上海交通大学硕士学位论文

面向智慧环保的大数据整合与分析平台

硕士研究生：樊润冬

学　　号：1140379030

导　　师：吴刚

副 导 师：

申请学位：工程硕士

学　　科：软件工程

所在单位：软件学院

答辩日期：2017 年 1 月

授予学位单位：上海交通大学

Dissertation Submitted to Shanghai Jiao Tong University
for the Degree of Master

BIG DATA CONSOLIDATION AND ANALYSIS PLATFORM FOR SMART ENVIRONMENT PROTECTION

Candidate：	Rundong Fan
Student ID:	1140379030
Supervisor：	Gang Wu
Assistant Supervisor:	
Academic Degree Applied for：	Master of Engineering
Speciality：	Software Engineering
Affiliation：	School of Software
Date of Defence：	Jan, 2017
Degree-Conferring-Institution：	Shanghai Jiao Tong University

面向智慧环保的大数据整合与分析平台

摘 要

近年来，充分利用发展迅速的信息技术来提高城市治理水平的“智慧城市”概念吸引了很多人的关注。“智慧城市”的概念注重可持续发展和生活质量的提升，因此“智慧环保”成为了“智慧城市”实施中的关键一环。

智慧环保要求对环境数据进行长期、高效、可靠、灵活的整合与分析，而这些环境数据有着海量、规模增长迅速、异构程度高、来源广泛、流式数据与批量数据并存等诸多特点，是进行智慧环保相关应用开发的重大技术挑战。为了降低智慧环保应用的开发难度，助力智慧环保的实施，本文提出了一个面向智慧环保的大数据整合与分析平台，能够支持环境大数据的整合和分析，并且具有长期的实用性。

本文对该平台进行了设计。首先设计了该平台的数据整合平台，针对实时环境数据，制订了一种基于 JSON 的标准中间格式，并设计了用于格式转换的数据网关层架构，还设计了基于 Kafka 和 Spark Streaming 的数据获取、清洗、分类、存储流程。针对非实时的环境相关数据，设计了基于 NiFi 的整合流程。然后设计了该平台的数据分析平台，设计了基于 HDFS、YARN、Spark 和 Spark Streaming 的流式数据与批量数据分析平台，还为数据展现应用设计了数据展现接口架构。该数据展现接口架构包括数据展现 API 层以及基于 MongoDB 和 Redis 的暂存数据库。最后设计了该平台基于 Ambari 和 SaltStack 的运维支持系统。

本文对该平台进行了实现。首先配置了该平台的运行环境，然后实现了其关键组成系统，包括其数据网关、基于 Spark Streaming 的清洗与分类任务和内置数据展现 API 服务器，显示出该平台的设计是切

实可行的。

为了进一步评估该平台的能力，本文在该平台上设计并实现了智慧环保的应用案例“区域空气 PM2.5 污染 24 小时预报”。该应用利用了整合的实时和非实时的环境数据，运用了批量大数据分析来训练预测模型，运用了流式大数据分析来产生实时预报结果，并以可视化的方式展现了预报结果。对该应用的案例分析显示该平台具备支持智慧环保应用进行环境大数据整合和分析的能力。

本文还对该平台的可用性、可伸缩性、可运维性进行了设计和分析，在运行环境中对该平台的性能也进行了测试，结果显示均达到了预期的目标。

关键词：智慧环保、数据整合、大数据平台、Spark、Hadoop

BIG DATA CONSOLIDATION AND ANALYSIS PLATFORM FOR SMART ENVIRONMENT PROTECTION

ABSTRACT

In recent years, the concept of “smart cities”, which involves leveraging fast-developing information technology to improve the governance of cities, attracted a lot of attention. The concept of “smart cities” emphasizes sustainable development and improvement of life quality, which makes “smart environment protection” a vital part of the implementation of “smart cities”.

The concept of “smart environment protection” requires long-term, efficient, reliable and flexible consolidation and analysis for environmental data. However, these data are high-volume, rapidly-growing, heterogeneous, containing a mixture of stream data and batch data and coming from a variety of sources, and these features are major technical challenges of developing applications related to “smart environment protection”. In order to reduce the difficulty of developing “smart environment protection” applications, and help with the implementation of “smart environment protection”, this paper proposed a big data consolidation and analysis platform for “smart environment protection”, which supports the consolidation and analysis of environmental big data with long-term practicality.

This paper designed the platform. First, this paper designed the data consolidation platform. In order to deal with real-time environmental data, this paper developed a JSON-based standard intermediate format. This paper also designed the architecture of data gateway layer to do the conversion to the intermediate format. In addition, this paper designed the

consolidation procedure of data ingestion, data cleaning, data classification and data storage based on Kafka and Spark Streaming. Regarding data that are not real-time, this paper designed the NiFi-based consolidation procedure. Then this paper designed the data analysis platform. This paper designed stream data and batch data analysis platform based on HDFS, YARN, Spark and Spark Streaming. This paper also designed the architecture of temporary storage databases based on MongoDB and Redis. Then this paper designed the architecture of data presentation interface for data presentation applications, and the data presentation interface consists of a data presentation API layer and temporary storage databases. Finally, this paper designed the operations and support system based on Ambari and SaltStack.

This paper also implemented this platform. First, this paper configured the operating environment for this platform, then implemented its key components, including its data gateways, a data cleaning and classification job based on Spark Streaming and an integrated data presentation API server. The implementation shows the design of this platform is feasible.

In order to further evaluate the abilities of this platform, this paper also designed and implemented a “smart environment protection” application case, “regional PM2.5 air pollution 24-hour forecast”. This application utilizes consolidated real-time environmental data and environmental data that are not real-time, uses big data batch processing to train the prediction model, uses big data stream processing to generate the forecast and presents the forecast through visualization. Case study on this application shows that this platform has the abilities to support “smart environmental protection” applications to consolidate and analyze environmental big data.

This paper also designed and analyzed the availability, scalability and operability of this platform. Also, this paper benchmarked its performance in the operating environment. Results show that this platform achieves

expected goals.

KEY WORDS: smart environment protection, data consolidation, big data platform, Spark, Hadoop

目 录

图 录

表 录

第一章 绪论

1.1 研究背景

近年来，城市化在全世界范围内的进展非常快，已经有超过一半的世界人口居住在城市里，而且这一比例仍然在快速增长中[1]。而逐渐膨胀的城市带来了很多挑战，传统的管理方法已经捉襟见肘。面对新的挑战，充分运用发展迅速的信息技术来管理城市的“智慧城市”概念应运而生。“智慧城市”为管理城市提供了一种新的视角，它的核心理念是充分运用物联网技术和信息通信技术来智能地管理城市的资源并且高效地向市民提供公共服务[2]。“智慧城市”重视可持续发展和生活质量的提升，因而，环境保护成为了“智慧城市”中至关重要的一环。

“智慧城市”中的环境保护，也被称为“智慧环保”，正在获得越来越多的关注。基于“智慧城市”的理念，智慧环保通过充分运用信息技术，来掌控城市的环境状态，管理城市的环境污染，应急处置环境相关的突发事件并协助城市管理者做出符合城市真实情况的决策，从而大幅提升一个城市的环境治理水平。

智慧环保涉及分析城市中海量的环境数据，其中有很大一部分是通过运用物联网技术和信息通信技术采集的实时传感器数据，这些数据往往对应着城市特定空间位置的环境状态，例如噪声级别、空气中的颗粒物浓度和水流断面的酸碱度。还有一部分数据反映了其他环境相关因素的特征，包括环境污染源数据，环境污染影响数据和城市状态数据等。环境数据存在着海量、规模增长迅速、异构程度高、来源广泛、流式数据与批量数据并存等诸多特点，而智慧环保又要求对这些数据进行长期、高效、可靠、灵活的整合和分析，并以适当的形式反馈给公众和城市决策者，从而对智慧环保的技术实现提出了很大的挑战。

另一方面，大数据技术进展迅速，越来越多的大数据处理产品，例如 Spark 和 Hadoop，正在迅速拓展计算机系统对各种类型的大数据的分析能力。这些大数据处理软件可以提供可靠，高效的处理能力，能够胜任环境数据的处理这一重任。因而，将大数据处理软件用于智慧环保的建设，正在逐渐获得研究人员、企业和政府部门的关注。

1.2 研究意义

智慧环保具有诸多潜力，通过开发不同的应用，对不同的城市环境数据进行整合、分析，能够产生大量有价值的分析结果。这些结果可以被用于监控城市环境变化动态，管控城市污染源，加强城市环境规划和治理以及降低环境污染影响等多个方面。但智慧环保涉及的环境数据存在海量、规模增长迅速、异构、来源广泛、流式与批量数据并存等诸多特点，使得其数据整合和分析过程有较大的技术难度。与此同时，不同的智慧环保应用有着大量的类似的需求，都需要海量环境数据的收集、清洗、整合和存取能力，都需要海量数据的计算、分析和展现能力。

由此可见，如果相对独立地针对各类特定需求分别开发智慧环保相关应用，将可能会导致大量的重复工作，应用与各个系统和数据源的独立对接过程会使得灵活性有所下降，系统整体的可用性、可伸缩性和性能也难以保证。相比之下，将智慧环保中主要的数据整合和分析功能独立出来，作为一个公共的数据整合与分析平台，从而向各个应用提供基础设施，将有助于提高智慧环保应用的开发便利性，提高可用性、可伸缩性、性能和灵活性，拓展应用的深度和广度。

本文力求提出一个面向智慧环保的大数据整合与分析平台，能够以统一的方式解决智慧环保实施中面临的环境数据整合和分析中的技术问题，并在这个过程中保持高可用性、高可伸缩性和高性能。这样一个平台将在智慧环保的实施中起到关键的支撑作用，大幅降低相关应用的开发难度，从而助力智慧环保的实施。

1.3 研究目标和工作内容

本文的研究目标即为设计一个面向智慧环保的大数据整合与分析平台（下文简称“本平台”），满足智慧环保相关应用对环境数据的整合以及进行大数据分析的需求，从而能够支持相关应用的快速开发。本平台的系统设计主要有如下目标：

（1） 本平台应当支持环境数据的整合。环境数据往往由不同的物联网设备采集而来，设备的厂商、型号、版本和供应商的数据存储格式等因素都会影响环境数据的形式。此外，环境数据既包含环境监测设备的实时数据，也包含其他非实时监测设备和其他系统中的批量数据。其中，环境监测设备的实时数据往往具有一定的数据标准，包括空气污染数据、噪声污染数据和水污染数据等，可以进行标准化。其他批量数据往往属于复杂的环境相关数据，通常具有特别的处理方式，不宜进行标准化，例如特殊污染源排放，城市居住人口密度和医院相关疾病

就诊数等。具体来说，本平台应当支持将来自多种环境监测设备的、异构的实时数据进行标准化、清洗、分类和存储，对于其他不宜标准化的复杂环境相关数据，也应当进行整合，以供后续处理。

（2） 本平台应当支持环境数据的分析。环境数据的分析是智慧环保的关键组成部分。环境数据的海量性，对本平台的存储能力和计算能力都提出了一定的要求，数据的展现也是分析过程中关键的一环。具体来说，本平台应当支持分析流式和批量的环境数据，并为数据的展现提供一定的支持。

（3） 本平台应当具备长期实用性。作为支撑智慧环保的关键系统，可用性对所有智慧环保应用的正常运作非常重要。另一方面，考虑到环境数据的海量性和智慧环保应用的多样性，本平台也应当具备相适应的处理性能。除此之外，随着监测项目、监测点和环境相关系统的迅速增多，环境数据规模的增长也非常迅猛，系统应当具有一定的可伸缩性以适应智慧环保的发展。最后，运维也是保障平台长期运转的重要手段。具体来说，本平台应当高可用、高性能、可伸缩并且易于运维。

对应地，具体的工作内容包括：

（1） 数据整合平台的设计与实现。其中，设计部分主要包括设计环境设备实时数据的整合方法，制定其标准中间格式，以及设计非实时的环境相关数据的整合方法。实现部分则主要包括配置相关的运行环境和实现数据网关、清洗分类任务等数据整合过程中的关键系统。

（2） 数据分析平台的设计与实现。其中，设计部分主要包括设计流式和批量大数据的分析方法以及设计数据展现的接口。实现部分则主要包括配置相关的运行环境和实现内置数据展现 API 等数据分析过程中的关键系统。

（3） 运维支持系统的设计与实现。主要包括运维管理系统的选择及相关运行环境的配置。

（4） 分析和评估本平台的功能、可用性、可伸缩性、可运维性和性能。本文主要通过对实际应用进行案例分析来评估本平台的功能，对应的工作包括在本平台上设计和实现智慧环保应用“区域空气 PM2.5 污染 24 小时预报”，并生成测试数据进行测试。本平台的可用性和可伸缩性通过分析其组成部分的可用性和可伸缩性来进行评估，可运维性通过分析运维操作的实际实现方法进行评估。本平台的性能则主要采用性能测试进行评估。

1.4 本文结构

本文共六章，其章节的组织结构如下：

第一章为绪论，阐述了本文的研究背景，研究意义与目标和工作内容。

第二章为相关研究及技术。首先对当前“智慧城市”、 “智慧环保”领域的相关研究和实践的现状和现有的商业大数据平台进行了介绍。然后对本文涉及到的相关技术进行了介绍。

第三章为系统设计。首先介绍了本平台的总体设计，然后按照功能，分为三部分，分别介绍其数据整合平台、数据分析平台和运维支持系统的具体设计。最后介绍了用于评估本平台功能的应用案例“区域空气 PM2.5 污染 24 小时预报”的设计。

