(3)执行 Map 阶段。执行 Map 函数的 Mapper 类是 InverseMapper，它的输入是 InputFormat 解析出来的<offset,url>对。在 Map 函数中实现的功能是将 key、value 值互换，输出结果是 key 值 url，value 值为 offset。将这个结果写入中间文件。

(4)执行 Combiner 阶段。这个过程的作用是将每个 Map 输出的临时文件中的重复 key 做合并，如此本地化地优化可以减轻 reduce 的压力。

(5)执行 Partitioner 阶段。对中间结果分区，在 Combiner 之后，用此 Partition 函数根据 key 值将中间结果划分为 R 份，每份由一个 Reducer 负责处理。在 CrawlerDriver 模块中采用的分区算法是，计算每一个 URL 对应主机的哈希值，然后将相同主机的 URL 分到一个分区里，交给 Reduce 处理。这样做的目的是相同主机的 URL 会在一台机器上被爬取。

(6)执行 Reduce 阶段。Reduce 函数的载体是 CrawlerReducer 类，它也是一个我们实现的类。在 Reduce 函数中完成的功能是多线程地下载属于本分区的 URL，并将结果写在 HDFS 上。

(7)执行 OutputFormat 阶段。负责输出所有 Reduce 的最终结果。同 InputFormat 规定什么事 key、什么事 value 一样，OutputFormat 会约定什么是

火龙果•整理
uml.org.cn

输出的 key，什么是输出的 value。在 CrawlerDriver 模块中输出的 key 值为 URL，字符串类型；输出的 value 值为 DocumentWritable，是一个记录了网页内容和网页相关信息的复杂类型，实现了 Writable 接口。

3.4.2 ParserDriver 模块设计

该模块的功能是对原始网页库中的网页进行 HTML 解析，生成每个网页的链出链接。并将链出链接集保存在链出 URL 库中。这个模块只需要一个 Map 阶段不需要 Reduce 阶段。同时，ParserDriver 模块的输入就是 CrawlerDriver 模块的输出。

图 3-5 表示 ParserDriver 模块的 Map/Reduce 时序图：

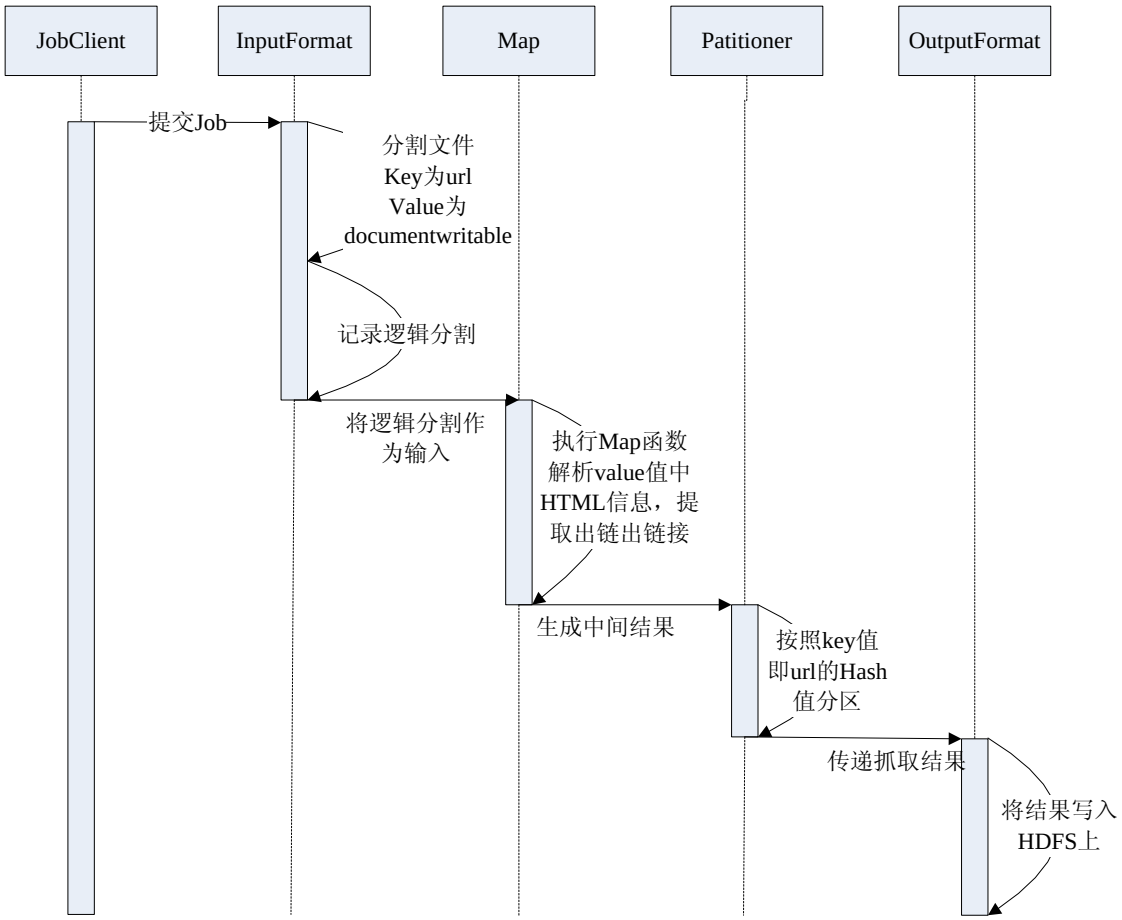

图 3-5 ParserDriver 模块的 Map/Reduce 时序图

火龙果•整理
uml.org.cn

从上图中可以看出，设计 ParserDriver 模块的 Map/Reduce 程序时不需要 Reduce 阶段，只用一个 Map 阶段即可完成任务。另外，由于没有 Reduce 阶段所以也不需要 Combiner 阶段，也就不会对中间结果中的重复 key 值做合并。下面我们通过这个时序图，来了解 ParserDriver 模块的具体 Map/Reduce 执行过程：

(1)客户端将 ParserDriver 模块的 Map/Reduce 任务提交给 Hadoop 集群，然后 Hadoop 为这个 Map/Reduce 程序创建一个任务，开始执行。

(2)执行 InputFormat 阶段。InputFormat 是执行 Map 之前的预处理，ParserDriver 模块的 InputFormat 阶段主要工作如下：

1)将输入的原始网页库中的已抓取的网页数据文件切割成小数据分片 InputSplits，同样一般是每 64M 为一个 Split，其中每一个 InputSplit 交给一个 Mapper 处理，这里的这个 Mapper 是一个独立执行 Map 函数的进程，一个节点上有一个或多个 Mapper 进程。

2)通过 RecordReader 接口的实现，将一个已切分的 InputSplit 解析成<key,value>对即一条记录(Record)，在 ParserDriver 模块中解析后的<key,value>对表示为 key 值是 url，value 值是 DocumentWritable。从解析后的输入格式可以看出 ParserDriver 模块的输入格式其实就是 CrawlerDriver 模块的输出格式。

3)记录逻辑分割。第一条所说的分割其实不是真正意义上的分割，而是逻辑上的分割。也就是找到输入文件的分割点，然后记录下这些分割点并把这些分割点传递给 Mapper，Mapper 再到 HDFS 上根据分割点读取自己分配得到的 Split。

(3)执行 Map 阶段。执行 Map 函数的 Mapper 类是我们自己实现的 ParserMapper，它的输入是 InputFormat 解析出来的<url, DocumentWritable>对。在 Map 函数中实现的功能是解析 value 值中 HTML 信息，提取出链出链接，输出结果是 key 值为 url，value 值为 OutLinksWritable，这个 OutLinksWritable 是一个实现 Writable 接口的复杂类型，它记录了从 key 值 url 对应网页 HTML 信息重抽取出来的链出链接等信息。将这个结果写入中间文件。

(4) 执行 Partitioner 阶段。对中间结果分区，在 Combiner 之后，用此 Partition 函数根据 key 值将中间结果划分为 R 份，每份由一个 Reducer 负责处理。在 ParserDriver 模块中采用的分区算法是，计算每一个 key 值的哈希值，这样相同 hash 值 key 会被分到同一个分区中保存。

(5)执行 OutputFormat 阶段。将这个 Map/Reduce 过程的结果输出，写到

火龙果•整理
uml.org.cn

HDFS 上。同时规定 key 值为 url，value 值为 OutLinksWritable。

3.4.3 OptimizerDriver 模块设计

该模块的功能是对链出 URL 库中已提取出来的 URL 进行优化，过滤掉那些在前几层已抓取过的 URL。并将优化好的链出链接集保存在待抓取 URL 库中，作为新的种子文件替换掉已经存在种子文件。同时注意新的种子文件的格式要和以前的相同。而且可以看出 OptimizerDriver 模块的输入就是 ParserDriver 模块的输入。

这个优化模块所作的工作可以看成是一个用 Map/Reduce 寻找环路的问题。把互联网中每一个超链接对应的网页想象成图中的一个结点，网页中链出的链接想象成向外指向的边。所以，优化链出链接集的工作就是过滤掉那些构成环路的链接。

图 3-6 表示 OptimizerDriver 模块的 Map/Reduce 时序图：

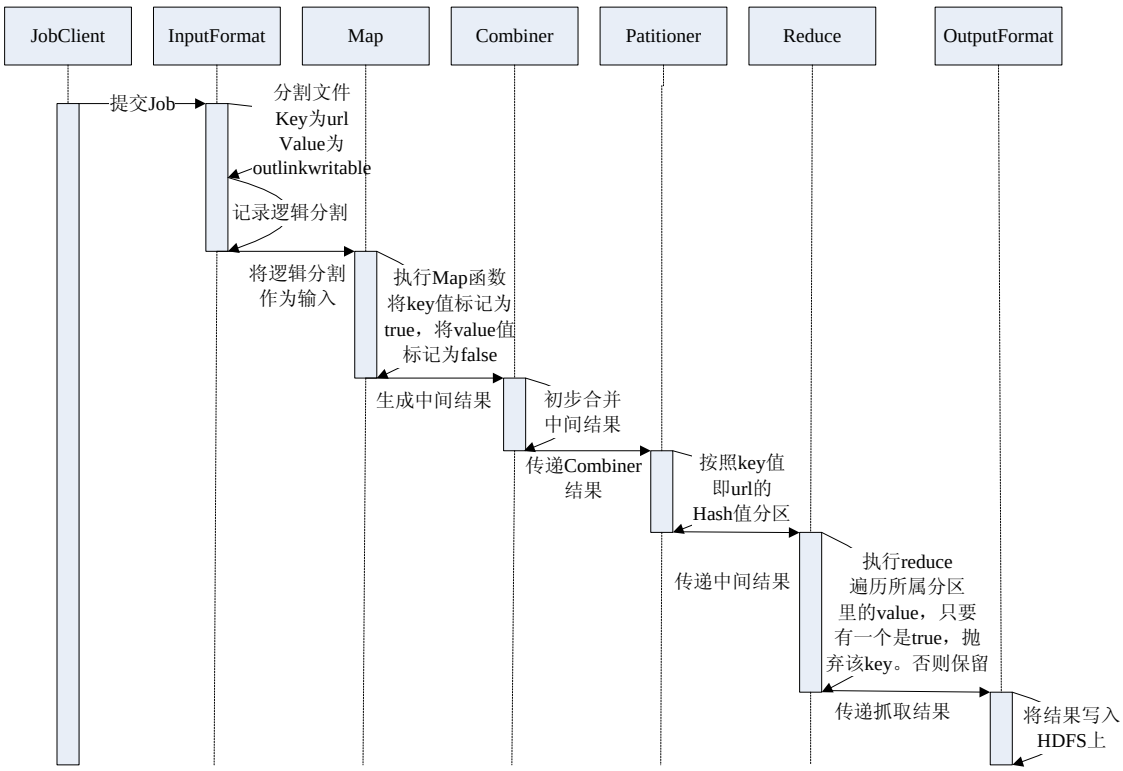

图 3-6 OptimizerDriver 模块的 Map/Reduce 时序图

火龙果•整理
uml.org.cn

依据上面这个时序图，我们可以知道 OptimizerDriver 模块的具体 Map/Reduce 执行过程：

(1) 客户端将 OptimizerDriver 模块的 Map/Reduce 程序提交给 Hadoop 集群，然后 Hadoop 为这个 Map/Reduce 程序创建一个任务，开始执行。

(2)执行 InputFormat 阶段。InputFormat 是执行 Map 之前的预处理，OptimizerDriver 模块的 InputFormat 阶段主要工作如下：

1)将输入的链出 URL 库中的已提取的链出链接数据文件切割成小数据分片 InputSplits，同样一般是每 64M 为一个 Split，其中每一个 InputSplit 交给一个 Mapper 处理，这里的这个 Mapper 是一个独立执行 Map 函数的进程，一个节点上有一个或多个 Mapper 进程。

2)通过 RecordReader 接口的实现，将一个已切分的 InputSplit 解析成<key,value>对即一条记录(Record)，在 OptimizerDriver 模块中解析后的<key,value>对表示为 key 值是 url，value 值是 OutLinksWritable。从解析后的输入格式可以看出 OptimizerDriver 模块的输入格式其实就是 ParserDriver 模块的输出格式。

3)记录逻辑分割。第一条所说的分割其实不是真正意义上的分割，而是逻辑上的分割。也就是找到输入文件的分割点，然后记录下这些分割点并把这些分割点传递给 Mapper，Mapper 再到 HDFS 上根据分割点读取自己分配得到的 Split。