第四章为平台运行环境的配置与实现。首先介绍了用于研究本平台部署方法的运行环境，描述了其配置方法，然后介绍了平台中关键系统的实现，最后介绍了应用案例的实现。

第五章为分析与评估。首先对应用案例进行了案例分析，评估了本平台的功能。然后对本平台的可用性、可运维性和可伸缩性进行了分析。最后进行了性能测试与评估。

第六章为结论。总结了本文的主要工作、创新点，评估了结果和主要贡献，分析了本文工作目前的缺点和不足之处，并展望了未来进一步的工作。

第二章 相关研究及技术

2.1 相关研究

2.1.1 智慧城市

飞速发展的城市和快速进步的信息技术带来了“智慧城市”的概念。“智慧城市”是一个较为模糊的概念，并没有确切的定义[3]。从城市治理和技术的角度来看，“智慧城市”通常强调的是运用各种信息技术来增强城市的综合治理水平。其常见的实现方式是，利用遍布城市的传感器收集与城市有关的各种关键信息，并加以分析，从而智能地做出响应。

技术因素影响着“智慧城市”的方方面面，其他因素都在一定程度上受到技术因素的影响，因而可以说技术因素是影响“智慧城市”实施的最重要的因素[4]。事实上，“智慧城市”这一概念刚刚被提出的时候，其主要关注点就是将信息通信技术融入现代城市的基础设施[5]。近年来，“智慧城市”也开始逐渐地引入云计算技术[6]、物联网技术[7]和大数据技术[8]。

信息技术在“智慧城市”中的应用已经有很多研究。其中，有很多研究发现了“智慧城市”中数据的重要性，例如文献[9]就提出数据生态对“智慧城市”的建设至关重要，而文献[10]则指出了大数据在“智慧城市”建设中显示出的巨大潜力。相应地，也有一些在“智慧城市”中进行数据整合和分析的研究，例如SmartSantander“智慧城市”实验项目[11]中的 CiDAP 平台[12]。CiDAP 平台是为SmartSantander 项目设计的大数据平台，其设计着眼于通用性，不支持数据的语义，因此并未对接收的传感器数据进行标准化、清洗和分类。同时，其架构设计中流式处理任务仍然需要从暂存的 NoSQL 数据库中取出数据进行处理，不支持直接处理实时数据。此外，CiDAP 的设计中也不包含运维支持系统。

2.1.2 智慧环保

“智慧城市”的主要议题包括“智慧经济”、“智慧政府”、“智慧交通”、“智慧环保”、“智慧生活”等[13]。其中“智慧环保”关系到城市的可持续发展和人类的生活质量，是“智慧城市”最关键的议题之一。

“智慧环保”的概念基于“智慧城市”发展而来，是一个非常具有吸引力的概念。近年来国内也有越来越多的研究围绕着智慧环保展开[14]。智慧环保常见的

实施方式是首先采用大量环境监测设备，例如空气污染监测仪、水污染监测仪、噪声监测仪，进行环境数据采集。然后对采集到的数据进行收集、汇总、分析，最后从中挖掘出高价值的信息并展现给相关的政府机构或发布给公众。事实上，已经有很多研究表明，对环境数据进行整合与分析能得到很多有价值的结果，包括生成细粒度的城市空气质量热力图[15]，对空气污染进行预报[16]，评估环境污染对人体健康的影响[17]等。

因此，国内也有大量智慧环保的实践[18]。其中，在衢州市的智慧环保实践中，识别出智慧环保实施的关键问题包括业务系统的数据整合问题和环境监测设备的数据整合问题，但是只对这两个问题进行了针对性的解决，没有形成数据整合的平台[19]。与此同时，在文献[20]为无锡市实施智慧环保提供的路线图中，也提到环境监测数据的整合非常重要，并建议采用云存储等方式来进行整合，但没有提出具体的解决方案。

可见，和“智慧城市”类似，“智慧环保”的实施在很大程度上也依赖于对环境大数据的整合和分析，因此，一个面向智慧环保的大数据整合与分析平台，能够有效地助力智慧环保的实施。

2.1.3 商用大数据平台

很多商业企业在开源方案的基础上提出了商用大数据平台的解决方案和产品[21]，这些企业包括国外的 Cloudera、Hortonworks，国内的星环科技等。这些企业的方案和产品往往更加强调通用性，并且更加技术导向，它们在数据平台中采用的技术栈和架构各有特点，有一定的参考价值。

Cloudera 支持的主要开源产品为 Cloudera Data Hub(CDH)，其主要使用 Hadoop 技术栈为核心，采用 Sqoop、Kafka 和 Flume 来管理数据流，支持 MapReduce 和 Spark 进行大数据处理，采用 Cloudera Manager 来进行集群部署和监控，由 ZooKeeper 提供协作服务。

Hortonworks 支持的开源的大数据解决方案分为两个产品，其 Hortonworks Data Platform(HDP)产品更加注重处理批量数据，而 Hortonworks Data Flow(HDF)则主要用于处理流式数据。HDP 中也主要使用 Hadoop 技术栈作为基础，批量大数据处理采用 MapReduce 和 Spark 等多种技术，流式大数据处理主要采用 Storm，数据流管理方面采用 Sqoop、Kafka 和 Flume 等，集群部署和监控则采用 Ambari。HDF 则主要以 NiFi，Kafka 和 Storm 组成，专注于数据流管理和流式数据处理。这两个产品都使用了 ZooKeeper 提供协作服务。

星环科技提供的解决方案为 Transwarp Data Hub(TDH)，其对 Hadoop 技术栈进行了定制。TDH 同样支持 MapReduce 和基于 Spark 的内存计算解决方案 Transwarp Stream。TDH 的数据流管理方面采用了 Sqoop 和 Flume，其集群部署和监控则采用 Transwarp Manager，其中的很多系统也依赖 ZooKeeper 来提供协作服务。

这些商用大数据平台着眼于企业级应用，以提供通用产品为目标，因而并未针对智慧环保的数据特点和应用特点进行优化和完善，虽然能够提供一定的技术基础设施，但通常不能直接用于支撑智慧环保的数据整合和数据分析需求。

2.2 相关技术

2.2.1 数据获取技术

广义上的大数据整合包括采用各种手段对大数据进行共享和合并以备使用，其实质是对大数据进行管理，包括管理大数据的生命周期和其在生命周期中的变化[22]，即大数据的整理、变换、清洗和分类等处理也可以纳入大数据整合的范畴。因此数据收集、数据获取、数据集成、数据清洗都属于大数据整合的相关技术。

其中，数据获取是大数据整合中的关键一环，数据获取的目的是从不同来源接收数据，包括其他数据库、在线服务、后端日志等[23]，在数据获取领域，Apache Kafka 常常被用于直接获取流式数据，而 Apache NiFi 则可用于各种数据的获取[24]。

Kafka 消息队列最初被设计用于高性能地进行日志处理[25]，现在已经发展成为一个综合的流式数据处理平台。Kafka 中主要通过“发布-订阅”方式传输消息。在 Kafka 中存在生产者和消费者两种角色，同时可以设置很多主题。生产者和消费者通过主题收发消息。Kafka 会根据设置将主题分成多个分区，当集群部署时，一个主题的多个分区就可以分布在多台机器上，从而并行地进行收发，因此 Kafka 具有非常良好的水平扩展性。Kafka 能提供持久化、高可用、高性能的消息传输，已经广泛地集成进各种企业级的数据平台基础设施中。

NiFi 是一个通用的数据获取框架。它能够作为数据的中介连接多个系统。它拥有一个基于 Web 的用户界面，可以用拖放和图形化的方式灵活地进行配置，也能提供非常直观的反馈。NiFi 中有几个关键的概念，数据被封装成流文件 FlowFile 的形式在 NiFi 的组件之间流动，由流文件处理器 FlowFile Processor 进行处理，FlowFile Processor 可以读写 FlowFile 的元数据和内容，从而实现数据变换、数据路由、数据中转等各种功能。组件之间的数据传输称为 Connection，其实质是队

列，从而允许不同组件以不同的速度处理数据。NiFi 内置了数十种实用的流文件处理器，可以满足数据库访问、消息队列收发、HDFS 访问、服务接口调用、文件下载等多种数据获取需求，同时还能执行替换、筛选等数据变换操作，具有极高的通用性。

常见的数据获取系统还包括 Sqoop、Flume 和 Chukwa，其中 Sqoop 擅长从传统数据库中获取数据，而 Flume 和 Chukwa 擅长获取日志数据。

2.2.2 大数据分析平台

当前，对大数据进行分析的需求正在迅速增长，大数据分析平台应运而生。而对大数据进行处理的主流方向之一是尽可能增加平台的横向可扩展性[26]，从而应对不断增长的数据处理性能需求。经典的 MapReduce 平台基于可横向扩展的分布式存储，通过将任务分发给大量节点实现了可横向扩展的大数据分析平台[27]，其分布式存储的开源实现 HDFS 已经成为当前主流的分布式存储解决方案，之后 Spark 在其上进一步改进，通过将计算结果存储在内存中，大幅提高了大数据分析的性能[28]。通过改进 Hadoop 的资源调度，可横向扩展的，通用的计算资源协商器 YARN 被设计出来[29]，包括 Spark 在内的各种大数据分析平台得以在 YARN 上执行，进一步完善了大数据分析平台的生态。近年来，流式大数据计算逐渐进入人们的视野，Spark Streaming 通过将流式数据拆分为一个个小的数据包，实现了在 Spark 平台上对流式数据的处理，在牺牲了一定延迟的代价下，获得了较高的处理性能。

HDFS 是一个分布式文件系统，其目标是在廉价的通用硬件上面实现高容错的分布式文件系统，存放在 HDFS 上的数据会复制到多个主机上从而提高容错度。一个典型的 HDFS 集群中包含一个 NameNode，一个 Secondary NameNode 和一些 DataNode。数据存放在 DataNode 上，元数据存放在 NameNode 上，Secondary NameNode 为 NameNode 合并日志文件。在最新的版本里，NameNode 可以实现高可用，即建立一个待命的 NameNode，此时 Secondary NameNode 的工作即被待命的 NameNode 取代。在高可用的环境下，为了防止 NameNode 分裂，日志由 Quorum Journal Manager(QJM)管理，QJM 由大于等于 3 个的 JournalNode 实现，日志必须保证写入 JournalNode 中的绝对多数才能生效，从而防止了日志的不一致。其他常见的分布式存储还包括 MapR-FS 等。

YARN 是 Hadoop 的资源协商器，可以在集群中为分布式应用分配计算资源。一个典型的 YARN 集群包含一个 ResourceManager 和若干 NodeManager。应用提

交到 ResourceManager 上，ResourceManager 为其分配一些容器，然后在其中一个容器上启动该应用的 ApplicationMaster，ApplicationMaster 会协调该应用在分配到的容器上执行。在最新的版本里，ResourceManager 可以实现高可用，即增加一个待命的 ResourceManager 节点。除了 YARN，常见的资源管理器还包括 Mesos。

Spark 是一种分布式计算框架。相比 MapReduce 基于 HDFS，Spark 基于内存。它通过运用 Resilient Distributed Dataset(RDD)，使得分布式计算的中间结果可以保存在内存中，同时还能保持容错性，从而大大提高了计算性能。Spark Streaming 基于 Spark，通过封装 DStream[30]，Spark Streaming 将数据流打包成 RDD，并交给 Spark 引擎进行计算。这种方式虽然牺牲了一定的数据延迟，但是易于均衡负载，易于从故障中恢复，易于统一编程模型，而且具有很好的性能。除此之外，常见的大数据批量处理方案还包括 MapReduce。Storm 也是常见的流式数据处理平台，Kafka 的最新版本也包含了流式数据处理平台 Kafka Streaming。

2.2.3 相关技术的特点

HDFS 是经典的开源分布式文件系统，是 Hadoop 生态中的标准配备。MapR-FS 是商业公司 MapR 提出的一个改进的分布式文件存储，对性能、可靠性和易用性进行了改进，但在免费版中并不提供大多数高级功能，开放性也较差。采用 MapR-FS 会为平台的长期运行增加不确定性，一旦需要进行技术迁移可能需要付出很高的代价，而 HDFS 则受到社区和主流企业的广泛支持。

YARN 和 Mesos 的理念有一定的不同，相对来说 Mesos 更加接近底层，YARN 通常能提供更好的数据本地性。实际使用中，YARN 与 HDFS 同属 Hadoop 生态，社区支持较好，且 Spark 在 YARN 平台上的部署较为主流。

MapReduce 是经典的批量大数据处理框架，但是由于其需要将结果保存在 HDFS 上，导致性能表现不佳。Spark 利用 RDD 将结果保存在内存中，相比 MapReduce 具有更好的性能。Storm 不采用微批量的处理方式，适用于要求极低延迟的流式数据处理需求，但吞吐量较差，Kafka Streaming 的功能则较为基本且不太成熟。Spark Streaming 基于 Spark 和微批量，功能强大，较为成熟，性能高。

2.3 本章小结

本章首先介绍了“智慧城市”和“智慧环保”相关领域的研究现状，介绍了其中具有代表性的研究和实践的特点，然后介绍了本文涉及的相关技术，并介绍了其特点。

第三章 平台设计

3.1 平台总体设计

本文设计了一个面向智慧环保的大数据整合与分析平台。其包括三个主要组成部分，即数据整合平台、数据分析平台和运维支持系统。其中数据整合平台负责将多种环境数据进行整合，包括数据的接收、抓取、清洗、标准化、分类和存储等。数据经过整合后将由数据分析平台进行分析，数据分析平台的主要功能包括数据的存取、处理和提供展现接口等。而运维支持系统的主要功能则是部署平台软件，维护平台运行等。通过这些组成部分，本平台可以支持自定义的数据网关、自定义的数据展现 API、批量处理任务和流处理任务，进行环境大数据的整合和分析，从而支持面向智慧环保的应用。总体架构见图 3-1。