(3)执行 Map 阶段。执行 Map 函数的 Mapper 类是我们自己实现的 OptimizerMapper，它的输入是 InputFormat 解析出来的<url, OutLinksWritable>对。在 Map 函数中实现的功能是将 key 值 url 标记为 true，将 OutLinksWritable 中所有的 url 标记为 false。同时，输出的中间结果是 key 值为 url，value 值为正反标记(true 或 false)。Map 结束后将中间结果写入中间文件。

(4)执行 Combiner 阶段。这个过程的作用是将每个 Map 输出的临时文件中的重复 key 做合并，如此本地化地优化可以减轻 reduce 的压力。

(5)执行 Partitioner 阶段。对中间结果分区，在 OptimizerDriver 模块中采用的分区算法是，计算每一个 key 值的哈希值，这样相同 hash 值 key 会被分到同一个分区中然后交给 reduce 阶段处理。通过此阶段之后，reduce 函数会处理相同 key 值的 value。

(6)执行 Reduce 阶段。Reduce 函数的载体是我们自己实现的 OptimizerReducer 类。在 Reduce 函数中完成的功能是遍历所属分区里的 value，即遍历相同 url 的所有标记集合。如果有一个标记是 true，抛弃该 url；

火龙果•整理
uml.org.cn

相反如果所有标记都为 false，则把这个 url 作为最终结果写到 HDFS 上。其中，url 与 url 之间用“\n”分隔，因为要与以前的 url 种子文件格式相同。

(7)执行 OutputFormat 阶段。将这个 Map/Reduce 过程的结果输出，写到 HDFS 上。同时规定 key 值为每一行起始偏移量即 offset，value 值为 url。可以看出 OptimizerDriver 模块的输出格式是 CrawlerDriver 模式的输入格式。

3.4.4 MergeDriver 模块设计

本爬虫系统采用的是广度优先搜索策略去抓取网页，CrawlerDriver、ParserDriver 和 OptimizerDriver 这三个模块构成一层完整的抓取过程。从上面给出的流程图可以看出这三个模块循环执行，每抓取一层后会在原始网页库中生成若干个分区文件，分区文件的个数可表示为 depth*r，其中 depth 为拟爬取的层数，r 为 Hadoop 集群中 Reduce 的个数。MergeDriver 模块的功能就是对原始网页库中每一层抓取下来的网页数据文件进行合并，将数据文件的个数由 depth*r 合并成 r 个。这样做的目的是一方面可以便于原始网页库的管理，另一方面最重要的目的是去重工作。

OptimizerDriver 模块的优化并不能达到百分百无重复的目的，因为它并不能过滤掉那些能够产生重定向的 URL，下面举个例子来说明这个问题，我们知道 http://www.xiao.com 和 http://www.renren.com 这两个 URL 指向的是同一个网页，即前者会重定向成后者，但是在 OptimizerDriver 模块中并不能区分这两个 URL，从而这两个 URL 指向的共同网页会被下载两次。所以在前三个模块完成每一层爬行任务之后，运行 MergeDriver 模块以达到去重的目的。MergeDriver 模块的输入是原始网页库。

图 3-7 表示 MergeDriver 模块的 Map/Reduce 时序图：

依据图 3-7 这个时序图，我们可以知道 MergeDriver 模块的具体 Map/Reduce 执行过程：

(1) 客户端将 MergeDriver 模块的 Map/Reduce 程序提交给 Hadoop 集群，然后 Hadoop 为这个 Map/Reduce 程序创建一个任务，开始执行。

(2)执行 InputFormat 阶段。InputFormat 是执行 Map 之前的预处理，MergeDriver 模块的 InputFormat 阶段主要工作如下：

1)将输入的原始网页库中的各层已下载的网页数据文件切割成小数据分片 InputSplits，同样一般是每 64M 为一个 Split，其中每一个 InputSplit 交给一个 Mapper 处理，这里的这个 Mapper 是一个独立执行 Map 函数的进程，一个节

火龙果•整理
uml.org.cn

点上有一个或多个 Mapper 进程。

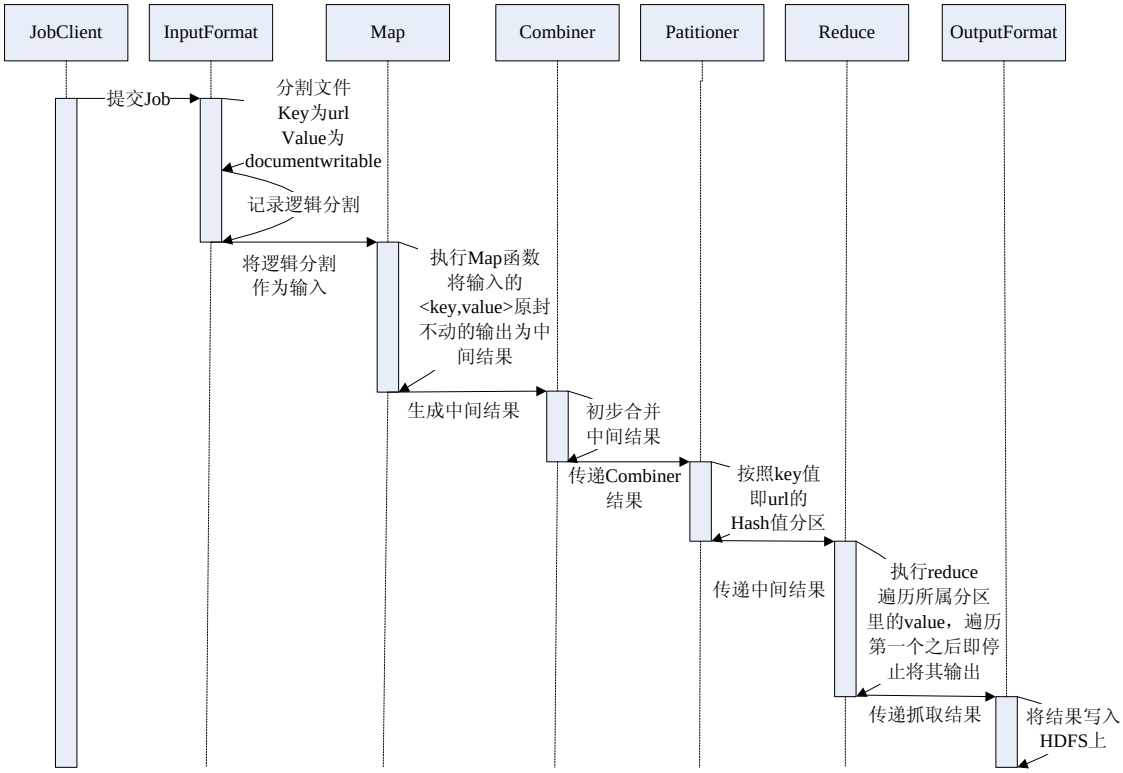

图 3-7 MergeDriver 模块的 Map/Reduce 时序图

2) 通过 RecordReader 接口的实现，将一个已切分的 InputSplit 解析成<key,value>对即一条记录(Record)，在 MergeDriver 模块中解析后的<key,value>对表示为 key 值是 url，value 值是 DocumentWritable。从解析后的输入格式可以看出 MergeDriver 模块的输入格式其实就是 CrawlerDriver 模块的输出格式。

3)记录逻辑分割。第一条所说的分割其实不是真正意义上的分割，而是逻辑上的分割。也就是找到输入文件的分割点，然后记录下这些分割点并把这些分割点传递给 Mapper，Mapper 再到 HDFS 上根据分割点读取自己分配得到的 Split。

(3)执行 Map 阶段。执行 Map 函数的 Mapper 类是我们自己实现的 MergeMapper，它的输入是 InputFormat 解析出来的<url, DocumentWritable >对。在 Map 函数中实现的功能是将输入的<key,value>对原封不动地作为为中间结果输出。Map 结束后将中间结果写入中间文件。

(4)执行 Combiner 阶段。这个过程的作用是将每个 Map 输出的临时文件中

火龙果•整理
uml.org.cn

的重复 key 做合并，如此本地化地优化可以减轻 reduce 的压力。

(5)执行 Partitioner 阶段。对中间结果分区，在 MergeDriver 模块中采用的分区算法是，计算每一个 key 值的哈希值，这样相同 hash 值 key 会被分到同一个分区中然后交给 reduce 阶段处理。通过此阶段之后，reduce 函数会处理相同 key 值的 value。

(6)执行 Reduce 阶段。Reduce 函数的载体是我们自己实现的 MergeReducer 类。在 Reduce 函数中完成的功能是遍历所属分区里的 value，即遍历相同 url 的所有网页信息(DocumentWritable)集合，当遍历第一个之后就停止遍历，url 仍作为 key 值，而第一个网页信息作为 value 值。将 key 值和 value 值作为最终结果输出到 HDFS 上。

(7)执行 OutputFormat 阶段。将这个 Map/Reduce 过程的结果输出，写到 HDFS 上。同时规定 key 值 url，value 值为 DocumentWritable。可以看出 MergeDriver 模块的输出格式与 CrawlerDriver 模式的输出格式一样，并未改变。

3.4.5 HtmlToXMLDriver 模块设计

该模块的功能是对原始网页库中的已采集的网页 HTML 信息进行简单的预处理，将其转化为 XML 信息。将预处理后的 XML 信息保存在 XML 库中。这个模块只需要一个 Map 阶段不需要 Reduce 阶段。也就是说 HTML 向 XML 转化的操作是在 Map 函数里完成的。

设计这个模块的意义是对网页 HTML 信息预处理，增加抓取时间，字符编码等附加信息。这样充分利用 XML 半结构化特性有效地管理原始网页库中的数据，同时方便了后面创建索引等模块。

图 3-8 表示 HtmlToXmlDriver 模块的 Map/Reduce 时序图：

从上图中可以看出 HtmlToXmlDriver 模块不需要 Reduce 阶段，一个 Map 阶段即可完成所需功能。另外，由于没有 Reduce 阶段所以也不需要 Combiner 阶段，也就不会对中间结果中的重复 key 值做合并。下面我们通过这个时序图，来了解 HtmlToXmlDriver 模块的具体 Map/Reduce 执行过程：

(1)客户端将 HtmlToXmlDriver 模块的 Map/Reduce 任务提交给 Hadoop 集群，然后 Hadoop 为这个 Map/Reduce 程序创建一个任务，开始执行。

(2)执行 InputFormat 阶段。InputFormat 是执行 Map 之前的预处理，HtmlToXmlDriver 模块的 InputFormat 阶段主要工作如下：

火龙果•整理
uml.org.cn

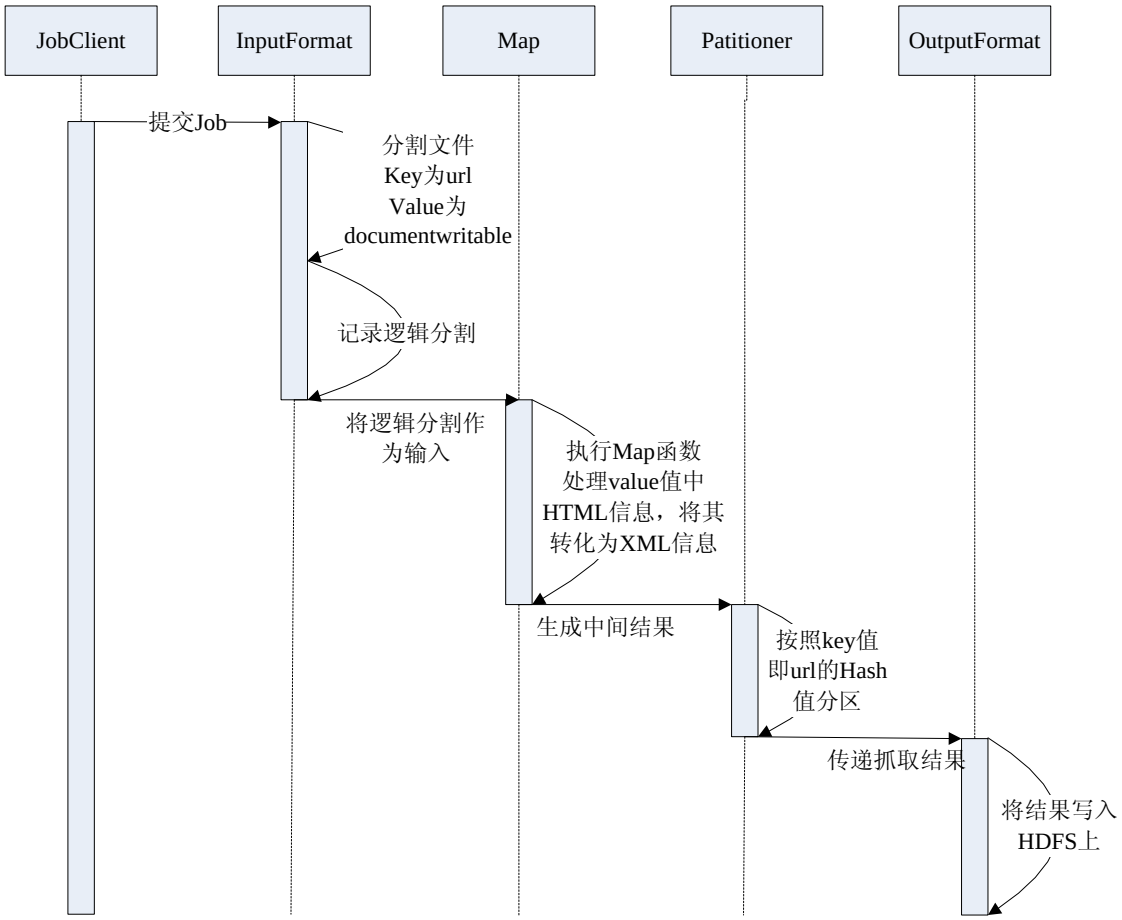

图 3-8 HtmlToXmlDriver 模块的 Map/Reduce 时序图

1)将输入的原始网页库中的每一层已抓取的网页数据文件切割成小数据分片 InputSplits，同样一般是每 64M 为一个 Split，其中每一个 InputSplit 交给一个 Mapper 处理，这里的这个 Mapper 是一个独立执行 Map 函数的进程，一个节点上有一个或多个 Mapper 进程。