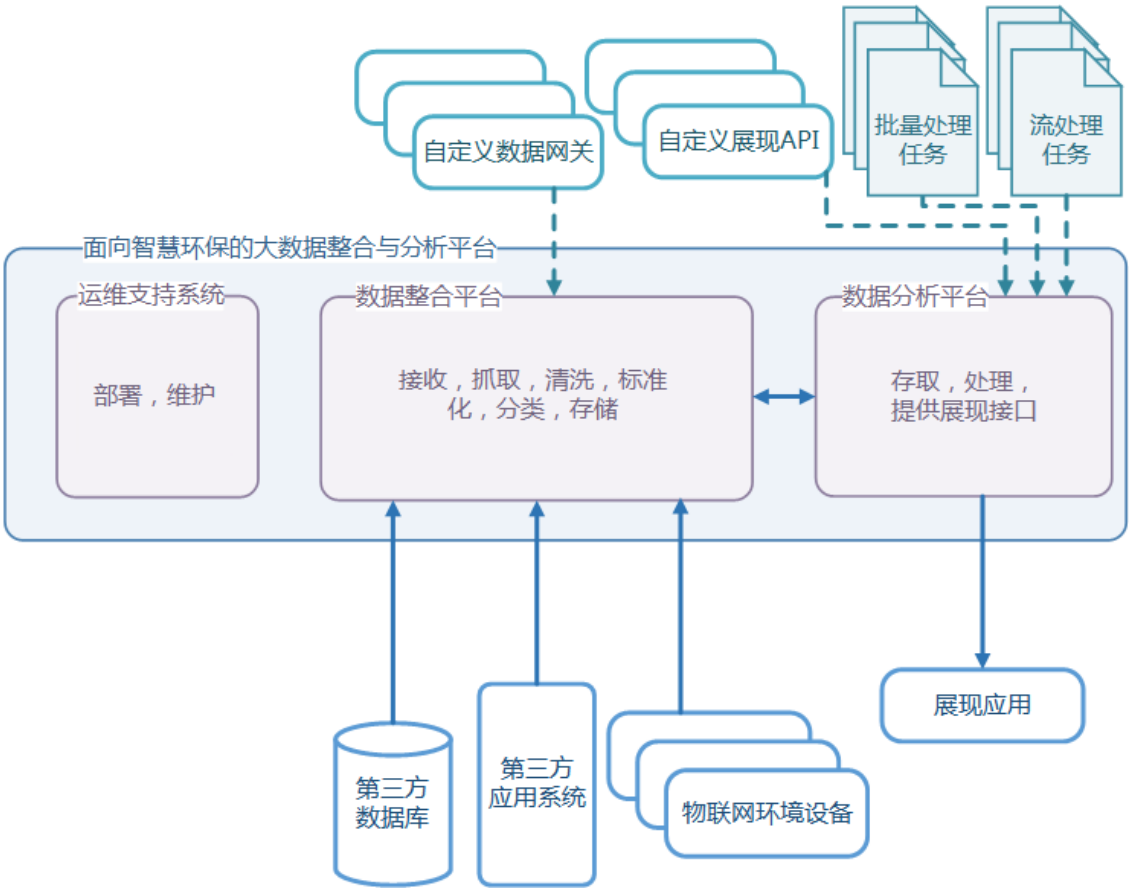

图 3-1 平台架构概览
Fig.3-1 Platform architecture overview

如图 3-1 所示，数据整合平台需要与第三方数据库、第三方应用系统以及物联网中的环境设备进行通信并获取数据，从而进行整合。其中，物联网环境设备包括空气污染监测设备、水污染监测设备、气象监测设备等直接采集数据并且上传的设备，不同的设备往往会采用不同的协议和通信方式，从而需要专门的数据网关层服务器负责收集这些设备的数据。本平台中包含两种内置的数据网关服务器，分别支持行业标准协议和本平台特有的协议，也支持运行其他自定义的数据网关进行数据收集。数据分析平台的功能主要包括执行批量数据和流式数据的分析任务，以及向展现应用提供相应的数据访问接口，除了本平台已经提供的内置数据展现 API 以外，也支持根据数据展现应用的需求，部署自定义的数据展现 API。

本章从数据整合平台、数据分析平台和运维支持系统三个方面对本平台的设计进行了描述。除此之外，本章还描述了用于评估本平台功能的应用案例“区域空气 PM2.5 污染 24 小时预报”的设计。

3.2 数据整合平台的设计

3.2.1 数据整合平台设计概述

数据整合平台的主要设计目的是对海量、异构、来源多样的环境数据进行接收、抓取、清洗、标准化、分类和存储。在智慧环保的应用场景中，环境数据往往有两个主要来源，一是来自大量物联网环境设备的实时采集上报，二是来自其他第三方数据库和应用系统的非实时环境相关数据。

物联网环境设备直接上报的数据主要为直接采集到的实时数据，例如实时空气粉尘浓度，实时噪声级别等。这些数据由于通常是数值形式，并且通常有相关的国家或者行业标准，所以可标准化程度较高。来自第三方数据库和应用系统的环境相关数据由于结构差异性较大，不适合进行标准化。这两部分数据的整合方式有一定的差异。

针对环境监测设备采集的可标准化的实时环境数据，本文制订了一种基于 JSON 的环境数据中间格式。这些可标准化的环境数据将由数据网关层的数据网关服务器进行接收，数据网关服务器会将数据翻译为该中间格式，并发布到 Kafka 消息队列中。之后，运行在 YARN 集群上的一组基于 Spark Streaming 的流式处理任务会从 Kafka 消息队列中取出数据，并对其进行进一步的清洗和分类，处理后的数据会被保存至 HDFS 集群中，并同时再次发布到 Kafka 消息队列中供平台上运行的其他流式处理任务使用。采用这种分层的处理方式，能够很好地解除数据

处理过程中不同组件之间的耦合[31]，并且能实现较好的性能。

针对来自第三方数据库和第三方应用系统的不宜进行标准化的环境数据，本平台会使用 NiFi 来管理这些数据的整合。通过内置的各种处理器，NiFi 可以直接访问第三方数据库并抓取数据，也可以直接访问第三方应用系统的接口来获得数据。通过在 NiFi 的 Web 界面上配置，这些数据可以被发布到 Kafka，也可以被存入 HDFS 中以供平台上运行的其他流式处理任务或批量处理任务使用。使用 NiFi 来整合这些数据有助于增加数据整合平台的通用性，与此同时，还能够保持很高的灵活性。

考虑到高可用性、可伸缩和性能方面的要求，本平台上的大部分系统采用了集群和高可用的设计，ZooKeeper 集群可以提供组建集群的一些基本服务[32]，Kafka 集群，NiFi 集群和 HDFS/YARN 集群都依赖于 ZooKeeper。

按照以上设计，数据整合平台的总体设计见图 3-2。其中 ZooKeeper 集群和 HDFS/YARN 集群为数据整合平台和数据分析平台所共用，这两个集群的具体细节将在 3.3 节介绍数据分析平台时进一步进行阐述。

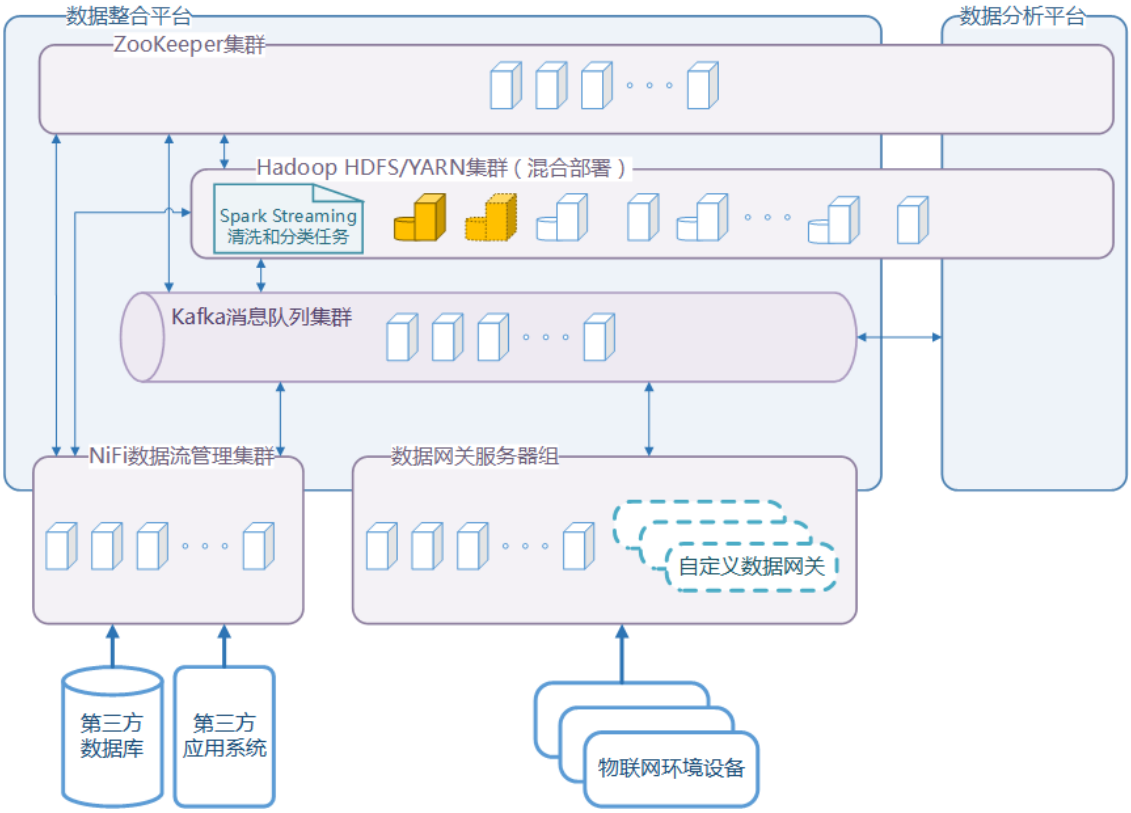

图 3-2 数据整合平台设计

Fig.3-2 Data consolidation platform design

3.2.2 环境设备采集数据的整合

来自物联网环境设备采集的数据通常为可标准化的实时数据，本平台会将这些数据进行标准化后以统一的格式进行处理和存储。

本文主要通过参考行业标准，即中国国家环境保护总局发布的“污染源在线自动监控（监测）系统数据传输标准”（HJ/T212-2005，下文简称 HJ/T212）[33]，同时结合行业特点和其他污染物相关国家标准[34-38]，对部分常用的环境监测项目及其对应的标识符、标准单位、数据类型、数据范围等进行了标准化，详见附录 1。对于不在附录 1 中的监测项目，可以在实际使用中采用与附录 1 中标识符不冲突的自定义标识符。

在此基础上，本文根据环境数据的领域特点，并且预留一定的可扩展性，设计了一种基于 JSON 的环境数据中间格式。该中间格式采用两级嵌套的 JSON 结构，其中的时间数据项需要采用符合 ISO8601:2004 标准[39]中表示时刻的时间字符串来描述。其中第一级的结构如表 3-1 所示，表中未注明是否可选的键即为必须包含。每条数据都代表环境监测设备的一次数据上传，其中包含了设备唯一标识符，采集时间，设备本地时间和实际采集到的监测数据值。

表 3-1 基于 JSON 的环境数据中间格式第一级结构
Table 3-1 First level structure of the JSON-based intermediate format for environmental data

键	数据类型	值
uuid	字符串形式的 64 位长整型	设备的唯一标识符
dataTime	字符串，需符合 ISO8601 中描述时刻的格式	本次上传数据的采集时间
deviceTime	字符串，需符合 ISO8601 中描述时刻的格式	本次上传数据时设备的本地时间
dataItemList	数组，元素为监测数据项 DataItem，监测数据项 DataItem 格式见表 3-2，可以为空数组	本次上传的监测数据的集合

第一级中包含的监测数据列表 dataItemList 为一个包含了监测项类型 DataItem 的数组。表示一次采集中采集到的多个数据，其中每个元素表示一项监测数据。监测项类型 DataItem 的数据结构如表 3-2 所示，同样，表中未注明是否可选的键即为必须包含。每项数据都包含一个标识符，标识了该项数据所属的监测类型，每项数据的采集时间可能有所不同，故每项监测数据都包含独立的采集时间信息。监测数据中除了监测值以外还包括监测值的单位标识符，显式地对监测值的单位进行说明有利于对数据进行标准化。

表 3-2 基于 JSON 的环境数据中间格式中的监测数据项 DataItem 结构
Table 3-2 DataItem type structure of JSON-based intermediate format for environmental data

键	数据类型	值
itemId	字符串	监测项目的标识符，常用标识符见附录 1，不在附录 1 中的监测项目可以使用与附录 1 中不冲突的自定义标识符
dataTime	字符串，需符合 ISO8601 中描述时刻的格式，可选，如不存在则采用第一级结构中的 dataTime 作为默认值	本次上传数据的采集时间
value	字符串形式，具体类型需与监测项目的类型相匹配	监测值
unit	字符串	监测值的单位标识符

数据网关服务器可以将接收到的数据转换为该标准格式。并且发布到 Kafka 消息队列中原始数据的主题中。本平台自身提供了两种内置的数据网关，使用 Node.js 编写。其中一种数据网关能够按照行业标准 HJ/T212 中实时数据上传的协议处理环境设备的实时上传数据。另一种数据网关能够处理以 HTTP POST 方式传输的，上文描述的基于 JSON 的标准化环境数据中间格式。

这两种数据网关分别以行业标准和平台内部格式为基准，可以支持大部分环保行业的标准设备以及特别为本平台配套的设备，从而覆盖了实际使用中的大部分设备。除此之外，如果需要接收使用特殊协议的设备的数据，只需要编写对应的自定义数据网关，对设备数据进行接收，并且把转换后的中间数据发送到 Kafka 消息队列中的原始数据主题即可。同时可以看出，由于消息传递功能主要由 Kafka 承担，数据网关服务器能够以服务器组的方式部署，并且根据具体需要灵活地采用各种负载均衡方式。

Kafka 以“发布-订阅”的模式处理消息。数据网关作为生产者向 Kafka 的原始数据主题发布 JSON 格式的中间数据消息。平台中关心某些数据的流式处理任务会订阅相关的主题，成为消息的消费者。考虑到可用性和性能，本平台中的 Kafka 采用集群部署，集群部署使得 Kafka 中一个主题的分区可以存在于更多的 broker 上，使得消息的发布和消费能在多个 broker 上并行地进行，从而扩展了消息队列的性能。综上，数据网关服务器组、Kafka 消息队列和流式处理任务的通信方式如图 3-3 所示，箭头方向为监测数据的流动方向，虚线连接线表示在逻辑上数据是发送到 Kafka 的原始数据主题上然后被订阅该主题的任务接收的。该主题会跨越多个分区，故监测数据实际上并行地流向多个 broker，并且被并行地接收。

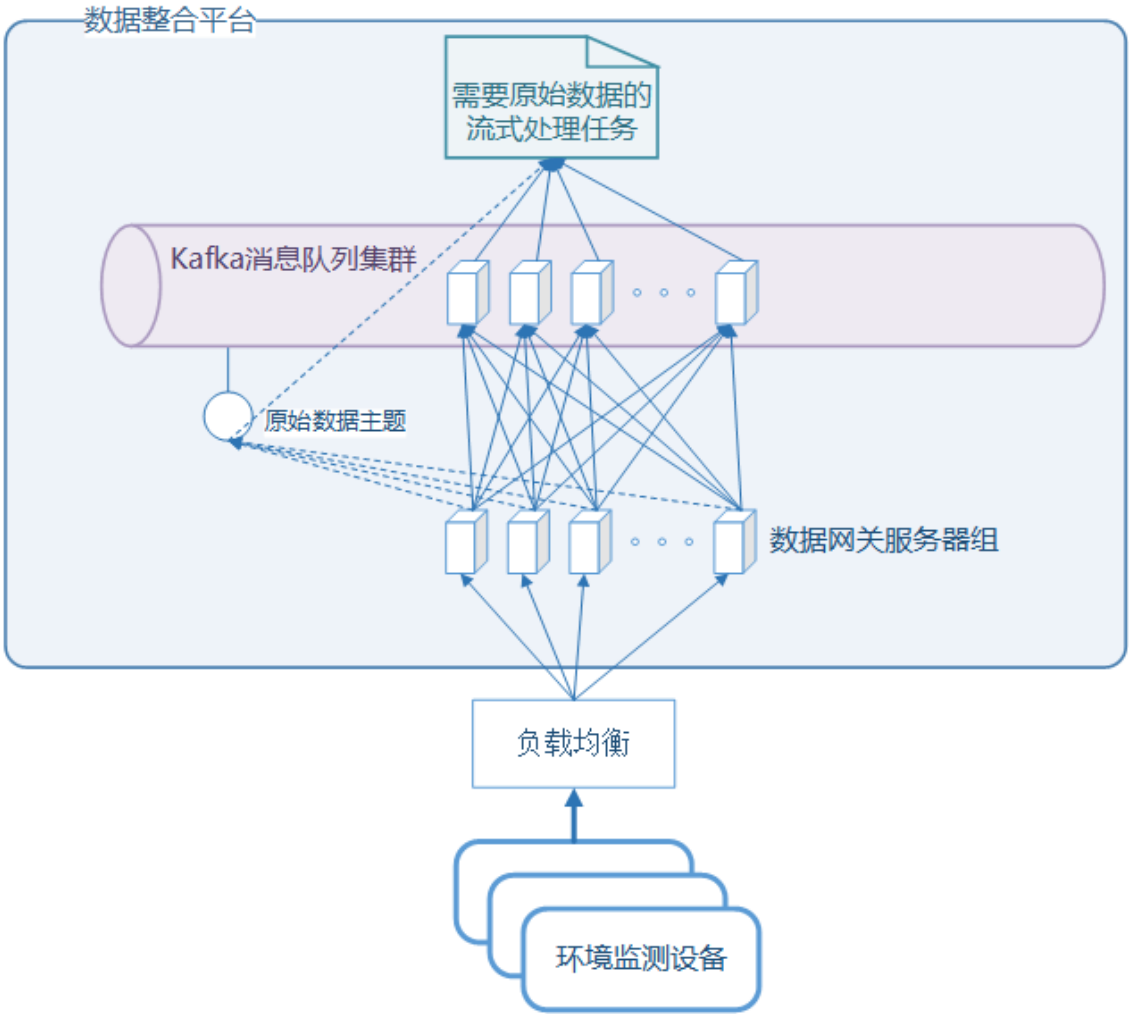

图 3-3 数据网关与 Kafka 消息队列的通信
Fig.3-3 Communication between data gateways and Kafka message queue

在 YARN 集群上运行的基于 Spark Streaming 的数据清洗和分类任务会订阅 Kafka 消息队列中的原始数据主题，并且取出原始数据进行清洗和分类。数据清洗和分类的主要工作是按照附录 1 的标准单位和数据范围进行格式转换并且将超出数据范围的数据和错误数据放入错误数据对应的主题中。清洗、分类后的数据，一方面发布到 Kafka 对应的主题上，另一方面以唯一标识符为划分基准，存储在 HDFS 中的不同文件里。

总结来说，对环境设备采集数据进行整合的数据流图如图 3-4 所示，设备以专有协议格式传输的数据将经过数据网关转换为一种基于 JSON 的环境数据中间格式。之后该未经分类和清洗的原始数据被发布到 Kafka 消息队列中，再由清洗和分类的 Spark Streaming 任务取出，经过清洗和分类后被存储到 HDFS 中，同时

也再次发布到 Kafka 队列中，从而完成整合。

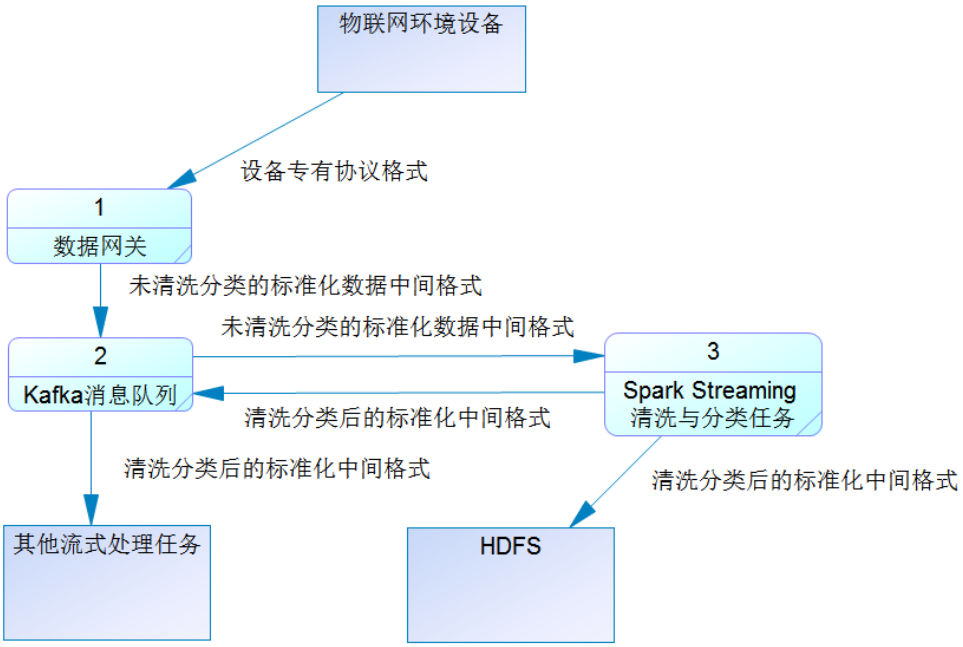

图 3-4 环境设备采集数据整合过程的数据流图

Fig.3-4 Data flow diagram of environmental device sample data consolidation procedure

3.2.3 其他环境相关数据的整合

智慧环保往往还需要处理很多非标准的环境相关数据。其中很多是与环境监测无关，但是可能对环境相关的数据分析有帮助的数据，例如天气预报、人口密度、企业信息和医院就诊数等，这些数据往往是 JSON，XML，CSV 等格式，以文件的形式存储或者需要通过调用 API 来获取，还有一些数据可能存放在第三方数据库中。

本平台采用了 NiFi 来整合这些来源的数据。通过适当的配置，NiFi 可以通过其内置的数据处理器将各种类型的数据转化为数据流文件的形式并加以管理。通过选择合适的数据处理器，如图 3-5，它可以通过 QueryDatabaseTable 处理器来访问第三方关系型数据库，通过 GetHTTP 处理器来访问第三方应用接口，通过 GetFTP 处理器来下载 FTP 文件。这些获取的数据会以数据流文件的形式在 NiFi 的处理器之间传输，其中包括一些能够对 JSON，CSV，XML 等格式进行转换、验证等处理的处理器。最后，经过处理后的数据流会通过 PutHDFS 处理器存储到 HDFS 上，也可以通过 PublishKafka 处理器来发布到 Kafka，以备平台上后续的数据分析任务使用。

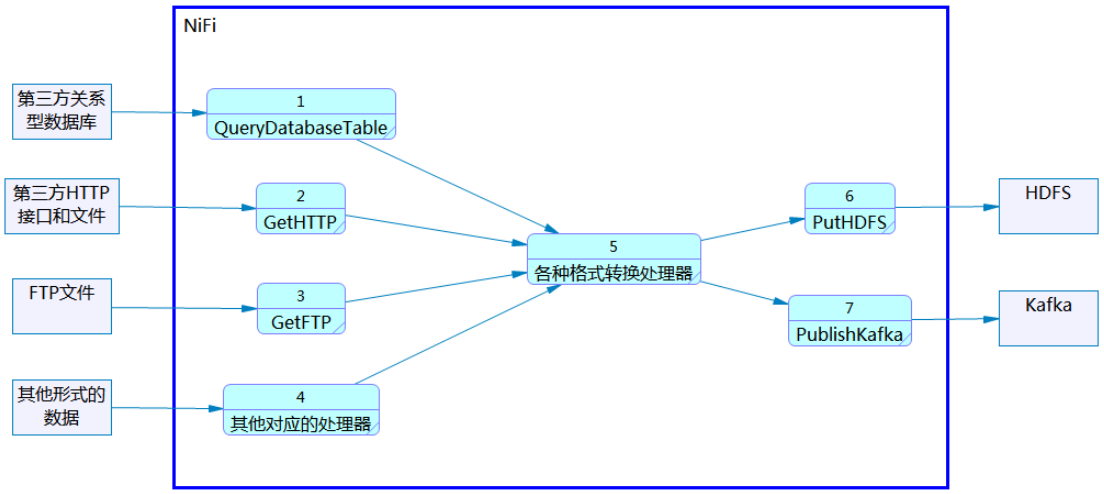

图 3-5 在 NiFi 中整合数据的数据流图
Fig.3-5 Data flow diagram of the data consolidation in NiFi

3.3 数据分析平台的设计

3.3.1 数据分析平台设计概述

数据分析平台的主要设计目标是存取海量的环境数据并进行分析、计算和挖掘，从中获得有价值的信息、规律乃至智慧，同时以适当的方式展现给相关的人员。数据分析的数据直接来源于数据整合平台，其中既包含经过清洗分类后的流式数据，也包含批量数据。

对于批量数据，运行在 YARN 集群上的基于 Spark 的批量处理任务可以从 HDFS 集群中加载环境数据并进行计算。计算的结果可以存储在 HDFS 集群中以供后续分析，同样也可以存储在暂存数据库中。

针对流式数据，运行在 YARN 集群上的基于 Spark Streaming 的流式处理任务可以直接向 Kafka 消息队列订阅需要的数据。计算的结果可以存储于 HDFS 集群中以供后续分析，也可以方便地存放在暂存服务器中以便数据展现 API 进行调取。

数据展现接口提供了一种方式将本平台内部的整合和分析结果暴露给外部应用，可以用于数据展现。数据展现接口包含一系列服务器，本平台内置了一个基础的数据展现接口 API 服务器，该服务器可以通过一定格式的对外接口，将 HDFS 集群中的文件，Kafka 中的流式数据和暂存数据库中的数据提供给外部应用。如果展现应用有特殊需要，本平台也支持自定义数据展现 API 服务器。