2)通过 RecordReader 接口的实现，将一个已切分的 InputSplit 解析成<key,value>对即一条记录(Record)，在 HtmlToXmlDriver 模块中解析后的<key,value>对表示为 key 值是 url，value 值是 DocumentWritable。从解析后的输入格式可以看出 HtmlToXmlDriver 模块的输入格式其实就是 CrawlerDriver 模块的输出格式。

3)记录逻辑分割。第一条所说的分割其实不是真正意义上的分割，而是逻辑上的分割。也就是找到输入文件的分割点，然后记录下这些分割点并把这些

火龙果•整理
uml.org.cn

分割点传递给 Mapper，Mapper 再到 HDFS 上根据分割点读取自己分配得到的 Split。

(3)执行 Map 阶段。执行 Map 函数的 Mapper 类是我们自己实现的 HtmlToXmlMapper，它的输入是 InputFormat 解析出来的 <url, DocumentWritable>对。在 Map 函数中实现的功能是解析 value 值中 HTML 信息，将 HTML 转换成 XML 信息，增加抓取时间、字节编码等附加信息，输出结果是 key 值为 url，value 值为 XML。将这个最终结果写入中间文件。

(4) 执行 Partitioner 阶段。对中间结果分区，在 Combiner 之后，用此 Partition 函数根据 key 值将中间结果划分为 R 份，每份由一个 Reducer 负责处理。在 HtmlToXmlDriver 模块中采用的分区算法是，计算每一个 key 值的哈希值，这样相同 hash 值 key 会被分到同一个分区中保存。

(5)执行 OutputFormat 阶段。将这个 Map/Reduce 过程的结果输出，写到 HDFS 上。同时规定 key 值为 url，value 值为 url 对应网页的 XML 信息。

3.5 本章小结

本章是基于 Hadoop 分布式网络爬虫系统的概要设计，为系统的实现提供设计框架和实施方案。

首先本章阐述了本爬虫系统的设计需求，明确本爬虫系统目的。然后，描述跨语言检索系统的系统布局，对其进行模块划分，确定本爬虫系统在跨语言检索系统中的位置，明确本文所需作的工作。接下来，设计本爬虫系统的基本结构，包括流程设计和框架设计。最后，对爬虫系统中的各个模块进行 Map/Reduce 设计。

火龙果•整理
uml.org.cn

第4章 分布式爬虫具体实现

本章论述的是基于 Hadoop 分布式爬虫系统的详细设计和具体实现。依据上一章介绍的结构设计，详细论述 5 个功能模块的主要实现方法。其中，还包括各模块的数据结构、数据格式、具体代码的编写等等。

4.1 存储结构的实现

依据上面的分布式爬虫系统框架的介绍，我们可以看出本系统共用到 4 个存储结构，分别是：待抓取 URL 库、原始网页库、链出 URL 库和 XML 库。这四个存储结构是依托于 Hadoop 的分布式文件系统(HDFS)之上。下面我们将详细介绍这四个存储结构的实现。

4.1.1 待抓取 URL 库

待抓取 URL 库是存放当前预抓取的 URL 种子文件，这个存储库的数据结构比较简单，是 HDFS 上的文本文件，表 4-1 是它的说明：

表 4-1 待抓取 URL 库数据结构

名称	变量	数据类型	说明
网页的超链接	url	String	网页的 url

其中 URL 之间以“\n”为分隔符。

4.1.2 原始网页库

原始网页库是 HDFS 上保存每一层抓取下来的原始网页的存储结构。表 4-2 是这个存储库的数据结构说明：

在这个存储库中 key 值为 url_redirect，value 值为 DocumentWritable 类型。其中，DocumentWritable 是一个包含 document、redirectFrom、metaFollow 和 metaIndex 信息的复杂类型。采用 Java 的持久化将 DocumentWritable 保存在分布式文件系统中。

火龙果•整理
uml.org.cn

表 4-2 原始网页库的数据结构

名称	变量	数据类型	说明
网页的超链接	url_redirect	String	网页的真实 url，如果发生重定向，url_redirect 为重定向后的 url，否则 url_redirect 为原 url。
网页信息	document	String	网页的 HTML 信息，即原始数据
重定向源头	redirectFrom	String	网页的原 url。如果未发生重定向，redirectFrom ==url_redirect
网页 Meta 中的 Follow 信息	metaFollow	Boolean	如果网页 meta 信息中的 content 参数中有 follow 则 metaFollow=true，否则 metaFollow=false
网页 Meta 中的 Index 信息	metaIndex	Boolean	如果网页 meta 信息中的 content 参数中含有 index 则 metaIndex =true；相反含有 noindex，则 metaIndex=false

4.1.3 链出 URL 库

链出 URL 库是 HDFS 上保存每一层解析出来的链出链接。下面是这个存储库的数据结构说明：

表 4-3 链出 URL 库数据结构

名称	变量	数据类型	说明
网页的超链接	url	String	网页的 url，如果发生重定向，url 为原始 url，否则 url 为重定向后的真是 url。
解析出来的链出链接集合	outlinks	String[]	一个存放 url 的数组。如果未发生重定向则 outlinks 为从网页中解析出来的链出链接集合；否则，outlinks 只有一个元素，为重定向后的真实 url。
链出链接类型	typeOfOutlink	Int	标记 outlinks 数组里的 url 类型。如果是链出链接集合则 typeOfOutlink 为 0；如果是真实 url 则 typeOfOutlink 为 1。
时间戳	timeStamp	Long	由于后三个变量构成一个复杂类型 OutLinksWritable。每当链出 URL 库中写入一个 OutLinksWritable 类型的数据时，会为这个数据生成一个时间戳，用来区分不同的数据。

在这个存储库中 key 值为 url，value 值为 OutLinksWritable 类型。其中，OutLinksWritable 是一个包含了 outlinks、typeOfOutlink 和 timeStamp 信息的复杂类型。采用 Java 的持久化将 DocumentWritable 保存在分布式文件系统中。

火龙果•整理
uml.org.cn

4.1.4 XML 库

XML 库是 HDFS 上保存网页 XML 信息的存储结构。下面是这个存储库的数据结构说明：

表 4-4 XML 库数据结构

名称	变量	数据类型	说明
网页的超链接	url	String	网页的真实 url，如果发生重定向，url_redirect 为重定向后的 url，否则 url_redirect 为原 url。
网页 xml 信息	xml	String	网页 xml 信息

在这个存储库中 key 值为 url，value 值为 xml 信息。其中 xml 中包含的信息为：url、抓取时间、存储目录、头条、正文。url 为链接地址。抓取时间是网页抓取的时间。存储目录是网页在 XML 库中存放的目录，每一种语言是一个目录。由于本爬虫系统主要是抓取新闻网站，所以头条表示新闻头条。正文是网页提取出的正文。

4.2 爬虫总体数据结构

本爬虫系统是在 Hadoop 的 API 基础上进行开发的。按照类的功能来划分，爬虫系统的总体数据结构可划分为主类和功能类。其中，主类为 Crawler；功能类包括 CrawlerDriver、ParserDriver、OptimizerDriver、MergeDriver 和 HtmlToXmlDriver。图 4-1 是爬虫总体数据结构的逻辑视图：

火龙果•整理
uml.org.cn

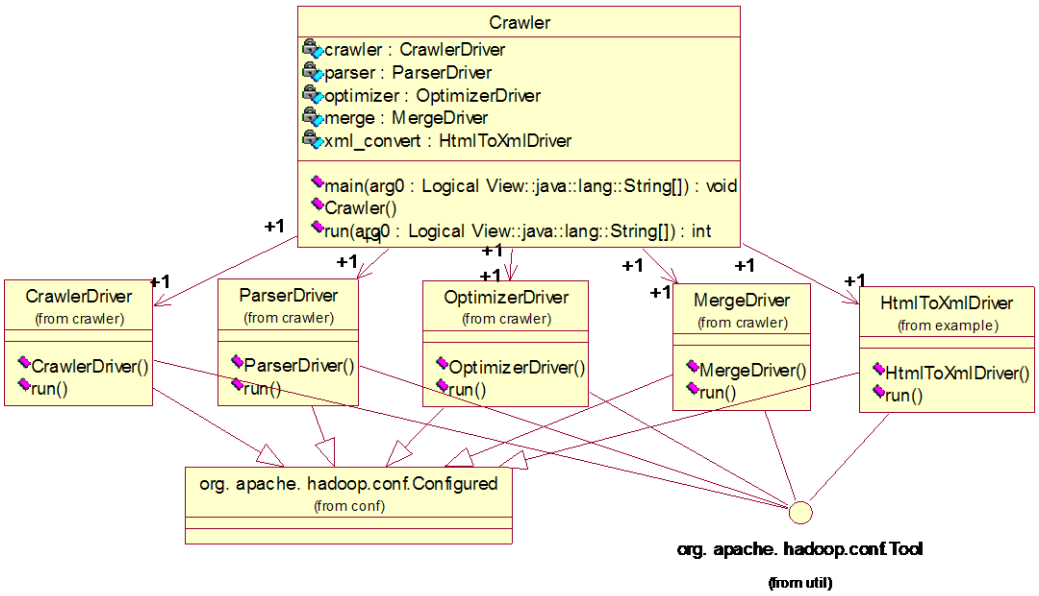

图 4-1 总体数据结构逻辑视图

(1)Crawler 类：是爬虫系统的主类，即整个系统的 main 入口类。Crawler 主类以一个 String 数组作为输入参数，数组中拥有 3 个成员：seeds，depth，conf_file。如前所述每种语言有一个初始的种子文件，seeds 表示爬虫系统初始种子文件的名字；depth 表示爬虫系统抓取的深度；前面我们已经介绍每种语言都有一个配置文件，conf_file 表示配置文件的名字。Crawler 类共有 5 个私有成员，这 5 个成员分别是 CrawlerDriver 类型、ParserDriver 类型、OptimizerDriver 类型、MergeDriver 类型和 HtmlToXmlDriver 类型。主类通过调用 5 个功能类来控制整个爬虫系统的运行过程。

(2)CrawlerDriver 类：功能类。完成网页的抓取工作。

(3)ParserDriver 类：功能类。完成网页分析功能，提取链出链接。

(4)OptimizerDriver 类：功能类。完成链出链接优化功能，从上一步提取出来的链接集合中过滤掉已经爬过链接。

(5)MergeDriver 类：功能类。完成网页合并功能，将每一层抓取的 HDFS 网页数据文件合并，同时达到去重的功能。

(6)HtmlToXMLDriver 类：功能类。完成网页的预处功能，将已抓取网页的 HTML 信息转化为 XML 信息。

这 6 个类构成整个基于 Hadoop 分布式爬虫系统的总体框架。其中，5 个

火龙果•整理
uml.org.cn

功能类都是基于 Hadoop 开发的 Map/Reduce 应用程序，它们继承了 Hadoop 的 Configured 类并实现了 Tool 接口，用来配置和启动 Map/Reduce 作业。另外，这 5 个功能类要完成它们自己相应的功能还需要若干个辅助类协同工作。后面介绍各个功能模块的实现时，我们会详细分析。

4.3 功能模块的实现

本系统共分为 5 个模块：CrawlerDriver、ParserDriver、OptimizerDriver、MergeDriver 和 HtmlToXMLDriver，每个模块完成不同的功能，它们组合到一起组成整个系统完成爬取网页的任务。下面介绍这 5 个功能模块的详细设计：

4.3.1 CrawlerDriver 模块

CrawlerDriver 模块完成网页抓取的工作。该模块可划分为：功能类、辅助类、Map/Reduce 核心类和 Key/Value 存储类型。其中，功能类为 CrawlerDriver；辅助类包括 Downloader、HostPartitioner、HostShuffle、MetaParser；Map/Reduce 核心类包括 InverseMapper 和 CrawlerReducer；Key/Value 存储类型包括 Text、LongWritable 和 DocumentWritable。下面图 4-2 是 CrawlerDriver 模块组成类的逻辑视图。

首先来介绍功能类：

CrawlerDriver 类：功能是配置和启动 Map/Reduce 程序。在配置时主要完成的工作是设置执行 map 函数的类、设置执行 reduce 函数的类、设置执行分区 Partition 的类、设置 Key/Value 对的类。

其次来介绍辅助类：

(1) Downloader 类：对于一个输入的 URL 完成该 URL 对应的网页的下载。

(2) HostPartitioner 类：在做 reduce 阶段之前按照主机哈希值进行分区，保证具有同一主机名的 url 被分到一个 reduce 中处理。

(3) HostShuffle 类：由于一个执行 reduce 函数的类可能接收到不同主机的 URL，所以为了避免连续爬取一个主机的网页，该类将 URL 队列重新洗牌，保证相同主机的 url 尽量间隔。

(4) MetaParser 类：对于输入的网页 HTML 信息，分析 meta 信息中的 content 参数是否含有 follow/nofollow 和 index/noindex。

火龙果•整理
uml.org.cn

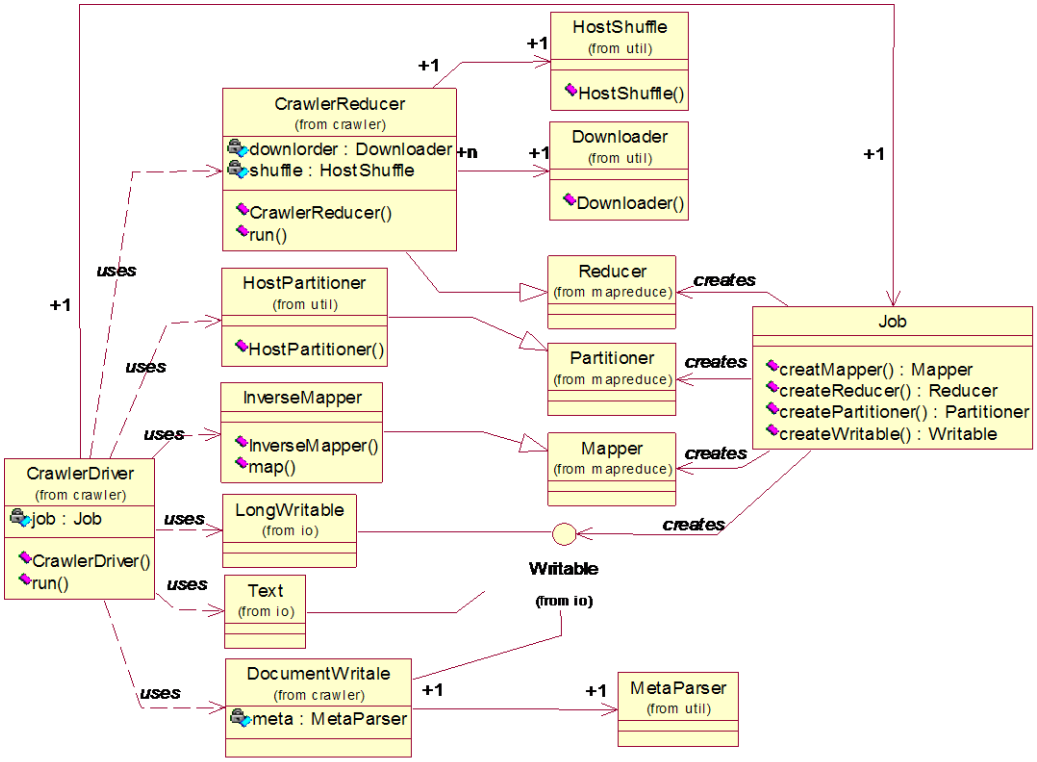

图 4-2 CrawlerDriver 模块组成类的逻辑视图

再次来介绍 Map/Reduce 核心类：

(1) InverseMapper 类：执行 map 函数的类，将 key、value 值互换。这个 map 函数输出的中间格式类型为<Text，LongWritable>，值为<url，offset>。

(2)CrawlerReducer 类：执行 reduce 阶段的类，在该类的 run 函数里完成 Downloader 的多线程下载和 HostShuffle 重排列 url 队列的功能。另外，这个类还有一个 URL 规则限制的操作，即前面我们所说的 url 必须满足配置文件中的正则表达式，以防止抓取过程抓到其他的网站上去。

最后来介绍 Key/Value 存储类型：

(1)Text 类型：字符串 String 的可序列化类型。

(2)LongWritable 类型：长整形 long 的可序列化类型。

(3)DocumentWritable 类型：包含网页 HTML 等相关信息的复杂类型。在创建这个类型的对象时，会调用 MetaParser 类型，记录网页 Meta 信息分析。

对于 Map 函数，输入时它的 key 值为 LongWritable，value 值为 Text；输出时它的 key 值为 Text，value 为 LongWritable。对于 Reduce 函数，输入时它

火龙果•整理
uml.org.cn

的 key 值为 Text，value 值为 LongWritable；输出时它的 key 值为 Text，value 值为 DocumentWritable。

下面详细介绍这些类之间的关系，通过上图 4-2

(1)CrawlerDriver 类有一个成员 Job 类型，在它的 run 函数中通过 Job 配置执行 Map 阶段、Partition 阶段和 Reduce 阶段的类型，另外还要配置每一个阶段 Key/Value 的类型。

(2)启动 Map/Reduce 作业，这时会将这些配置信息提交给 Hadoop 的底层，在类图中我们把底层全部抽象成 Job 类型，这样做是为了专注于本系统实现类的关系。

(3)Job 类通过反射机制创建 InverseMapper、CrawlerReducer、HostPartitioner、Text、LongWritable 和 DocumentWritable 对象。

(4)CrawlerDriver 通过是通过使用这些对象完成 CrawlerDriver 模块的抓取功能。

4.3.2 ParserDriver 模块

ParserDriver 模块完成网页分析工作，从 CrawlerDriver 模块下载的网页 HTML 信息中分析出链出链接。该模块可划分为：功能类、辅助类、Map/Reduce 核心类和 Key/Value 存储类型。其中，功能类包括 ParserDriver；辅助类包括 Parser；Map/Reduce 核心类包括 ParserMapper；Key/Value 存储类型包括 Text、DocumentWritable 和 OutLinksWritable。ParserDriver 模块没有 reduce 阶段，只有一个 Map 阶段。下面图 4-3 是 ParserDriver 模块组成类的逻辑视图。

首先来介绍功能类：

ParserDriver 类：配置和启动 Map/Reduce 程序。在配置时主要完成的工作是设置执行 map 函数的类和设置 Key/Value 对的类。这个 map 函数输出的中间格式类型为<Text，OutlinkWritable>，值为<url，链出链接集合>。

其次来介绍辅助类：

Parser 类：对于输入的网页 HTML 信息，抽取出链出链接集合，返回一个 URL 数组。

再次来介绍 Map/Reduce 核心类：

ParserMapper 类：执行 map 阶段的类，在该类的 map 函数里，调用 Parser 类，完成抽取链出链接的功能。

火龙果•整理
uml.org.cn

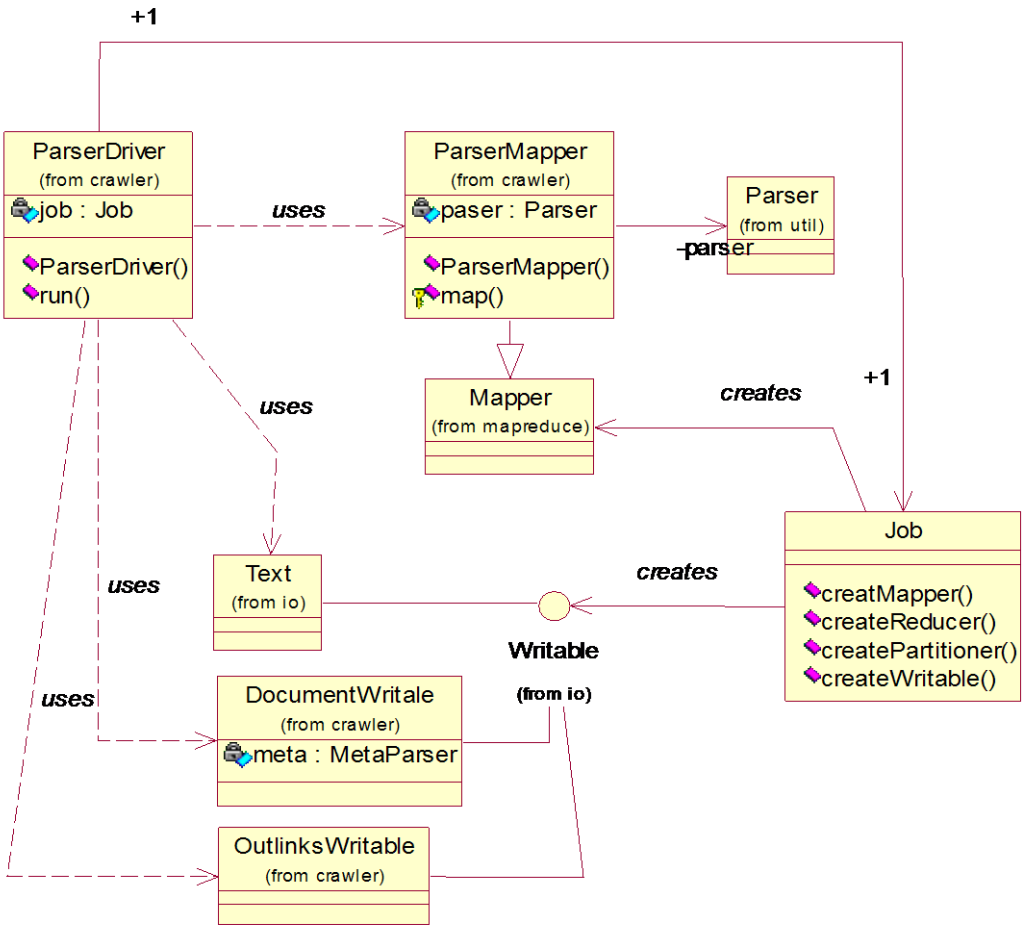

图 4-3 ParserDriver 模块组成类的逻辑视图

最后来介绍 Key/Value 存储类型：

(1)Text 类型：字符串 String 的可序列化类型。

(2) DocumentWritable 类型：包含网页 HTML 等相关信息的复杂类型，具体信息在前面存储结构的实现中已经介绍了。在创建这个类型的对象时，会调用 MetaParser 类型，记录网页 Meta 信息分析。

(3) OutLinksWritable 类：包含链出链接数组等相关信息的复杂类型，具体信息在前面存储结构的实现中已经介绍了。

对于 Map 函数，输入时它的 key 值为 Text，value 值为 DocumentWritable；输出时它的 key 值为 Text，value 为 OutLinksWritable。

下面详细介绍这些类之间的关系，通过上图 4-3

火龙果•整理
uml.org.cn

(1)ParserDriver 类有一个成员 Job 类型，在它的 run 函数中通过 Job 配置执行 Map 阶段的类型，另外还要配置每一个阶段 Key/Value 的类型。

(2)启动 Map/Reduce 作业，这时会将这些配置信息提交给 Hadoop 的底层，在类图中我们把底层全部抽象成 Job 类型，这样做是为了专注于本系统实现类的关系。

(3)Job 类通过反射机制创建 ParserMapper、Text、DocumentWritable 和 OutLinksWritable 对象。

(4) ParserDriver 通过是通过使用这些对象完成 ParserDriver 模块的抽取链出链接功能。

4.3.3 OptimizerDriver 模块

OptimizerDriver 模块完成链出链接优化工作，对 ParserDriver 模块解析出来的链出链接进行优化，过滤掉那些在前几层已经爬过的 URL，并将优化好的 URL 集合交给 CrawlerDriver 模块开始下一层的抓取。该模块可划分为：功能类、辅助类、Map/Reduce 核心类和 Key/Value 存储类型。其中，功能类包括 OptimizerDriver；辅助类包括 HashPartitioner；Map/Reduce 核心类包括 OptimizerMapper 和 OptimizerReducer；Key/Value 存储类型包括 Text 和 BooleanWritable。

图 4-4 是 OptimizerDriver 模块组成类的逻辑视图。

首先来介绍功能类：

OptimizerDriver 类：配置和启动 Map/Reduce 程序。在配置时主要完成的工作是设置执行 map 函数的类和设置 Key/Value 对的类。

其次来介绍辅助类：

HashPartitioner 类：在做 reduce 阶段之前按照 key(即 url)的 Hash 进行分区，保证相同 url 的 value 值被分到一个 reduce 中处理。

再次来介绍 Map/Reduce 核心类：

(1) OptimizerMapper 类：执行 map 函数的类，给 key(即 url) 赋值为 true，给 value(即 OutlinksWritable)里的所有 url 赋值 false。这个 map 函数输出的中间格式类型为<Text，BooleanWritable>，值为<url，true/false>。

(2) OptimizerReducer 类：执行 reduce 函数的类，遍历所属分区里的 value，即遍历相同 url 的所有标记集合。只要有一个标记是 true，抛弃该 url；相反如果所有标记都为 false，则把这个 url 作为最终结果写到 HDFS 上。