按照以上设计，数据分析平台的总体设计见图 3-6。

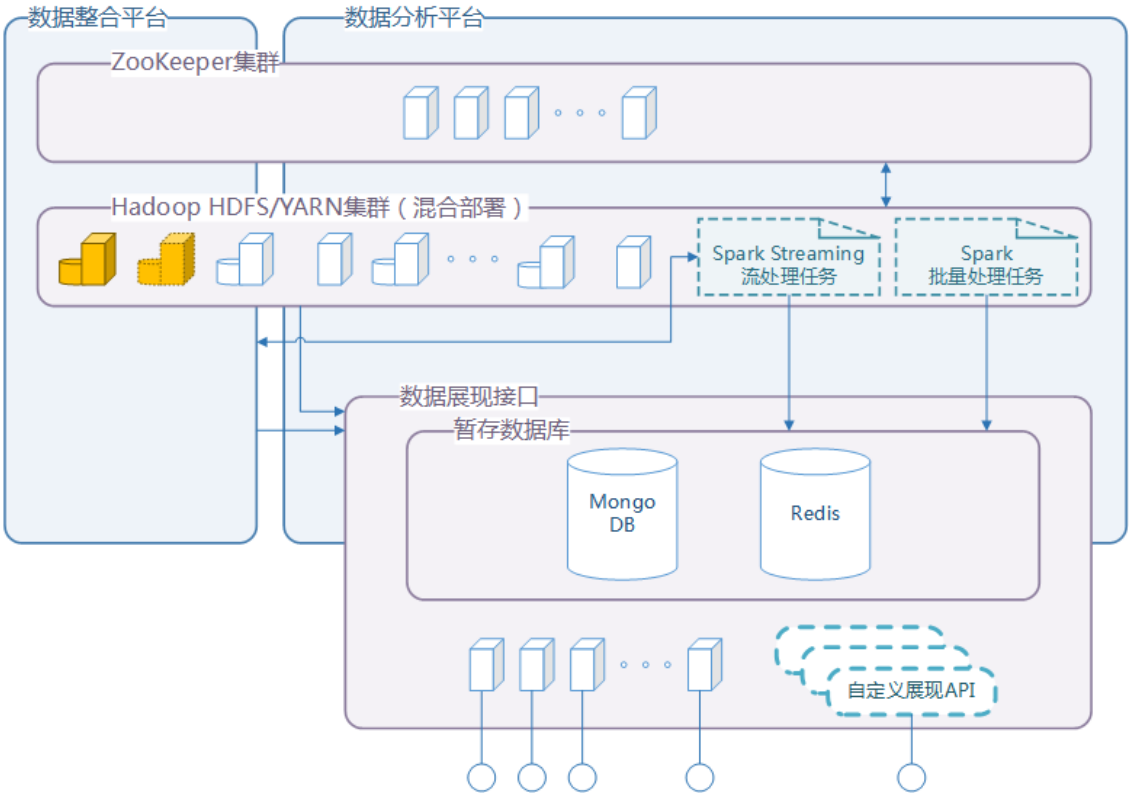

图 3-6 数据分析平台设计
Fig.3-6 Data analysis platform design

3.3.2 流式和批量大数据处理

本平台的大数据处理能力建立在 Hadoop 平台的基础之上，通过在同一个集群上部署 HDFS 集群和 YARN 集群，本平台获得了能够应对大数据的存储资源和计算资源。利用 YARN 提供的计算资源，本平台得以使用 Spark Streaming 处理流式数据，Spark 处理批量数据。ZooKeeper 集群则为 HDFS 集群和 YARN 集群主节点的热备提供支持。

出于充分利用计算资源和增加本地性的考虑，同时考虑到智慧环保业务的实际需求中，计算资源的伸缩需求往往高于存储资源，所以本平台的 HDFS 和 YARN 集群采用混合部署，即部署在同一个集群上，部分普通节点同时部署 HDFS 的 DataNode 和 YARN 的 NodeManager，其他节点作为纯计算节点只部署 NodeManager，以便于独立于存储资源进行伸缩。除此之外，为了便于管理，本平台的 HDFS 集群的 NameNode 以及 YARN 集群的 ResourceManager 也在一台机器上。出于提高可用性的考虑，都采用依赖于 ZooKeeper 集群的自动切换的热备方案，即还存在一个运行 NameNode 和 ResourceManager 的节点，在两个 NameNode

之间和两个 ResourceManager 之间都配置自动切换。由于采用了热备方案，Secondary NameNode 的工作被待命的 NameNode 取代，故集群中不再需要 Secondary NameNode 节点。活跃的 NameNode 和待命的 NameNode 之间通过 Quorum Journal Manager(QJM)来共享编辑日志，故还需要增加一组三台 JournalNode 服务器，在本平台中与 ZooKeeper 集群同位部署，整体的 HDFS 集群和 YARN 集群的部署方案如图 3-7 所示。

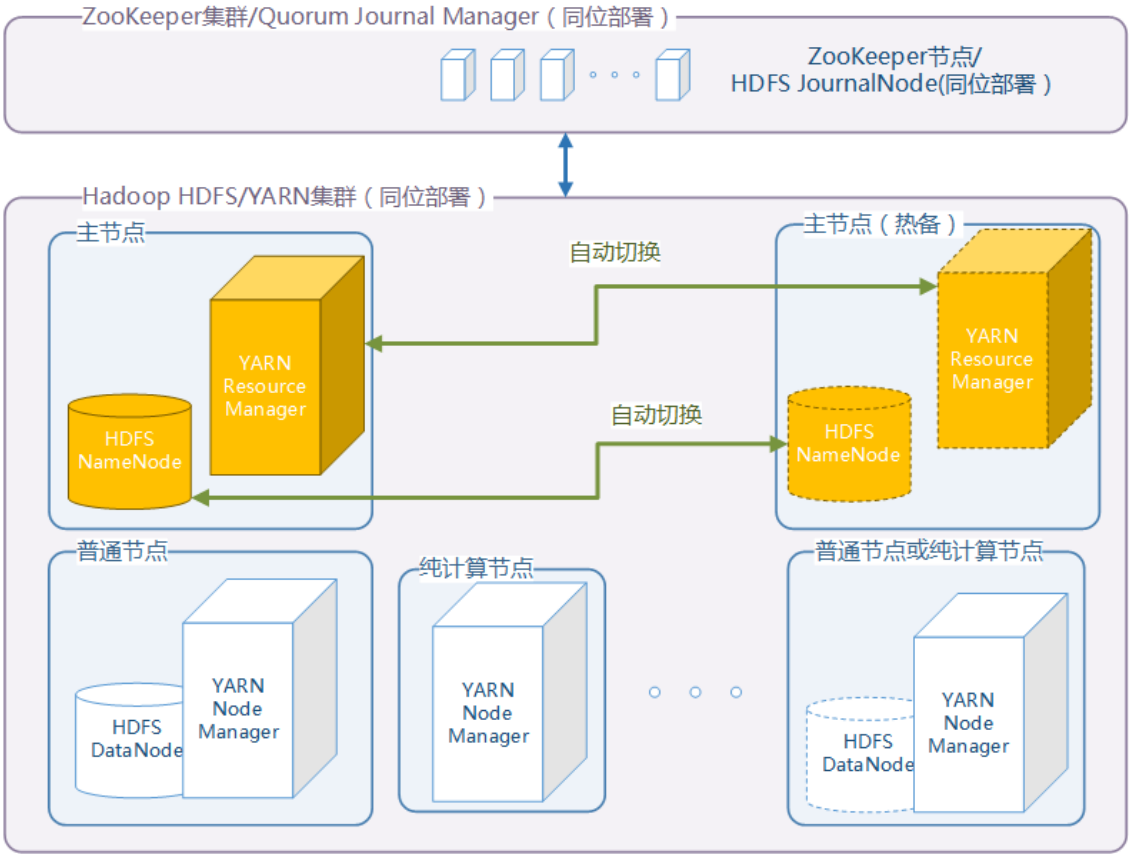

图 3-7 针对 HDFS 集群和 YARN 集群的部署方案
Fig.3-7 The deployment plan of HDFS cluster and YARN cluster

3.3.3 数据展现接口

数据展现是数据分析的重要组成部分，通过展现，数据分析的结果才能更好地影响公众和城市决策者，从而提高城市环境保护的“智慧”程度。本平台设计了较为灵活的数据展现接口，包括暂存数据库和数据展现 API 层。数据展现 API 层中的服务器通过订阅 Kafka 主题、读取 HDFS 和读取暂存数据库来获得展现所需的数据并且提供接口给外部应用。数据暂存服务器比起 HDFS 更加轻量化，更加易于访问，MongoDB 比较适合保存异构程度较高的日志和警告等信息[40]，Redis

读写性能很高[41]，比较适合保存需要频繁访问的状态数据，大数据分析任务会将一部分结果保存在暂存服务器中，数据展现 API 服务器可以从这些暂存服务器中直接读取需要的数据，也可以利用这些暂存服务器缓存一些结果。MongoDB 和 Redis 在使用集群部署方式时，功能都会受到一定的限制，为了提高数据展现接口实现的灵活性，数据暂存数据库没有采用集群和高可用的设计。数据展现 API 服务器类似数据网关，是可扩展的，除了本平台已经实现的内置数据展现 API 以外，还可以根据具体需求编写自定义的数据展现 API 服务器。

本平台实现的内置数据展现 API 服务器由 Node.js 编写，是一个基本的数据展现 API，并通过提供第三方图形化 Web 库文件来提供有限的可视化支持。其主要功能是映射平台内的数据，并通过 HTTP API 和 socket.io[42]的方式提供静态数据结果和流式数据结果给外部应用。其接口协议见表 3-3。

表 3-3 本平台内置的数据展现 API
Table 3-3 The integrated data presentation API of the platform

协议	URL（方括号内是参数）	映射的数据或文件（方括号内是参数）
HTTP GET	/hdfs/[path]	HDFS 中路径为[path]的文件
HTTP GET	/mongo/[objectId]	MongoDB 中对象 ID 为[objectID]的文档
HTTP GET	/redis/[key]	Redis 中键为[key]的数据
socket.io	/kafka?topic=[topic]	Kafka 中的[topic]主题
HTTP GET	/lib/[scriptName]	接口库目录下文件名为[scriptName]的文件

该数据展现 API 开放了跨域访问，方便外部的 Web 应用直接集成本平台的数据。库目录下提供了 plotly.js 库作为基本的浏览器可视化支持，还提供了 socket.io 库用于浏览器长连接通信支持。根据具体需要，还可以开发自定义的数据展现 API，用于对暂存数据库进行复杂查询，增加服务端渲染功能等，从而增强本平台的数据展现能力。

3.4 运维支持系统的设计

运维支持系统的主要设计目的是部署服务、维护本平台的正常运作，环境数据有着数据量大、增长迅猛的特点，可以预期在平台的实际运行中会逐步遇到扩容的需求，随着机器的增多，监控和维护的需求也会越来越高，因而，运维支持系统的设计将关系到本平台的长期实用性。

Ambari 是 Apache 旗下的开源项目，Hortonworks 向 Ambari 贡献了大量的代

码，并且将自身的商用软件栈 HDP 以开源形式集成在 Ambari 中，其中包含了本平台需要的重要基础服务，例如 ZooKeeper，HDFS，YARN 和 Kafka。因此，这些基础服务都可以很好地通过 Ambari 进行部署、运维、监控和管理。平台中还有一些服务器，不能被 Ambari 很好的管理，例如数据网关服务器和数据展现 API 服务器，我们可以采用 SaltStack[43]对这些机器进行自动化的运维。

Ambari 通过在节点上安装 ambari-agent 实现集群运维的功能，安装 Ambari Server 的机器为主节点，通过 Web 可以管理所有安装了 ambari-agent 的机器。类似地，SaltStack 也通过在节点上运行 salt-minion 服务实现运维功能，安装了 salt-master 服务的节点为主节点，可以在该节点上执行集群的各种运维命令。但 SaltStack 的通用性更强，还可以用于数据网关服务器、数据展现 API 服务器等。

本平台中，一台独立的管理服务器用于同时部署 Ambari Server 和 salt-master。Ambari-agent 被部署到 HDFS 集群，YARN 集群，Kafka 集群和 ZooKeeper 集群所属的机器上，salt-minion 被部署到除了管理服务器以外的所有机器上。

3.5 应用案例“区域空气 PM2.5 污染 24 小时预报”的设计

为了验证和评估本平台支持智慧环保应用的能力，本文从智慧环保的实际需求和现实应用场景出发，选择了智慧环保的一个典型应用案例“区域空气 PM2.5 污染 24 小时预报”，对其在本平台上的实现进行了设计。该案例涉及了“智慧环保”相关的大数据整合和分析的需求，并使用了本平台的主要技术，具有一定的代表性，能够充分验证本平台的数据整合和分析的功能。

智慧环保场景中一类重要的应用是对环境污染做出预测，通过对环境污染做出预测，城市可以提醒公众对即将到来的重污染提前做好防护措施，以期将环境污染带来的影响最小化。应用案例“区域空气 PM2.5 污染 24 小时预报”就是在这一背景下对 PM2.5 进行预报的典型案例，该应用需要获取与区域环境污染高度相关的数据，包括区域工业企业数量，日期，小时数，星期几，前三日同小时的 PM2.5 数值和天气，与同时间的 PM2.5 值建立起关联模型，并以此做出 24 小时后的预报，预报将以小时为时间间隔进行更新，并以图表的形式进行可视化。总结来说，该应用整体的数据流图如图 3-8 所示。

该应用案例的典型应用场景如下：包括标准行业设备和专属配套设备在内的环境监测设备会上传按小时采样的 PM2.5 数据，这些数据会被保存为历史数据，工业企业数量信息通常存储在外部的 MySQL 数据库中，天气信息通常需要调用第三方天气服务器的 API 来获取，最终用户会通过 Web 页面来查看未来 24 小时