火龙果•整理
uml.org.cn

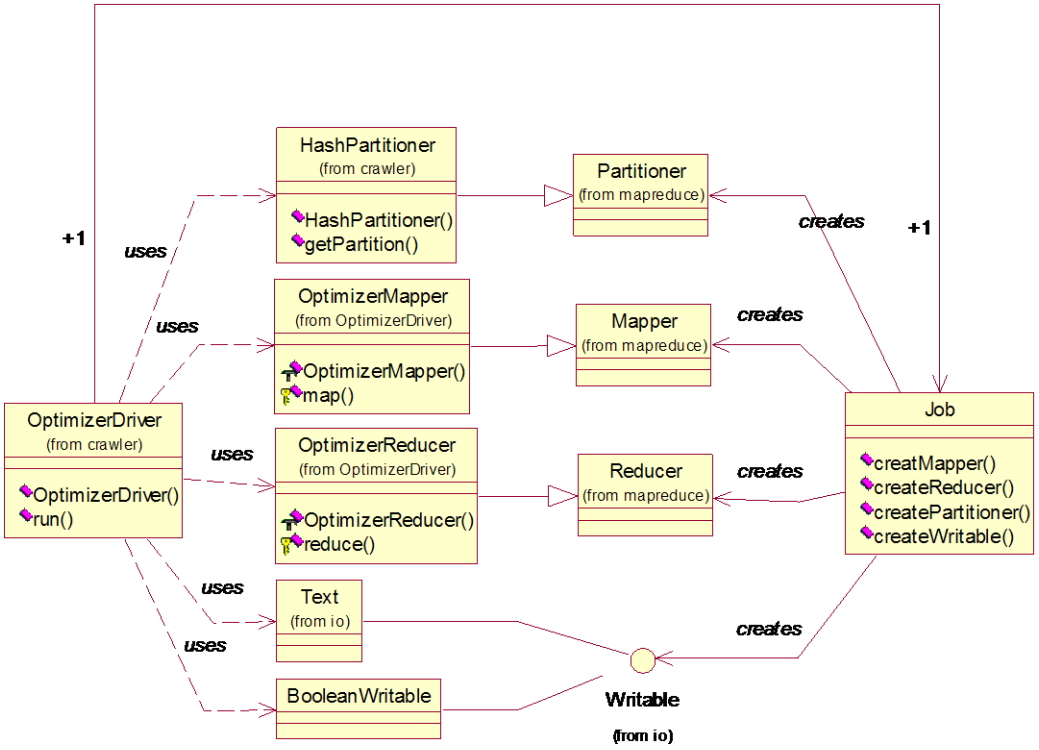

图 4-4 OptimizerDriver 模块组成类的逻辑视图

最后来介绍 Key/Value 存储类型：

(1)Text 类型：字符串 String 的可序列化类型。

(2) BooleanWritable 类型：布尔类型 Boolean 的可序列化类型。

对于 Map 函数，输入时它的 key 值为 Text，value 值为 OutlinksWritable；输出时它的 key 值为 Text，value 为 BooleanWritable。对于 Reduce 函数，输入时它的 key 值为 Text，value 值为 BooleanWritable；输出时它的 key 值为 LongWritable，value 值为 Text。

下面详细介绍这些类之间的关系，通过上图 4-4

(1) OptimizerDriver 类有一个成员 Job 类型，在它的 run 函数中通过 Job 配置执行 Map 阶段、Partition 阶段和 Reduce 阶段的类型，另外还要配置每一个阶段 Key/Value 的类型。

(2)启动 Map/Reduce 作业，这时会将这些配置信息提交给 Hadoop 的底层，在类图中我们把底层全部抽象成 Job 类型，这样做是为了专注于本系统实

火龙果•整理
uml.org.cn

现类的关系。

(3)Job 类通过反射机制创建 OptimizerMapper、OptimizerReducer、HashPartitioner、Text 和 BooleanWritable 对象。

(4)OptimizerDriver 通过是通过使用这些对象完成 OptimizerDriver 模块的优化功能。

4.3.4 MergeDriver 模块

MergeDriver 模块完成网页的合并工作。该模块可划分为：功能类、辅助类、Map/Reduce 核心类和 Key/Value 存储类型。其中，功能类包括 MergeDriver；辅助类包括 HashPartitioner；Map/Reduce 核心类包括 IdentityMapper 和 MergeDocReducer；Key/Value 存储类型包括 Text 和 DocumentWritable。图 4-5 是 MergeDriver 模块组成类的逻辑视图：

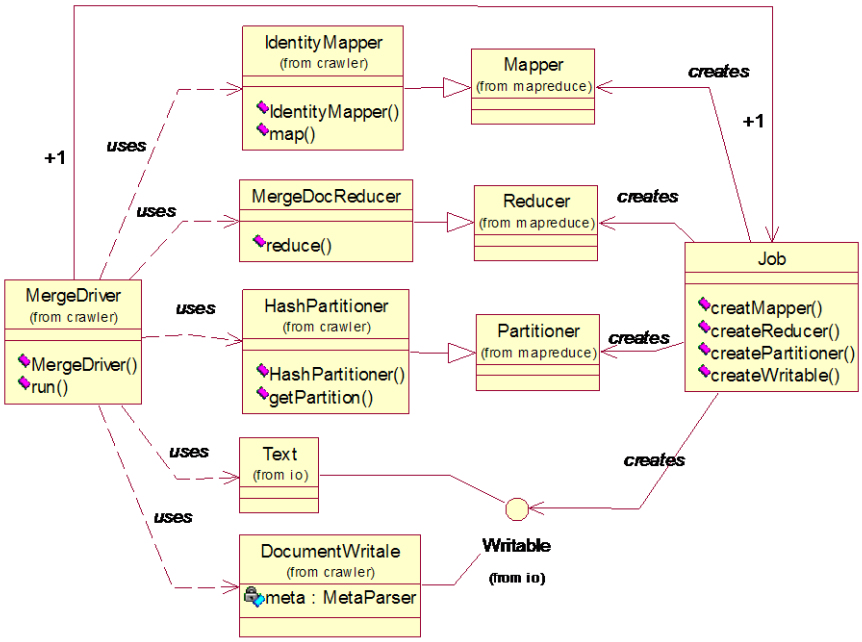

图 4-5 MergeDriver 模块组成类的逻辑视图

首先来介绍功能类：

MergeDriver 类：配置和启动 Map/Reduce 程序。在配置时主要完成的工作是设置执行 map 函数的类和设置 Key/Value 对的类。

其次来介绍辅助类：

火龙果•整理
uml.org.cn

HashPartitioner 类：在做 reduce 阶段之前按照 key(即 url)的 Hash 进行分区，保证相同 url 的 value 值被分到一个 reduce 中处理。

再次来介绍 Map/Reduce 核心类：

(1) IdentityMapper 类：执行 map 函数的类，将输入的 <key,value> 原封不动的输出为中间结果。这个 map 函数输出的中间格式类型为<Text，DocumentWritable>，值为<url，网页 HTML 信息>。

(2) MergeDocReducer 类：执行 reduce 函数的类，遍历所属分区里的 value，即遍历相同 url 的所有网页信息(DocumentWritable)集合，当遍历第一个之后就停止遍历，url 仍作为 key 值，而第一个网页 HTML 信息作为 value 值。

最后来介绍 Key/Value 存储类型：

(1)Text 类型：字符串 String 的可序列化类型。

(2) DocumentWritable 类型：包含网页 HTML 等相关信息的复杂类型。在创建这个类型的对象时，会调用 MetaParser 类型，记录网页 Meta 信息分析。

对于 Map 函数，输入时它的 key 值为 Text，value 值为 DocumentWritable；输出时它的 key 值为 Text，value 为 DocumentWritable。对于 Reduce 函数，输入时它的 key 值为 Text，value 值为 DocumentWritable；输出时它的 key 值为 Text，value 值为 DocumentWritable。

下面详细介绍这些类之间的关系，通过上图 4-4

(1) MergeDriver 类有一个成员 Job 类型，在它的 run 函数中通过 Job 配置执行 Map 阶段、Partition 阶段和 Reduce 阶段的类型，另外还要配置每一个阶段 Key/Value 的类型。

(2)启动 Map/Reduce 作业，这时会将这些配置信息提交给 Hadoop 的底层，在类图中我们把底层全部抽象成 Job 类型，这样做是为了专注于本系统实现类的关系。

(3)Job 类通过反射机制创建 IdentityMapper、MergeDocReducer、HashPartitioner、Text 和 DocumentWritable 对象。

(4) MergeDriver 通过是通过使用这些对象完成 MergeDriver 模块的优化功能。

4.3.5 HtmlToXMLDriver 模块

HtmlToXMLDriver 模块完成网页的合并工作。该模块可划分为：功能类、

火龙果•整理
uml.org.cn

辅助类、Map/Reduce 核心类和 Key/Value 存储类型。其中，功能类包括 HtmlToXMLDriver； Map/Reduce 核心类包括 HtmlToXmlMapper；Key/Value 存储类型包括 Text 和 DocumentWritable。图 4-5 是 HtmlToXMLDriver 模块组成类的逻辑视图：

首先来介绍功能类：

HtmlToXMLDriver 类：配置和启动 Map/Reduce 程序。在配置时主要完成的工作是设置执行 map 函数的类和设置 Key/Value 对的类。

其次来介绍辅助类：

HashPartitioner 类：在做 reduce 阶段之前按照 key(即 url)的 Hash 进行分区，保证相同 url 的 value 值被分到一个 reduce 中处理。

再次来介绍 Map/Reduce 核心类：

HtmlToXmlMapper 类：执行 map 函数的类，通过 htmltoxml()方法将输入的 value 值即网页 HTML 信息转化为 XML 信息。这个 map 函数输出的中间格式类型为<Text，Text>，值为<url，网页 XML 信息>。

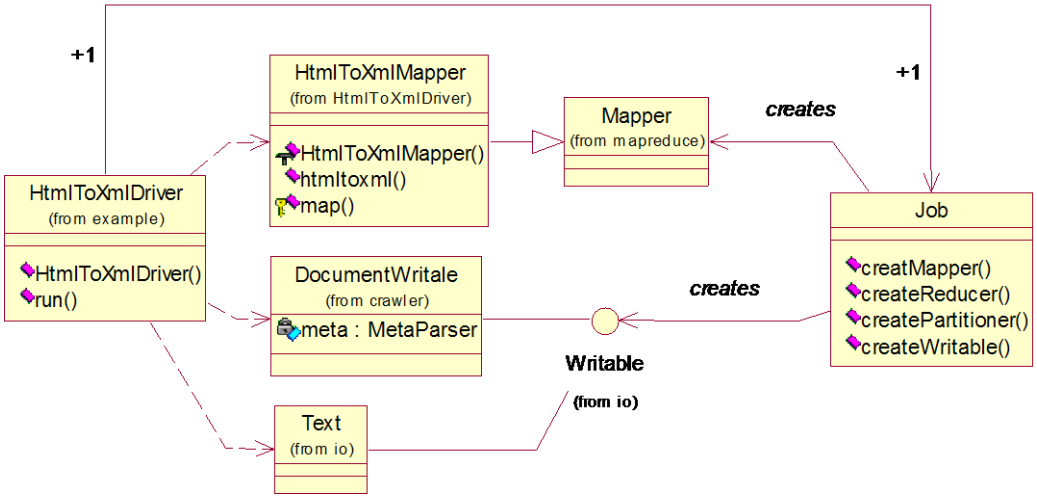

图 4-6 HtmlToXMLDriver 模块组成类的逻辑视图

最后来介绍 Key/Value 存储类型：

(1)Text 类型：字符串 String 的可序列化类型。

(2) DocumentWritable 类型：包含网页 HTML 等相关信息的复杂类型，具体信息在前面存储结构的实现中已经介绍了。在创建这个类型的对象时，会调用 MetaParser 类型，记录网页 Meta 信息分析。

对于 Map 函数，输入时它的 key 值为 Text，value 值为

火龙果•整理
uml.org.cn

DocumentWritable；输出时它的 key 值为 Text，value 为 Text。

4.4 本章小结

本章是系统的实现部分，基于上面的概要设计，本章开始实现基于 Hadoop 的分布式文件系统。

首先，本章实现了 4 个库存储结构，每个库承担着相应的功能。然后，阐述爬虫的整体数据结构，对爬虫的个功能模块之间的联系有一个整体的认识。最后，实现各个功能模块。

火龙果•整理
uml.org.cn

第5章 性能分析与评价

本章将对本系统的性能进行分析与评价。首先，展示本系统的界面，使人们直观地认识本爬虫系统存储网页的分布式文件系统。其次，目前系统已经运行 8 个月有余，所以展示和分析目前为止所抓数据也是本章的重点。再次，与单机版爬虫系统进行比较，突出分布式爬虫系统的优势。最后，对本系统中一些关键性的问题进行探讨和分析。

5.1 系统运行展示

对爬虫系统的运行界面进行展示，由于已抓取的网页是存放在 HDFS 分布式文件系统上，简单说所有数据存放于分散的物理机器上，所以管理这些数据是一个大问题，为了解决这个问题，本系统有一个友好地 WEB UI 界面去管理分布式文件系统上的所有网页数据。图 5-1 为 27 种语言网页数据的概览图：

http://61.167.60.78:50075/browseDirectory.jsp?dir=/user/root&namenodeInfoPort=50070

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
crawler1	dir				2011-06-09 09:39	rwxr-xr-x	root	supergroup
crawler2	dir				2011-06-08 21:42	rwxr-xr-x	root	supergroup
crawler3	dir				2011-06-09 07:13	rwxr-xr-x	root	supergroup
crawler_Korea	dir				2011-06-09 08:27	rwxr-xr-x	root	supergroup
crawler_ar	dir				2011-06-09 14:44	rwxr-xr-x	root	supergroup
crawler_de	dir				2011-06-09 16:15	rwxr-xr-x	root	supergroup
crawler_fr	dir				2011-06-09 16:21	rwxr-xr-x	root	supergroup
crawler_id	dir				2011-06-09 10:16	rwxr-xr-x	root	supergroup
crawler_in	dir				2011-06-09 11:47	rwxr-xr-x	root	supergroup
crawler_ir	dir				2011-06-09 13:43	rwxr-xr-x	root	supergroup
crawler_it	dir				2011-06-09 16:11	rwxr-xr-x	root	supergroup
crawler_kg	dir				2011-06-09 11:10	rwxr-xr-x	root	supergroup
crawler_kz	dir				2011-06-09 11:13	rwxr-xr-x	root	supergroup
crawler_la	dir				2011-06-09 09:06	rwxr-xr-x	root	supergroup
crawler_md	dir				2011-06-09 15:07	rwxr-xr-x	root	supergroup
crawler_mn	dir				2011-06-09 08:07	rwxr-xr-x	root	supergroup
crawler_my	dir				2011-06-09 09:07	rwxr-xr-x	root	supergroup
crawler_np	dir				2011-06-09 11:28	rwxr-xr-x	root	supergroup
crawler_pashto	dir				2011-06-09 12:42	rwxr-xr-x	root	supergroup
crawler_ph	dir				2011-06-09 09:08	rwxr-xr-x	root	supergroup
crawler_pt	dir				2011-06-09 16:29	rwxr-xr-x	root	supergroup
crawler_ru	dir				2011-06-09 14:51	rwxr-xr-x	root	supergroup
crawler_sp	dir				2011-06-09 17:13	rwxr-xr-x	root	supergroup
crawler_tj	dir				2011-06-09 11:08	rwxr-xr-x	root	supergroup
crawler_urdo	dir				2011-06-09 12:11	rwxr-xr-x	root	supergroup
crawler_vn	dir				2011-06-09 10:07	rwxr-xr-x	root	supergroup
crawler_zang	dir				2011-06-08 20:09	rwxr-xr-x	root	supergroup

图 5-1 27 种语言网页数据的概览图

火龙果•整理
uml.org.cn

通过分布式文件系统(HDFS)的网络接口就可以查看爬虫系统抓取的全部网页数据，从图中可以看到 27 个文件夹，每个文件夹相当于一个库存放着一种语言的网页数据。具体对应关系如表 5-1：

表 5-1 存储库与各种语言网页对应关系

库	语言	库	语言
crawler1/	中文网页	crawler2/	英文网页
crawler3/	日文网页	crawler_Korea/	韩文网页
crawler_ar/	阿拉伯语网页	crawler_de/	德语网页
crawler_fr/	法语网页	crawler_id/	印尼语网页
crawler_in/	印地语网页	crawler_ir/	波斯语网页
crawler_it/	意大利语网页	crawler_kg/	吉尔吉斯语网页
crawler_kz/	哈萨克语网页	crawler_la/	老挝语网页
crawler_md/	缅甸语网页	crawler_mn/	蒙古语网页
crawler_my/	马来语网页	crawler_np/	尼泊尔语网页
crawler_pashto/	普什图语网页	crawler_ph/	菲律宾语网页
crawler_pt/	葡萄牙语网页	crawler_ru/	俄语网页
crawler_sp/	西班牙语网页	crawler_tj/	塔吉克语网页
crawler_urdo/	乌尔都语网页	crawler_vn/	越南语网页
crawler_zang/	藏语网页		

另外，图 5-1 中 Modification Time 栏显示的是各种语言网页数据最近的更新时间。

看完了概览图下面我们以中文网页库为例，看看库中的结构，如图 5-2 所示：

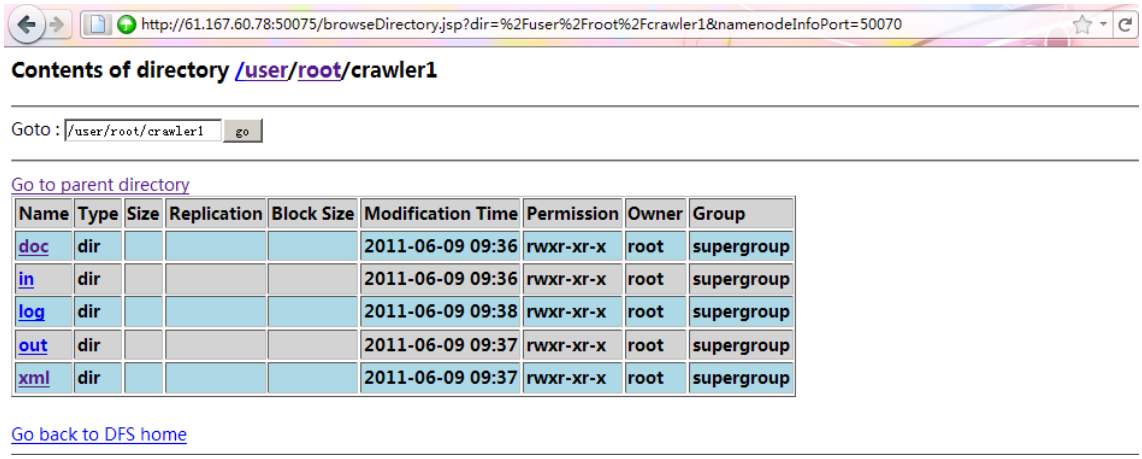

图 5-2 网页库组成结构

如上图所示，Contents of directory /user/root/crawler1 表示当前浏览的是 crawler1 文件夹即中文网页库。其中 doc 文件夹就是第四章提到的原始网页库

火龙果•整理
uml.org.cn

存储结构、in 文件夹为待抓取 URL 库存储结构、out 文件夹为链出 URL 库存储结构、xml 文件夹为 XML 库存储结构。

下面看 xml 库中内容，其他库我们就没有必要介绍了，因为 xml 文件夹存放的是本爬虫系统的最终结果文件，该文件记载了分布式爬虫系统抓取下来网页的 XML 信息。图 5-3 为中文网页库中 XML 库中信息：

http://61.167.60.78:50075/browseDirectory.jsp?dir=%2Fuser%2Froot%2Fcrawler1%2Fxml&namenodeInfoPort=50070

2011_05_15_09_19_55	dir				2011-05-15 09:20	rwxr-xr-x	root	supergroup
2011_05_16_09_31_33	dir				2011-05-16 09:32	rwxr-xr-x	root	supergroup
2011_05_17_09_23_19	dir				2011-05-17 09:24	rwxr-xr-x	root	supergroup
2011_05_18_09_42_09	dir				2011-05-18 09:43	rwxr-xr-x	root	supergroup
2011_05_19_09_23_06	dir				2011-05-19 09:24	rwxr-xr-x	root	supergroup
2011_05_20_09_30_14	dir				2011-05-20 09:31	rwxr-xr-x	root	supergroup
2011_05_21_09_24_08	dir				2011-05-21 09:25	rwxr-xr-x	root	supergroup
2011_05_22_10_00_03	dir				2011-05-22 10:00	rwxr-xr-x	root	supergroup
2011_05_23_09_44_22	dir				2011-05-23 09:45	rwxr-xr-x	root	supergroup
2011_05_24_09_36_48	dir				2011-05-24 09:37	rwxr-xr-x	root	supergroup
2011_05_25_09_43_47	dir				2011-05-25 09:45	rwxr-xr-x	root	supergroup
2011_05_26_09_33_37	dir				2011-05-26 09:34	rwxr-xr-x	root	supergroup
2011_05_27_09_38_19	dir				2011-05-27 09:39	rwxr-xr-x	root	supergroup
2011_05_28_09_23_25	dir				2011-05-28 09:24	rwxr-xr-x	root	supergroup
2011_05_29_09_19_07	dir				2011-05-29 09:19	rwxr-xr-x	root	supergroup
2011_05_30_09_35_10	dir				2011-05-30 09:36	rwxr-xr-x	root	supergroup
2011_05_31_09_38_45	dir				2011-05-31 09:39	rwxr-xr-x	root	supergroup
2011_06_01_09_46_55	dir				2011-06-01 09:48	rwxr-xr-x	root	supergroup
2011_06_02_09_33_17	dir				2011-06-02 09:34	rwxr-xr-x	root	supergroup
2011_06_03_09_32_30	dir				2011-06-03 09:33	rwxr-xr-x	root	supergroup
2011_06_04_09_19_20	dir				2011-06-04 09:20	rwxr-xr-x	root	supergroup
2011_06_05_09_16_03	dir				2011-06-05 09:16	rwxr-xr-x	root	supergroup
2011_06_06_09_16_19	dir				2011-06-06 09:17	rwxr-xr-x	root	supergroup
2011_06_07_09_26_38	dir				2011-06-07 09:27	rwxr-xr-x	root	supergroup
2011_06_08_09_34_51	dir				2011-06-08 09:36	rwxr-xr-x	root	supergroup
2011_06_09_09_37_51	dir				2011-06-09 09:38	rwxr-xr-x	root	supergroup

图 5-3 中文网页的 XML 库

如上图所示，在 XML 库中，爬虫系统为当天抓取的网页创建一个单独的文件夹，并以日期命名。本系统从 2010 年 9 月 23 日开始，截止本文截稿日期 2011 年 6 月 9 日，已爬取 8 个月有余。

接下来，我们以中文和日文为例来看网页的 XML 信息具体的保存形式。以 6 月 9 日抓取的中文网页和日文网页为例：

(1)中文网页：

火龙果•整理
uml.org.cn

Contents of directory /user/root/crawler1/xml/2011_06_09_09_37_51

Goto : 1/2011_06_09_09_37_51 go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
_logs	dir				2011-06-09 09:37	rwxr-xr-x	root	supergroup
part-r-00000	file	371.91 MB	3	64 MB	2011-06-09 09:38	rw-r--r--	root	supergroup

Go back to DFS home

图 5-4 中文网页的 XML 库 6 月 9 日所抓取网页

从上图中我们可以看到 6 月 9 日抓取的网页 XML 信息一共 371.91MB，Block Size 列显示 64MB 说明这个文件以 64MB 为一个文件块分散地存放于一台或者多台机器上。下图 5-5 显示中文网页 XML 信息的具体内容：


```
1 http://3g.ifeng.com/3g/index.html <?xml version="1.0" encoding="GB18030"?><html url="http://3g.ifeng.com/3g/index.html"
. gettime="2011_06_09_09_37_58"><xml_headline><meta http-equiv="Content-Type" content="text/html; charset=gb2312"
. /><title>手机凤凰网-手机上网综合门户</title><meta name="keywords"
. content="凤凰新媒体旗下中国领先的无线互联网门户网站，主流人群手机上网首选。随时随地为中国亿万主流手机用户提供包括大中华地区时
. 政、国际、社会、财经、时尚、娱乐等综合新闻资讯以及图铃下载、视频点播、读书、博报等互动内容，是目前中国最有影响力的无线互联网
. 门户之一！ 手机登陆网址：wap.ifeng.com" /></xml_headline><xml_contents>          <div id="wrap">        <div class="mainStop">
. <div id="footer"> <div><span><a href="http://www.ifeng.com/corp/about/" target="_blank">凤凰新媒体介绍</a> | <a
. href="http://www.ifeng.com/corp/ad/" target="_blank">广告服务</a> | <a href="http://www.ifeng.com/corp/job/"
. target="_blank">招聘信息</a> | <a href="http://www.ifeng.com/corp/privacy/" target="_blank">保护隐私权</a> | <a
. href="http://www.ifeng.com/corp/exemption/" target="_blank">免责条款</a> | <a href="http://www.ifeng.com/corp/counselor/"
. target="_blank">法律顾问</a> | <a href="http://www.ifeng.com/corp/feedback/" target="_blank">意见反馈</a> | <a
. href="http://phtv.ifeng.com/intro/" target="_blank">凤凰卫视介绍</a> <a href="http://www.irasia.com/listco/hk/phoenixtv/"
. target="_blank">投资者关系 Investor Relations</a></span></div> <div style="color:#666;"><span>凤凰新媒体 版权所有 不得转载 <a
. href="mailto:legal@ifeng.com">legal@ifeng.com </a>京ICP证030609号 本站通用网址：凤凰网</span></div> <div
. style="color:#666;"><span>建议使用IE5.5以上版本浏览</span></div> </div> </div>   </div></xml_contents></html>
2 http://3g.ifeng.com/web/index.html  <?xml version="1.0" encoding="GB18030"?><html url="http://3g.ifeng.com/web/index.html"
. gettime="2011_06_09_09_37_58"><xml_headline><meta http-equiv="Content-Type" content="text/html; charset=gb2312"
. /><title>手机凤凰网-手机上网综合门户</title><meta name="keywords"
. content="凤凰新媒体旗下中国领先的无线互联网门户网站，主流人群手机上网首选。随时随地为中国亿万主流手机用户提供包括大中华地区时
. 政、国际、社会、财经、时尚、娱乐等综合新闻资讯以及图铃下载、视频点播、读书、博报等互动内容，是目前中国最有影响力的无线互联网
. 门户之一！ 手机登陆网址：wap.ifeng.com" /></xml_headline><xml_contents>          <div id="wrap">        <div class="mainStop">
. <div id="footer"> <div><span><a href="http://www.ifeng.com/corp/about/" target="_blank">凤凰新媒体介绍</a> | <a
. href="http://www.ifeng.com/corp/ad/" target="_blank">广告服务</a> | <a href="http://www.ifeng.com/corp/job/"
. target="_blank">招聘信息</a> | <a href="http://www.ifeng.com/corp/privacy/" target="_blank">保护隐私权</a> | <a
. href="http://www.ifeng.com/corp/exemption/" target="_blank">免责条款</a> | <a href="http://www.ifeng.com/corp/counselor/"
. target="_blank">法律顾问</a> | <a href="http://www.ifeng.com/corp/feedback/" target="_blank">意见反馈</a> | <a
. href="http://phtv.ifeng.com/intro/" target="_blank">凤凰卫视介绍</a> <a href="http://www.irasia.com/listco/hk/phoenixtv/"
. target="_blank">投资者关系 Investor Relations</a></span></div> <div style="color:#666;"><span>凤凰新媒体 版权所有 不得转载 <a
. href="mailto:legal@ifeng.com">legal@ifeng.com </a>京ICP证030609号 本站通用网址：凤凰网</span></div> <div
. style="color:#666;"><span>建议使用IE5.5以上版本浏览</span></div> </div> </div>   </div></xml_contents></html>
3 http://astro.ifeng.com/ <?xml version="1.0" encoding="GB18030"?><html url="http://astro.ifeng.com/"
. gettime="2011_06_09_09_37_58"><xml_headline><meta http-equiv="Content-Type" content="text/html; charset=utf-8"
. /><title>星座首页_星座频道_凤凰网</title><meta
```