的 PM2.5 预报。

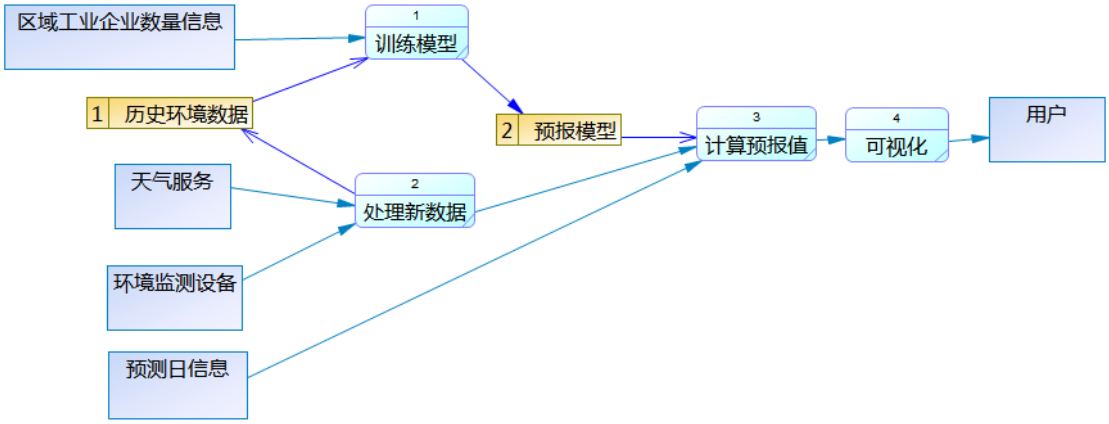

图 3-8 应用案例“区域空气 PM2.5 污染 24 小时预报”的数据流图

Fig.3-8 Data flow diagram of the application case “regional PM2.5 air pollution 24-hour forecast”

考虑到应用场景，本文对该应用案例在本平台上的实现进行了设计，其实现方式以及与本平台交互的方式如图 3-9 所示，绿框中的部分是该应用需要实现的部分，通过虚线箭头指向这些部分在整个交互流程中的位置，其他部分则是由本平台提供的功能。

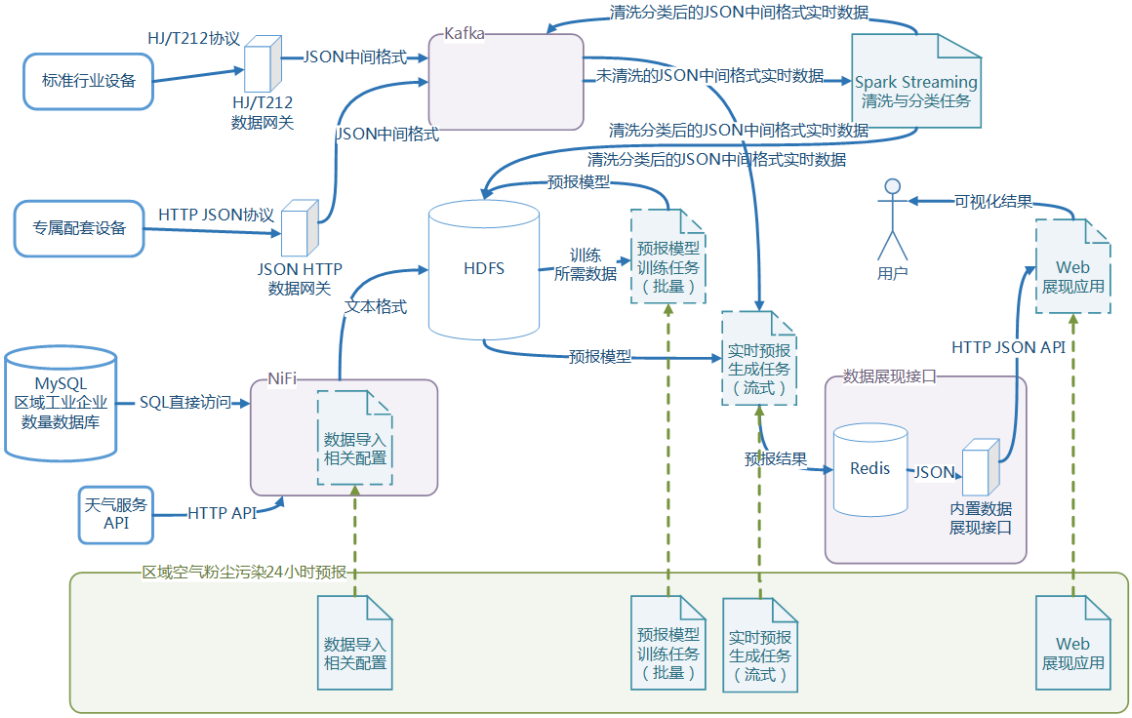

图 3-9 应用案例“区域空气 PM2.5 污染 24 小时预报”的设计

Fig.3-9 Design of the application case “regional PM2.5 air pollution 24-hour forecast”

在该应用案例的应用场景中，来自标准行业设备和专属配套设备的不同协议的实时数据由数据网关转换成基于 JSON 的环境数据中间格式，并发送到 Kafka 消息队列。之后这些数据经过平台的清洗与分类任务处理后，被发送回 Kafka 消息队列中相应的主题，同时存储到 HDFS 上成为历史数据的一部分。另一方面，通过配置 NiFi，来自 MySQL 的区域工业企业数量数据和来自天气服务 API 的天气数据被导入 HDFS。该应用的预报模型训练任务基于 Spark，采用手动执行的方式，采用随机森林模型[44]，从 HDFS 中获取历史数据、区域工业企业数量数据和天气数据，加上日期相关数据，共同作为输入向量进行训练，生成的模型保存在 HDFS 上。在平时，实时数据会被输入该应用中基于 Spark Streaming 的实时预报生成任务，该任务会从 HDFS 中读取定期训练的模型，并产生实时的 24 小时后的区域 PM2.5 预报，结果写入 Redis 暂存数据库。最后，该应用的 Web 展现应用会访问本平台内置的数据展现 API 获取 Redis 中保存的预报数据，并展示给用户。

3.6 本章小结

本章首先介绍了本平台的总体设计，包括其主要组成部分和功能，然后分三个部分，对数据整合平台、数据分析平台和运维支持系统的设计进行了介绍。其中，数据整合平台考虑了环境数据中适合标准化的设备监测实时数据和其他不适合标准化的复杂环境相关数据，并分别对其整合方法做出了设计。数据分析平台则主要考虑了大数据存储平台和大数据计算平台的搭建方案以及数据展现接口的设计。而运维支持系统的设计则主要是针对需要进行运维操作的机器选择了适合的运维工具栈。最后，本章介绍了用于分析和评估本平台功能的典型智慧环保应用案例“区域空气 PM2.5 污染 24 小时预报”，并对其在本平台上的实现方式和与本平台的交互方式做出了设计。

第四章 平台运行环境的配置与实现

4.1 平台运行环境的配置与实现概述

为了充分验证本平台的可行性，研究在平台的实际部署中可能会遇到的问题，本文按照设计，配置了本平台的运行环境，对本平台进行了搭建，并对其中的 HJ/T212 数据网关，HTTP JSON 数据网关，基于 Spark Streaming 的清洗分类任务和内置数据展现 API 服务器等关键组成部分进行了实现。除此之外，为了充分评估本平台的功能和指标，本文还实现了运行在本平台上的应用案例“区域空气 PM2.5 污染 24 小时预报”，并生成了其测试数据。

4.2 平台运行环境的配置

4.2.1 服务器环境

为了模拟本平台在真实生产环境下的实现，该运行环境的环境配置方法力求保持与生产环境一致。即在真实生产环境中会采用集群和热备方案的系统，在该运行环境中也采用集群和热备方案。理论上，该运行环境与真实生产环境的区别只有集群中机器的数量和单个节点的硬件配置，配置该运行环境的方式也可以用于部署真实生产环境。

为了统一命名约定，该运行环境中系统和主机的命名统一采用“智慧环保整合与分析平台”（Consolidation and Analysis Platform for Smart Environment Protection）的英文缩写 CAPSEP 作为前缀。为了便于灵活地、高性价比地进行实验，本文选择在虚拟机上搭建运行环境，采用一台物理机作为虚拟机的宿主机，在其上运行基于 KVM[45]的开源虚拟机管理器 Proxmox Virtual Environment，软件版本为 4.2.2，其硬件配置如表 4-1 所示。在该宿主机上创建标准模板虚拟机，其软件版本和系统配置等软件环境如表 4-2 所示。通过该虚拟机模板创建运行环境相关的虚拟机，根据服务特点为虚拟机分配了一定的虚拟 CPU 和内存资源，其配置信息如表 4-3 所示。考虑到在实验过程中总的磁盘占用量不会超出宿主机硬盘容量，故除了 HDFS 数据节点以外的虚拟机磁盘统一分配为 32G，HDFS 数据节点的虚拟机磁盘分配为 96G，均采用不提前分配空间的方式以实现超出宿主机硬盘总容量的分配。此外，每台机器都分配了内网 IP 地址。

表 4-1 运行环境宿主机的硬件配置
Table 4-1 Hardware configuration of the host machine in the operating environment

硬件项目	硬件配置	说明
CPU	Intel Xeon L5639 * 2	共 2 颗，共 12 核，运行频率为 2.13Ghz
内存	48G DDR ECC REG	单条 4G，共 12 条
硬盘	480G SSD	SATA2 标准，连接板载 SATA 控制器
网络接口	Intel 82576	板载，全双工千兆，双 RJ45 端口

表 4-2 运行环境虚拟机模板软件环境
Table 4-2 Software environment of the virtual machine template in the operating environment

软件项目或配置项目	软件版本或配置值	说明
操作系统	CentOS 7	内核版本 3.10.0-514.2.2 .el7.x86_64
SELinux	Permissive	运行环境不强制 SELinux
firewalld 防火墙	关闭	运行环境关闭防火墙
Java 环境	JDK 8 update 111	稳定版本

表 4-3 运行环境虚拟机信息
Table 4-3 Virtual machines information in the operating environment

主机名	功能	vCPU	内存	内网 IP 地址
capsep-salt	运维主控机	1 核	2G	10.156.0.41
capsep-zk1	ZooKeeper 节点 HDFS QJM 日志节点	1 核	2G	10.156.0.42
capsep-zk2				10.156.0.43
capsep-zk3				10.156.0.44
capsep-kafka1	Kafka 中继节点	2 核	4G	10.156.0.45
capsep-kafka2				10.156.0.46
capsep-nifi1	NiFi 节点	2 核	2G	10.156.0.48
capsep-nifi2				10.156.0.49
capsep-hmaster1	HDFS 和 YARN 主控节点（热备）	2 核	2G	10.156.0.51
capsep-hmaster2				10.156.0.52
capsep-hnode1	HDFS 和 YARN 普通节点	4 核	6G	10.156.0.53
capsep-hnode2				10.156.0.54
capsep-hcalcnode1	YARN 纯计算节点	4 核	4G	10.156.0.55
capsep-dg1	HJ/T212 数据网关	1 核	1G	10.156.0.61
capsep-dg2	HTTP JSON 数据网关	1 核	1G	10.156.0.62
capsep-mongo1	MongoDB 暂存数据库	2 核	2G	10.156.0.71
capsep-redis1	Redis 暂存数据库	2 核	2G	10.156.0.72
capsep-di	内置数据展现 API	1 核	1G	10.156.0.73

运行环境的搭建、实现以及应用案例的实现涉及各类服务器软件和库，本文综合版本的稳定性，使用流行度，维护状态等进行考虑，选择了在该运行环境中采用的软件版本，如表 4-4 所示。

表 4-4 运行环境软件版本
Table 4-4 Software versions in the operating environment

软件	版本
Hortonworks HDP	2.4.3.0-227
Apache ZooKeeper	3.4.6.2.4
Apache Kafka	0.9.0.2.4
Apache HDFS	2.7.1.2.4
Apache Hadoop YARN	2.7.1.2.4
Apache NiFi	1.1.1
Apache Ambari	2.2.1.1
SaltStack	2016.11.1
MongoDB	3.2.11
Redis	2.8.19
Node.js	6.9.2
Node Version Manager	0.32.1
PM2	2.2.3
Spark	2.0.2

本地网络中的 DNS 服务器进行了配置，将机器名对应到表 4-3 的内网 IP 地址上。后续配置中一般采用主机名来访问这些服务器。

4.2.2 运维支持系统相关环境的配置

运维支持系统支持着本平台的部署和维护，本平台中关键服务器软件的部署依赖于运维支持系统，因而该运行环境的搭建将从运维支持系统开始，包括搭建 Ambari 和 SaltStack 两个运维工具的环境。

SaltStack 的受控端 salt-minion 在虚拟机模板中即已安装好，并且其配置中主控机的机器名被设置为 capsep-salt，从而使得受控端可以通过 DNS 找到主控机。通过在运维主控机 capsep-salt 机器上安装并运行 SaltStack 的主控端 salt-master，该运行环境中的所有机器即能找到主控端并发送登记的请求，在 capsep-salt 机器上通过执行 salt-key 命令信任这些登记信息后，就可以从 capsep-salt 机器上使用 SaltStack 相关的命令统一管理这些机器。

Ambari 主要用于管理由 Hortonworks HDP 提供的 Hadoop 相关的服务器软件，Ambari 服务器同样安装在 capsep-salt 机器上，Ambari 服务器可以通过 ssh 连接到集群中的其他机器上并自动安装 Ambari 的受控端 ambari-agent。本文生成了管理用的密钥对，并使用 SaltStack 向集群中的其他机器同步 ssh 公钥信任列表，再将私钥导入 Ambari，从而使 Ambari 可以管理集群中的相关机器。

总结来说，该运行环境中运维支持系统的部署方式如图 4-1 所示，Kafka、ZooKeeper、HDFS 和 YARN 服务器同时由 Ambari 和 SaltStack 管理，其他机器仅由 SaltStack 管理。