图 5-5 中文网页的 XML 信息具体内容

上图显示了中文网页 XML 信息的具体内容，其中网页与网页之间以”\n”为分隔符，这个我们在前面的存储结构的实现上面已经讲过。每个网页包括两方面信息，一个是 url，另一个是 url 对应网页的 xml 信息。xml 信息中包括字节编码，url，headline，content 等信息。

(2)日文网页：

Contents of directory /user/root/crawler3/xml/2011_06_09_07_12_39

Goto : /2011_06_09_07_12_39 go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
_logs	dir				2011-06-09 07:12	rwxr-xr-x	root	supergroup
part-r-00000	file	111.58 MB	3	64 MB	2011-06-09 07:13	rw-r--r--	root	supergroup

Go back to DFS home

图 5-6 日文网页的 XML 库 6 月 9 日所抓取网页

从上图中我们可以看到 6 月 9 日抓取的网页 XML 信息一共 111.58MB，Block Size 列显示 64MB 说明这个文件以 64MB 为一个文件块分散地存放于一台或者多台机器上。下图 5-7 显示中文网页 XML 信息的具体内容：

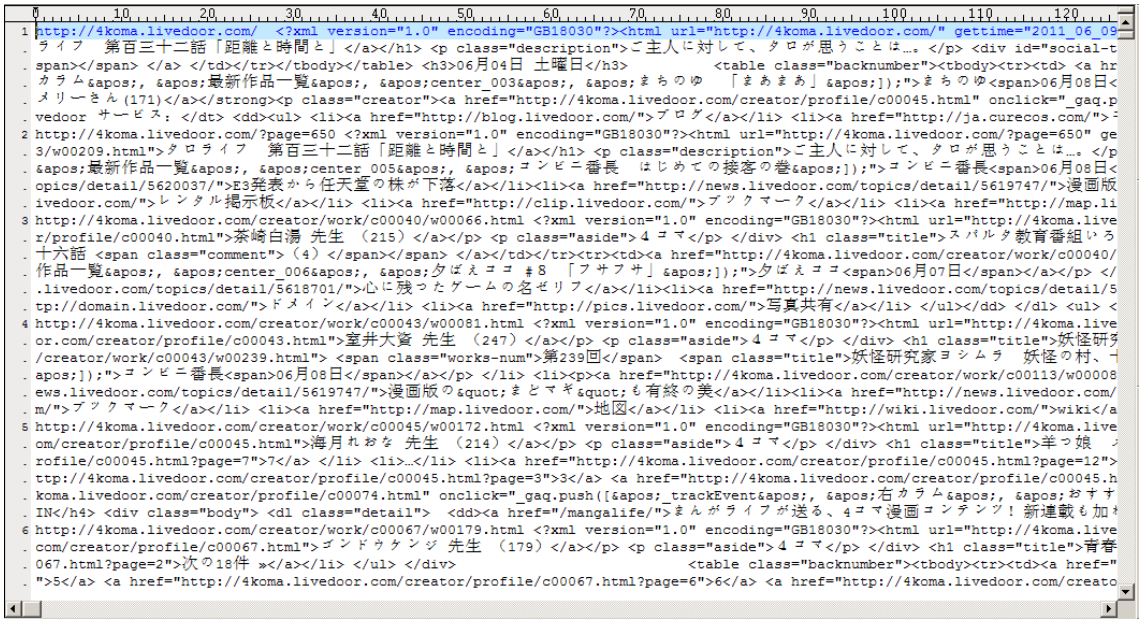

图 5-7 日文网页的 XML 信息具体内容

上图显示了中文网页 XML 信息的具体内容，其中网页与网页之间以”\n”为分隔符，这个我们在前面的存储结构的实现上面已经讲过。每个网页包括两方面信息，一个是 url，另一个是 url 对应网页的 xml 信息。xml 信息中包括字节编码，url，headline，content 等信息。

由于 27 种语言网页数目众多，在这里就不一一展示了。这里仅是两个示例给用户直观的感觉。

火龙果•整理
uml.org.cn

5.2 数据统计与分析

本小节对基于 Hadoop 分布式爬虫系统已抓取的网页数据进行统计和分析。本爬虫系统从 2010 年 9 月 23 日开始正式抓取网页。表 5-2 为已抓取网页数据统计和分析，限于篇幅，本文只截取 27 种语言的 7 种语言来展示，详细数据统计参见附录一：

表 5-2 已抓网页统计

库	语言	网页数(篇)	网站列表	空间
Crawler1/	中文	2,659,221	中国政府网、新华网、中国新闻网、台海网、中国经济网、中国台湾网、凤凰网、搜狐新闻、央视网	73G
Crawler2/	英文	1,657,098	美国政府、美国中央情报局、美国国家安全局、英国政府、英国王室、时代周刊、纽约时报、美国新闻与世界报道、美联社、英国广播公司(BBC)、泰晤士报、路透社、CNN、ABC、澳大利亚联邦政府、加拿大政府、澳大利亚总理和内阁部	29G
Crawler3/	日文	1,558,593	日本内阁府、日本外务省、读卖新闻、朝日新闻、日本经济新闻、NHK 新闻、共同社、msn 日本官网、时事通信社、日本证券新闻网、活力门	27G
Crawler_Korea/	韩文	1,946,202	朝鲜新闻、韩国联合通讯社、KBS、朝鲜中央通讯社、MBC 电视台、综合日报、汉城体育、Scout 、ohmynews、Edaily 娱乐、SBS 电视台集团网站、	40G
Crawler_ar/	阿拉伯语	641,760	卡塔尔半岛电视台、阿拉伯报纸联盟网、埃及《中东时报》、巴勒斯坦《耶路撒冷时报》、中东国家新闻媒体联合网、沙特新闻社、Maktoob.com-新闻、阿曼苏丹国信息产业部、迪拜市政府、阿尔及利亚新闻通讯社、阿尔及利亚外交部	12G
Crawler_de/	德语	2,146,148	德国联邦政府、德国《明镜周刊》、德国《图片报》、德国《世界报》、德国《莱茵邮报》、德国《焦点周刊》、德国《法兰克福汇刊》、德国《汉堡早邮报》、德国《斯图加特报》、德国《柏林晨邮报》、德国《南德意志报》、德国《金融时报》、德国《莱比锡人民报》	116G

依据上表给出的数据统计，我们给出如下分析：

自 2010 年 9 月 23 日开始，截止本文截稿时间 2011 年 6 月 9 日，在这 259 天时间内共抓取 15,977,969 篇网页，网页数据占用空间为 451.489G。共采集 165 个网站，涉及 27 种语言。

本爬虫系统针对每一种语言每天定时抓取一次网页数据，由上面已抓取数

火龙果•整理
uml.org.cn

据分析得出，本爬虫系统平均每日抓取 61,691 篇网页，每日抓取的数据量为 1.74G。

本爬虫系统不仅采用分布式结构，而且在每台机器上的 Reduce 进程中还采用了多线程技术，使得抓取效率进一步提高，所用的线程数可通过爬虫配置文件进行配置，默认的线程数为 30 个。

5.3 与非分布式爬虫的比较

本文开发的基于 Hadoop 的分布式网络爬虫系统与常见的单机版网络爬虫相比具有很多优点。下面以目前比较流行的开源单机版网络爬虫 Larbin 为例，对本系统和 Larbin 系统从多个角度进行对比。具体可从 7 个方面来描述：

(1)可扩展性：本爬虫系统是基于 Hadoop 开发的，可扩展性很好，可以很轻松地增加机器节点，使得多台机器协同工作。而 Larbin 系统是单机版系统缺乏节点的扩展性，即使是在多台机器都部署 Larbin 也无法使得这些机器协同工作。

(2)存储方式：本爬虫系统将抓取下来的网页保存在集群的分布式文件系统(HDFS)上，分布式文件系统的网页数据很容易管理。而 Larbin 系统采集下来的网页是保存在本地文件系统上。

(3)网页预处理：本系统对抓取下来的网页信息进行预处理，将原始网页转换为 XML 格式，这样的半结构化信息有利于后期的信息处理和索引。而 Larbin 系统只保存网页的原始 HTML 信息。

(4)抓取效率：由于本系统是采用分布式架构，多台机器可协同抓去网页，同时在每台机器上还是多线程抓取，所以效率必然比单机抓取网页的 Larbin 系统高。

(5)吞吐量：采用分布式架构的爬虫系统吞吐量远远大于单节点抓取网页的 Larbin 系统。

(6)稳定性：本爬虫系统是基于 Hadoop 分布式框架的，当某一台机器崩溃时，不会影响整个集群的工作，也不会因此而丢失那个崩溃机器上的数据。

(7)易操作性，只需将 Hadoop 集群配置好就可以很轻松地启动爬虫的抓取任务。同时本爬虫系统的配置文件很容易配置。

基于 Hadoop 的分布式爬虫系统与 Larbin 系统的对比如下：

由表 5-3 中可见本系统还有很多需要完善的地方，主要表现在两个方面，(1)功能方面，本系统的功能还是比较单一，目前只能抓取网页的 HTML 信

火龙果•整理
uml.org.cn

息。后面还应该增加抓取 doc、ppt、pdf、图片等信息的功能。(2)支持镜像方面，目前的系统不支持镜像，而 Larbin 可以轻而易举地镜像一个网站。

表 5-3 系统比较表

系统名称	开发语言	功能	是否采用分布式	效率	是否支持镜像
本系统	JAVA	功能单一只抓取网页信息	是	高	否
Larbin	C++	功能多样化，除了网页信息还可以可抓取 doc,pdf,ppt 和图片等信息	否	低	是

针对这两点需要改进的地方，本文给出如下解决方案：(1)增加本爬虫系统的抓取方式，可改进 CrawlerDriver 模块中的辅助类 Downloader 类，使得该辅助类能够抓取 doc，pdf，ppt 和图片等信息。同时还应该在分布式文件系统建立存储相应信息的库。(2)可参考 Larbin 或者 WebLech 开源爬虫系统的镜像网站模块，尽量支持按功能需求来下载 web 站点并能够尽可能模仿标准 Web 浏览器的行为。

5.4 本章小结

依据前四章设计和实现的分布式爬虫系统，本章是对基于 Hadoop 分布式爬虫系统的性能进行分析与评价。

首先，本章展示了本爬虫系统的运行界面，给用户以直观的感觉。然后，统计本系统截止至 2011 年 6 月 9 日的全部数据，并对这些数据作出分析，从侧面分析本爬虫系统的性能。最后，用本爬虫系统系统与单节点爬虫进行比较，主要是与目前较为流行的开源非分布式爬虫 Larbin 进行比较，凸显本分布式爬虫系统的优势。

火龙果•整理
uml.org.cn

结　论

本文的主要目标是设计并实现了一个针对 27 种语言新闻信息并基于 Hadoop 的分布式爬虫系统。

首先，本文介绍了本课题的来源、研究背景和意义。同时，对整篇文章的主要工作和内容给出综合性的阐述，然后简单介绍全篇文章的内容组织结构。

其次，本文对这个爬虫系统的技术背景进行简单地介绍，为后面设计和实现基于 Hadoop 分布式爬虫系统提供技术基础。这里的背景介绍主要包括云计算的定义、基本原理和体系结构；Hadoop 平台 HDFS 文件系统和 Map/Reduce 模型；一般的网络爬虫基本原理；当下分布式网络爬虫系统的研究现状。

再次，对整个基于 Hadoop 的分布式爬虫系统进行概要设计，指导后面爬虫系统的详细设计和代码编写。这部分的概要设计主要是指系统的功能分析和结构设计。包括四个部分：需求分析，描述爬虫系统的设计需求；系统布局，对整个跨语言信息获取和检索项目进行模块划分，明确爬虫系统的设计要点；爬虫系统基本结构，设计爬虫系统基本流程和框架；子模块 Map/Reduce 设计，对框架中各功能模块进行 Map/Reduce 设计。

接着，对爬虫系统进行详细设计并编写代码，实现整个系统。对于实现这个部分，本文主要是从三个方面进行设计：存储结构设计，对本系统用到的 4 个存储结构的数据结构和数据存放形式进行介绍；爬虫总体数据结构设计，主要介绍各个模块的功能类之间的业务逻辑；功能模块的实现，介绍各模块功能类的数据结构，并通过类图讲解各模块的代码实现。