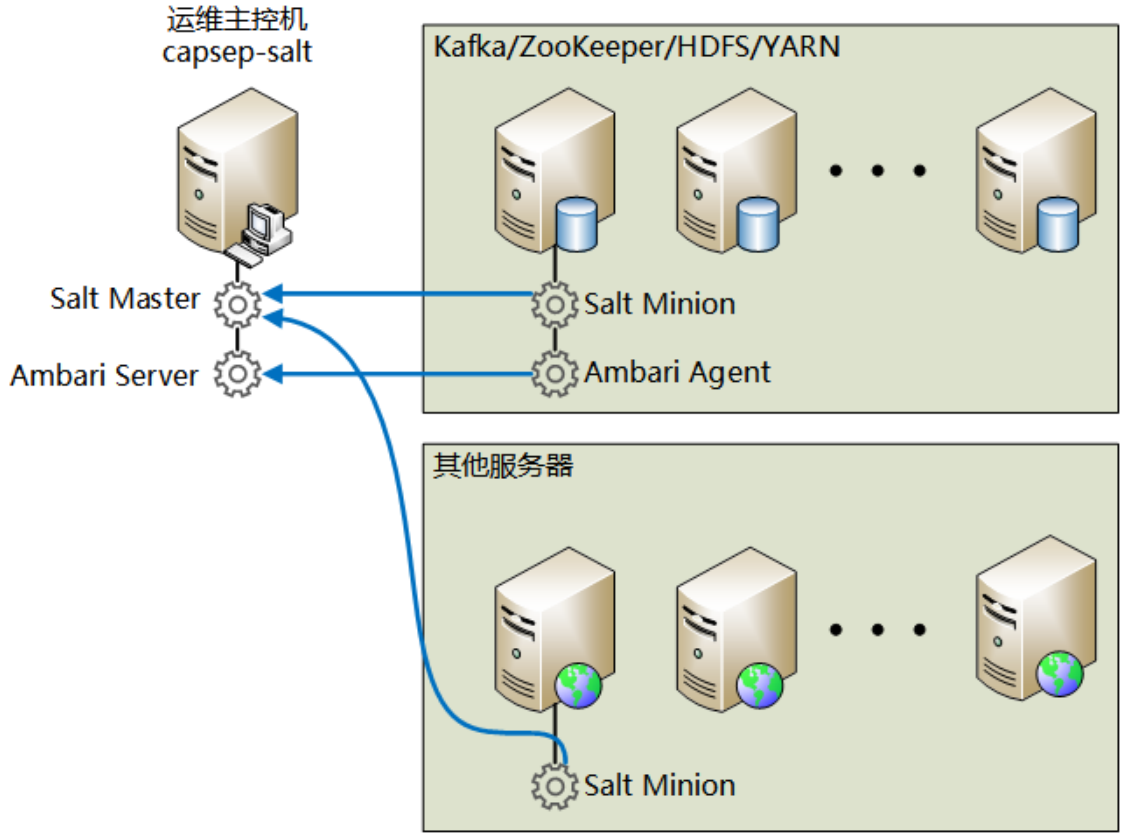

图 4-1 运维支持系统的部署
Fig.4-1 Deployment of the operations and support system

4.2.3 数据整合与分析相关服务器软件的部署

数据整合平台和数据分析平台的搭建涉及多种服务器软件的部署，包括 Kafka 集群，HDFS、YARN、ZooKeeper、NiFi、MongoDB 和 Redis。除此之外，还要为数据网关服务器配置 Node.js 运行环境。

其中，HDFS、YARN、ZooKeeper 和 Kafka 在 Ambari 提供的服务器软件范围内，由 Ambari 直接进行部署。首先创建集群，然后按照表 4-5 配置 Ambari 以部署这些服务器软件并设置高可用。需要注意的是，HDFS 的 DataNode 和 YARN 的 NodeManager 在普通节点上同位部署，在纯计算节点上则只部署 NodeManager。此外，ZooKeeper 节点也和 HDFS JournalNode 同位部署。由于只有两个 DataNode 节点，故在该运行环境中默认的块复制数被设置为 2，在真实生产系统中应当设置成 3 或者以上的数值以提高可用性。类似地，为了保证高可用性，Kafka 的默认消息复制比也被设置为 2。

表 4-5 针对 HDFS，YARN，ZooKeeper 和 Kafka 的部署方案
Table 4-5 Deployment plan of HDFS, YARN, ZooKeeper and Kafka

服务器软件	角色	部署虚拟机主机名
HDFS	NameNode（主备）	capsep-hmaster1
		capsep-hmaster2
	JournalNode	capsep-zk1
		capsep-zk2
		capsep-zk3
	DataNode	capsep-hnode1
		capsep-hnode2
YARN	ResourceManager（主备）	capsep-hmaster1
		capsep-hmaster2
	NodeManager	capsep-hnode1
		capsep-hnode2
		capsep-hcalcnode1
	App Timeline Server	capsep-hmaster1
ZooKeeper	ZooKeeper Server	capsep-zk1
		capsep-zk2
		capsep-zk3
Kafka	Broker	capsep-kafka1
		capsep-kafka2

成功部署后各个服务器会通过 Ambari Metrics 报告运行状态，从而可以在 Ambari 中对这些服务器软件进行监控，如图 4-2 所示，可以看到两个数据 DataNode 和三个 NodeManager 已经正确启动。

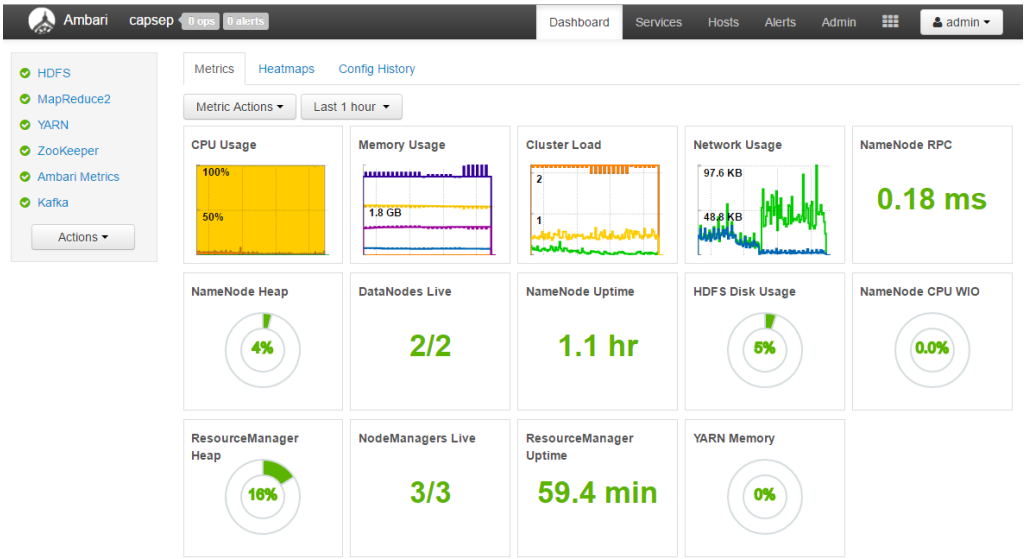

图 4-2 Ambari 管理的集群监控图表截图
Fig.4-2 Screenshot of monitor metrics of Ambari managed cluster

NiFi 软件并未包含在 Ambari 的套件中，故采用自行下载的二进制文件进行部署， 采用 SaltStack 管理配置。NiFi 采用集群部署，安装并运行在 capsep-nifi1 和

capsep-nifi2 两台机器上，通过连接到 ZooKeeper 集群来管理共享的状态。

MongoDB 和 Redis 作为数据展现接口的暂存服务器，没有采用集群部署的方式。MongoDB 和 Redis 均使用 CentOS 的包管理器 yum 从官方仓库获取包进行单独安装。MongoDB 服务器部署在 capsep-mongo1 机器上，Redis 服务器部署在 capsep-redis1 机器上。

Node.js 运行环境由 Node Version Manager 搭建，该工具可以方便地管理 Node.js 的版本，通过直接下载安装脚本安装。Node.js 服务器进程由 PM2 进程监视器进行监控和守护，PM2 通过 Node.js 自带的依赖管理器 npm 进行安装。这些环境部署在 capsep-dg1、capsep-dg2 和 capsep-di 机器上。

4.3 平台关键系统的实现

4.3.1 数据网关的实现

为了将物联网环境设备的自有格式转换成本平台的环境数据标准中间格式，本平台设计了数据网关层，数据网关层中包含若干数据网关服务器。本文设计并实现了两种平台自带的数据网关服务器，即 HJ/T212 协议数据网关和 HTTP JSON 协议数据网关，由 Node.js 语言编写。数据网关向 Kafka 消息队列传输数据的方式和格式是统一的，故这两种数据网关可以通过统一的方式来编写，通过配置来切换实际运行的数据网关类型。

其类图如图 4-3 所示，需要特别说明的是，Node.js 采用 JavaScript 编写，JavaScript 本身采用原型模式，其类结构的实现方法与传统面向对象语言不同，因此图 4-3 中的继承关系采用 JavaScript 的原型链方式进行实现，接口和简单对象采用文档约定的方式实现，私有和保护成员采用名称约定的方式实现。实际运行时，通过不同的入口文件，从 HttpListener 的 listen 启动时，启动的即为 HTTP JSON 协议的数据网关，从 StreamListener 的 listen 启动时即为 HJ/T212 协议的数据网关。

内置数据网关采用了多级抽象，DataUploader 类为向平台内上传数据的接口，按照本平台的设计，数据网关会将数据发送到 Kafka 消息队列，因而这一任务由 DataUploader 的具体实现 KafkaUploader 完成。EnvData 类和 DataItem 类是标准中间数据格式的实现。DataHandler 类是抽象类，是标准中间数据后续处理的接口和部分实现，其保持一个对 DataUploader 实例的引用，对中间数据格式进行处理后将通过这个实例上传数据，数据上传采用不等待返回值的方式，因而 DataHandler 和 DataUploader 均不设置异步回调。StreamHandler 是通用的 TCP 流处理类，其

接收回调，并将流放入内部的缓冲区，利用连接对象_conn 保持会话信息，并用于返回回复数据。Hjt212StreamHandler 是处理 HJ/T212 协议的具体实现，其会将 HJ/T212 的字符串包从字符流中拆分出来进行解析，产生 Hjt212Packet 格式，并进一步转化为标准中间格式。HttpHandler 是通用的 HTTP 请求处理类，其接收回调，处理请求并回复，HttpJsonHandler 即为处理 HTTP JSON 协议的具体实现，其会接收以 HTTP POST 方式发送的 JSON 中间格式并直接发送到 DataHandler 进行处理。StreamListener 和 HttpListener 分别是 TCP 流服务器和 HTTP 服务器的基础实现，StreamListener 主要负责维护管理会话，接受连接，并将数据转发给对应的 StreamHandler，HttpListener 以 Hapi.js 库为基础，实现基础的 HTTP 服务器框架，从 HttpHandler 的 getRoute 方法获得具体监听的 URL，并将该 URL 上收到的数据转发过去，以回调的形式接收回复数据。