最后，对本系统进行性能分析和评价。通过数据统计和分析来说明爬虫系统的效率，另外，通过与非分布式爬虫在各方面性能的比较，来说明本分布式爬虫系统的优势。

火龙果•整理
uml.org.cn

附录一 抓取数据统计表

库	语言	网页数(篇)	网站列表	空间
Crawler1/	中文	2,659,221	中国政府网、新华网、中国新闻网、台海网、中国经济网、中国台湾网、凤凰网、搜狐新闻、央视网	73G
Crawler2/	英文	1,657,098	美国政府、美国中央情报局、美国国家安全局、英国政府、英国王室、时代周刊、纽约时报、美国新闻与世界报道、美联社、英国广播公司(BBC)、泰晤士报、路透社、CNN、ABC、澳大利亚联邦政府、加拿大政府、澳大利亚总理和内阁部	29G
Crawler3/	日文	1,558,593	日本内阁府、日本外务省、读卖新闻、朝日新闻、日本经济新闻、NHK 新闻、共同社、msn 日本官网、时事通信社、日本证券新闻网、活力门	27G
Crawler_Korea/	韩文	1,946,202	朝鲜新闻、韩国联合通讯社、KBS、朝鲜中央通讯社、MBC 电视台、综合日报、汉城体育、Scout 、ohmynews、Edaily 娱乐、SBS 电视台集团网站、	40G
Crawler_ar/	阿拉伯语	641,760	卡塔尔半岛电视台、阿拉伯报纸联盟网、埃及《中东时报》、巴勒斯坦《耶路撒冷时报》、中东国家新闻媒体联合网、沙特新闻社、Maktoob.com-新闻、阿曼苏丹国信息产业部、迪拜市政府、阿尔及利亚新闻通讯社、阿尔及利亚外交部	12G
crawler_de/	德语	2,146,148	德国联邦政府、德国《明镜周刊》、德国《图片报》、德国《世界报》、德国《莱茵邮报》、德国《焦点周刊》、德国《法兰克福汇刊》、德国《汉堡早邮报》、德国《斯图加特报》、德国《柏林晨邮报》、德国《南德意志报》、德国《金融时报》、德国《莱比锡人民报》	116G
crawler_fr/	法语	672,461	法国外交部、世界报、费加罗报、法国新闻社、观点、回声报、人道报、法兰西西部报、法国 SIPA 图片新闻社、阿尔萨斯最新消息报	22G
crawler_id	印尼语	167,486	印尼政府、印尼安塔拉通讯社、印尼《共和国报》、印尼共和国电视台	2.5G
crawler_in/	印地语	210,062	BBC 印地语、oneindia、印度《商业标准》、福禄门电视台、今日印度、印度时报	6.5G
crawler_ir/	波斯语	347,436	伊朗外交部、伊朗共和国通讯社、伊朗伊斯兰共和国对外广播电台	4.4G
crawler_it/	意大利语	279,485	意大利安莎通讯社、意大利《24 小时太阳报》、意大利《金融日报》、意大利《晚邮报》、意大利政府	6.2G

火龙果•整理
uml.org.cn

(续表 1)

库	语言	网页数(篇)	网站列表	空间
crawler_kg/	吉尔吉斯语	12,460	吉尔吉斯斯坦政府、吉尔吉斯斯坦 24 小时通讯社、吉尔吉斯斯坦通讯社	586M
crawler_kz/	哈萨克语	148,909	哈萨克斯坦共和国政府、哈萨克斯坦外交部、哈萨克斯坦《真理报》、哈萨克斯坦《快报》	3.3G
crawler_la/	老挝语	3,329	老挝《人民报》、老挝新闻社、老挝国家广播电台	64M
crawler_md/	缅甸语	6,532	缅甸政府、IMNA	58M
crawler_mn/	蒙古语	33,833	蒙古政府、蒙古国家通讯社	581M
crawler_my/	马来语	87,655	马来西亚政府、马来西亚外交部、马来西亚前锋报	1.4G
crawler_np/	尼泊尔语	62,761	尼泊尔政府、eKantiput.com	2.1G
crawler_pashto/	普什图语	86,307	巴基斯坦通讯社、阿富汗总统府网站、塔利班政权的网站、阿富汗通讯部	1.9G
crawler_ph/	菲律宾语	46,377	菲律宾政府网、菲律宾快报	2.3G
crawler_pt/	葡萄牙语	333,295	巴西政府、巴西外交部网站、葡萄牙政府、巴西《环球报》、巴西《圣保罗州报》、巴西国家通讯社、葡萄牙《快报》、葡萄牙卢萨通讯社、莫桑比克银行、莫桑比克共和国	14G
crawler_ru/	俄语	966,784	俄罗斯联邦政府、Gazeta.ru、《共青团真理报》、俄通社－塔斯社、俄政治新闻社、俄罗斯国家通讯社、俄罗斯新闻社、国际文传电讯社	33G
crawler_sp/	西班牙语	1,305,753	西班牙政府、西班牙《世界报》、西班牙《国家报》、埃菲社、西班牙《加泰罗尼亚报》	35G
crawler_tj/	塔吉克语	40,738	塔吉克斯坦"亚洲之声"通讯社、塔通社、塔吉克斯坦国会	1.3G
crawler_urdo/	乌尔都语	119,920	巴基斯坦联合通讯社、Khabrain、巴基斯坦政府、巴基斯坦《每日时报》、BBC 乌尔都语频道	2.1G

火龙果•整理
uml.org.cn

(续表 2)

库	语言	网页数(篇)	网站列表	空间
crawler_vn/	越南语	284,370	越南政府、越南新闻社、越南通讯社、24H.com.vn	12G
crawler_zang/	藏语	152,994	中国共产党新闻网、青海藏语广播网、中国西藏信息中心、藏族历史网	3.2G

火龙果•整理
uml.org.cn

参考文献

[1] 中国互联网络信息中心(CNNIC).《中国互联网络发展状况统计报告 (第二十七次)》.2011。

[2] 程锦佳. 基于 Hadoop 的分布式爬虫及其实现[D]. 北京：北京邮电大学硕士学位论文，2010.

[3] 薛振华，杨艳娟. 分布式搜索引擎的研究[J]. 中国搜索研究中心，Vol.18,1999

[4] Luiz André Barroso , Jeffrey Dean , Urs Hölzle, Web Search for a Planet: The Google Cluster Architecture, IEEE Micro, v.23 n.2, p.22-28, March 2003 [doi>10.1109/MM.2003.1196112]

[5] Sanjay Ghemawat , Howard Gobioff , Shun-Tak Leung, The Google file system, Proceedings of the nineteenth ACM symposium on Operating systems principles, October 19-22, 2003, Bolton Landing, NY, USA [doi>10.1145/945445.945450]

[6] Jeffrey Dean , Sanjay Ghemawat, MapReduce: simplified data processing on large clusters, Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation, p.10-10, December 06-08, 2004, San Francisco, CA

[7] Mike Burrows, The Chubby lock service for loosely-coupled distributed systems, Proceedings of the 7th symposium on Operating systems design and implementation, November 06-08, 2006, Seattle, Washington

[8] Fay Chang , Jeffrey Dean , Sanjay Ghemawat , Wilson C. Hsieh , Deborah A. Wallach , Mike Burrows , Tushar Chandra , Andrew Fikes , Robert E. Gruber, Bigtable: a distributed storage system for structured data, Proceedings of the 7th symposium on Operating systems design and implementation, November 06-08, 2006, Seattle, Washington

[9] 维基百科官方网站. http://zh.wikipedia.org/wiki/云计算

[10]SearchCloudComputing 网站.http://searchcloudcomputing.techtarget.com/definition/cloud-computing

[11]Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion

火龙果•整理
uml.org.cn

Stoica, and Matei Zaharia. Above the clouds: A berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.

[12]斯蒂芬•贝克.Google 及其"云"智慧[J].商业周刊(中文版),2008

[13]张楠.解读亚马逊的云计算帝国[J].商业周刊(中文版),2011

[14]Dustin Amrhein, Scott Quint. 面向企业的云计算(IBM)[J]. IBM WebSphere Developer Technical Journal,2011

[15]黄华,杨德志,张建刚.分布式文件系統[J].中国计算机报.Vol 1429,1995.

[16]Najork, M., Heydon, A.: High-performance Web crawling. Technical Report Research Report 173, Compaq SRC (2001) Available at http://gatekeeper.research.compaq.com/pub/DEC/SRC/research-reports/abstracts/src-rr-173.html.

[17]Sergey Brin , Lawrence Page, The anatomy of a large-scale hypertextual Web search engine, Proceedings of the seventh international conference on World Wide Web 7, p.107-117, April 1998, Brisbane, Australia

[18]Paolo Boldi, Bruno Codenotti, Massimo Santini, and Sebastiano Vigna. Ubicrawler: A scalable fully distributed Web crawler. In Proc. AusWeb02. The Eighth Australian World Wide Web Conference, 2002. To appear in Software: Practice & Experience.

[19]Jenny Edwards , Kevin McCurley , John Tomlin, An adaptive model for optimizing performance of an incremental web crawler, Proceedings of the 10th international conference on World Wide Web, p.106-113, May 01-05, 2001, Hong Kong, Hong Kong [doi>10.1145/371920.371960]

[20]互动百科官方网站. http://www.hudong.com/wiki/云计算

[21]王鹏. 走进云计算[M]. 北京：人民邮电出版社，2009：117-121.

[22]Tom White. Hadoop: The Definitive Guide[M]. 周傲英/曾大聃，译. 北京：清华大学出版社. 32-35

[23]蒋建洪.主要分布式搜索引擎技术的研究[J].科学技术与工程,

[24]薛振华，杨艳娟.分布式搜索引擎的研究[J].南京理工大学学报，2003,5

[25]张建勋;古志民;郑超;云计算研究进展综述[J];计算机应用研究;2010 年 02 期

[26]E. Walker, "Benchmarking amazon ec2 for high-performance scientific computing," in USENIX ;login: magzine, Oct. 2008.

[27]Apache Hadoop. Available: http://hadoop.apache.org/

火龙果•整理
uml.org.cn

[28]Y. Gu and R. L. Grossman, "Sector and Sphere: the design and implementation of a high-performance data cloud," Philosophical transactions. Series A, Mathematical, physical,and engineering sciences, vol. 367, pp. 2429--2445, 2009.

[29]D. Borthakur. The Hadoop distributed file system: Architecture and design. http://lucene.apache.org/hadoop/hdfs_design.pdf, 2007.

[30]Jason Venner. Pro Hadoop[M]. USA:Apress .45-47.

[31]ABICloud http://sourceforge.net/projects/abicloud/

火龙果•整理
uml.org.cn

攻读学位期间发表的学术论文

1.Bowen Zheng, Dequan Zheng, Tiejun Zhao, Qingxuan Chen. A Re-Ranking Approach for Categorization Information Retrieval Based on Hierarchical Feature Selection[J]. Journal of Computational Information Systems,2009,5(6): 1609-1616

火龙果•整理
uml.org.cn

哈尔滨工业大学学位论文原创性声明及使用授权说明

学位论文原创性声明

本人郑重声明：此处所提交的学位论文《基于 Hadoop 的分布式网络爬虫技术》，是本人在导师指导下，在哈尔滨工业大学攻读学位期间独立进行研究工作所取得的成果。据本人所知，论文中除已注明部分外不包含他人已发表或撰写过的研究成果。对本文的研究工作做出重要贡献的个人和集体，均已在文中以明确方式注明。本声明的法律结果将完全由本人承担。

作者签名：郑博文　　日期：2011 年 6 月 27 日

学位论文使用授权说明

本人完全了解哈尔滨工业大学关于保存、使用学位论文的规定，即：

（1）已获学位的研究生必须按学校规定提交学位论文；（2）学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文；（3）为教学和科研目的，学校可以将学位论文作为资料在图书馆及校园网上提供目录检索与阅览服务；（4）根据相关要求，向国家图书馆报送学位论文。

保密论文在解密后遵守此规定。

本人保证遵守上述规定。

作者签名：郑博文　　日期：2011 年 6 月 27 日

导师签名：赵海燕　　日期：2011 年 6 月 27 日

火龙果•整理
uml.org.cn

致　谢

求学两载，终于到了要说再见的时候，回首这两年走过的足迹，有过迷茫，有过懈怠，但是更多的是收获和快乐。在此论文完成之际，我要向这两年帮助过我的老师、同学、朋友和家人表示感谢。

首先，我要感谢我的导师，赵铁军教授。赵老师在我两年硕士期间为我的学习和工作指明了大的方向。赵老师严谨的治学态度和甘为人梯的师德深深地影响了我，两年的时光，从赵老师身上获益颇多，他对待科研的态度也将一直影响我今后的工作和生活。

然后，感谢我的指导教师郑德权老师，郑老师是这两年来指导我最多的导师，不论是工作还是学习，他都给予我最大的帮助。郑老师待人热情、平易近人、风趣幽默，他身上所散发出来的人格魅力使得我由衷地对他生出了景仰之心。我想郑老师也是我的人生导师。

感谢杨沐昀、徐冰、李晗静老师，对我工作的指点。感谢朱聪慧师兄对我两年来硕士工作的指导，以及对我生活上的关心。

感谢杨宇航师兄、陈宇师兄、陈晨师兄、李凤环师姐在我刚进实验室时给予我的大力帮助和关怀。感谢黄海源、张练、姜舟、刘莎莎和张春越师兄师姐在我学习上的帮助。

感谢两年来和我共同奋斗和成长的胡亚楠、郑宏、王山雨、刘海波、朱晓宁、于墨、王超、李震、李大任和王垚尧。感谢他们在我硕士学习生活中的对我鼓励。

感谢费戈里、孙振龙、何晓春、田乐霄、秦艳霞、韩巍、胡鹏龙、吴建伟、王蟒、李伟、刘立群师弟师妹陪伴我度过快乐的硕士生活。

感谢培养我的母校哈尔滨工业大学，感谢计算机学院辛勤敬业的老师。

感谢我的女友毛梦亚对我一直以来支持，是你令我的生活充满色彩，也是你不断鼓励我勇往直前。

感谢养育我长大成人的父母。

感谢所有帮助过我的人。

火龙果•整理
uml.org.cn

基于 Hadoop的分布式网络爬虫技术

作者： 郑博文
学位授予单位： 哈尔滨工业大学