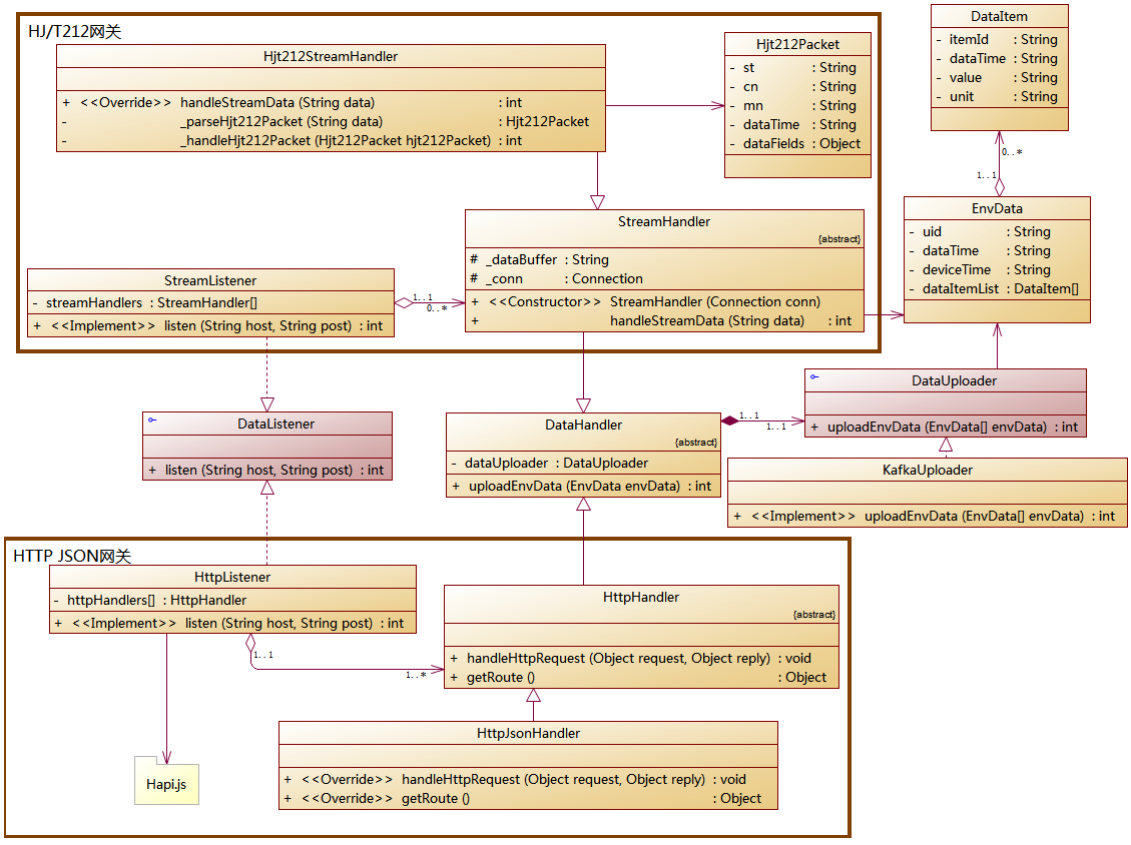

图 4-3 内置数据网关实现的类图

Fig.4-3 Class diagram of integrated data gateways implementation

按照设计，本文所实现的数据网关中有多个可以复用的组件，可以用于快速地实现新的基于 HTTP 或者 TCP 流的数据网关。

在该运行环境中，HJ/T212 数据网关部署在 capsep-dg1 机器上，HTTP JSON

数据网关部署在 capsep-dg2 机器上。

4.3.2 基于 Spark Streaming 的环境数据清洗分类任务的实现

为了对环境数据进行清洗与分类，本平台设计并实现了一个基于 Spark Streaming 的环境数据清洗分类任务。该任务的具体实现流程如图 4-4 所示。

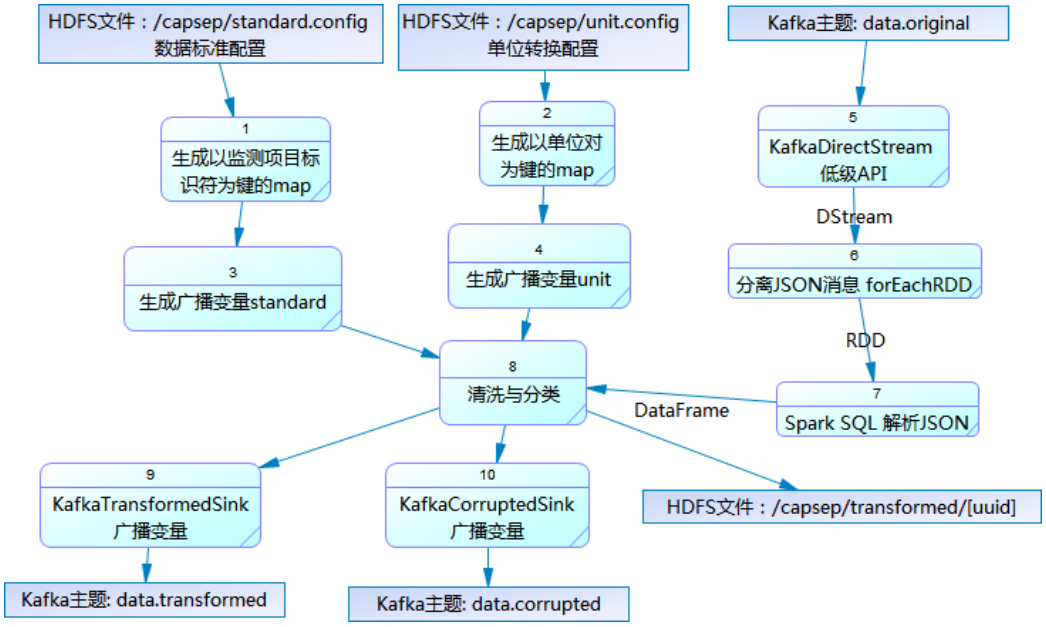

图 4-4 基于 Spark Streaming 的环境数据清洗分类任务数据流图

Fig.4-4 Data flow diagram of environmental data transformation job based on Spark Streaming

原始数据由数据网关发送至 data.original 主题，为了提高并行效率，本任务采用 KafkaDirectStream 低级 API 读取原始数据，并设置为从当前最新时间开始读取，读取的原始数据分离出 JSON 信息，转换为 RDD，再进一步通过 Spark SQL 解析为 DataFrame。另一方面，保存有环境数据标准的 standard.config 和保存着单位转换规则的文件 unit.config 被读取。这两个文件均为空格分隔的文本文件，对其分别生成 map，并且赋值给广播变量发送到每个执行节点上，用于后续的清洗与分类操作。清洗分类操作中，原始数据的 DataFrame 经过扁平化，从而对原始数据 DataItem 项下的每一项数据进行验证和转换，正确的数据将发送给 data.transformed 主题，错误的数据将发送给 data.corrupted 主题。为了避免 KafkaProducer 在执行节点上的频繁创建，本任务的实现对 KafkaProducer 进行了包装，并赋值给两个广播变量 KafkaTransformedSink 和 KafkaCorruptedSink，这两个广播变量在内部调用 KafkaProducer 将数据分别发送给 data.transformed 主题和 data.corrupted 主题。

Spark Streaming 的实现方式实际是连续的小批量处理，为了平衡性能和延迟，本任务采用 10 秒为间隔对流进行批量处理。

4.3.3 内置数据展现 API 服务器的实现

为了将计算得到的结果进行展示，本平台设计了数据展现 API 层，数据展现 API 层中包含若干数据展现 API 服务器，其主要作用是从平台内的各个服务器获取数据，并将其暴露给外部应用。本文设计并采用 Node.js 实现了一个平台内置的数据展现 API 服务器。其能够将平台内的 MongoDB、Redis 和 HDFS 中的数据通过 HTTP 接口暴露给外部应用，也能将平台内的 Kafka 消息队列中的特定主题的流式数据通过 socket.io 协议的接口暴露出来，此外，该服务器还能够服务静态文件，可以以 HTTP 方式提供相关的 JavaScript 库文件，外部的 Web 应用可以直接引入这些库文件，使可视化更加简便。

该数据展现 API 服务器的类图如图 4-5 所示，HTTPListener 同样以 Hapi.js 库为基础，具体监听的 URL 和控制逻辑由 DataInterfaceController 类管理。访问 HDFS、MongoDB、Redis 的请求分别交由 HdfsDao 类、MongoDbDao 类、RedisDao 类执行实际的存取操作。它们分别通过不同的第三方驱动库 webhdfs、redis 和 node-mongodb-native 等对目标数据进行访问。访问静态文件的请求则由 StaticFileServer 类进行处理，该类会根据指定的文件名读取 lib 目录下的静态文件。在当前的实现中，该内置数据展现 API 服务器主要提供了两个库文件，一个是用于与该 API 服务器进行长连接通信的 socket.io.js，另一个是用于在浏览器中绘制图表的 plotly.js 库。

针对 Kafka 的流式数据传输接口较为特别，首先，在 DataInterfaceController 类中通过设置插件，将访问 Kafka 数据对应的 URL 转交 socket.io 进行处理。当 socket.io 的客户端请求连接的时候，会附上一个参数表明其要监听的主题。该 API 服务器接收到连接请求后，会根据这个参数来监听对应的主题。其具体实现方式是通过 KafkaStreamConsumerManager 类添加一个包装了 Kafka 的消费者的类 KafkaStreamConsumer 的实例，将该连接的会话保存在该实例中。该实例内部通过 kafka-node 库转发该主题上的消息，具体的转发方式是设置_onMessage 回调，通过该回调接收消息并通过该实例中存储的会话_socket 发送给对应的 socket.io 客户端。所有 KafkaStreamConsumer 的实例被加入一个全局列表进行管理，当连接中断或错误时_onError 回调被触发，接到回调的实例会停止转发消息，并被移出全局列表。

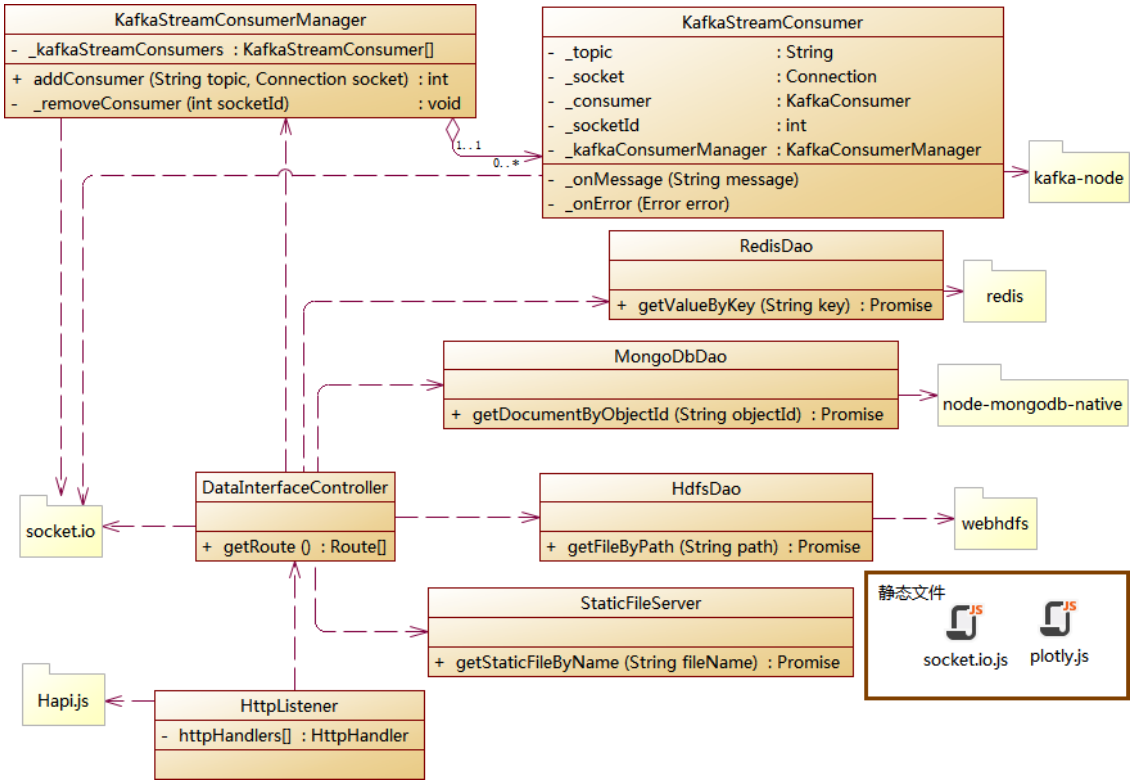

图 4-5 内置数据展现 API 服务器实现的类图
Fig.4-5 Class diagram of integrated data presentation API server

在该运行环境中，该内置数据展现 API 服务器部署在 capsep-di 机器上。

4.4 应用案例“区域空气 PM2.5 污染 24 小时预报”的实现

4.4.1 应用案例总体实现方案

为了在该运行环境中再现该应用案例的真实应用场景，本文需要对环境监测设备、第三方数据库和第三方应用系统等应用环境进行模拟。同时，由于该运行环境中没有真实设备和系统的数据，本文也需要对这些数据进行生成。

在该应用的实现过程中，MySQL 数据库和 HTTP API 数据的整合将主要通过配置 NiFi 实现，经过整合后的相关数据将由一个手动运行的基于 Spark 的预报模型训练任务来处理，生成的模型将被一个长期运行的基于 Spark Streaming 的实时预报生成任务读取，并与实时数据相结合，产生预报结果存入 Redis 中。最后该应用通过一个 Web 应用访问本平台的内置数据展现 API，从 Redis 中取出预报结果，并在网页上显示图表来实现可视化。

4.4.2 应用场景的模拟和测试数据的生成

该应用案例的实际应用场景如下：

某正在试点智慧环保的城市设置了若干区域环境监测点，其中有一部分监测点采用标准环保行业设备，使用 HJ/T212 协议上传 PM2.5 数据，还有一部分监测点采用专属配套设备，使用 HTTP JSON 协议上传 PM2.5 数据，所有监测点上传的数据均为小时数据，每个小时上传一次。各个监测点的设备已经运行了一年，HDFS 中已经保存了这些监测点 2015 年的完整监测数据。一个有关部门的 MySQL 数据库中保存有这些监测点监测区域中工业企业数量的数据，工业企业的数量没有随时间变化。一个第三方天气服务以 HTTP API 的形式提供 24 小时后的天气预报和当前天气。现在有关部门希望实时获得这这些区域监测点未来 24 小时的污染浓度预报走势图表，以便提前向公众发布重污染预警。

因此，为了在该运行环境中再现该应用案例的应用环境，本文需要模拟的系统主要有两种环境监测设备、区域工业企业数量数据库和第三方天气服务。同时，本文还需要生成对应的测试数据，包括 PM2.5 历史数据、天气历史数据、PM2.5 实时数据、天气实时数据和区域工业企业数量数据。

其中环境监测设备的模拟使用 Node.js 简易脚本实现，模拟两种各 1 台环境监测设备，从 JSON 文件中读取预先准备好的数据并分别以 HJ/T212 协议和 HTTP JSON 协议上传给数据网关。本文搭建了模拟的 MySQL 区域工业企业数量数据库，其中包含一个表 region_company，其表结构如表 4-6 所示，数据通过人工进行填写。模拟的天气服务 API 为 Node.js 实现的简易服务器，共包括两个 API，如表 4-7 所示，天气代码的范围是 0 至 5，本文模拟的天气服务返回值呈现以 6 天为周期的循环，为了便于测试本文只选择了晴雨天气作为返回值。天气代码 0-5 对应六种天气，分别是晴、多云、阴、小雨、中雨和大雨。这些模拟系统均运行在一台额外的测试虚拟机上，其机器名为 capsep-test。

表 4-6 区域工业企业数量表 region_company 的结构
Table 4-6 Schema of regional industrial company table region_company

列名	数据类型	说明
id	int	序列号
uuid	char(20)	该区域部署的监测点设备唯一标识符
name	varchar(32)	区域名称
count	int	该区域工业企业的数量

表 4-7 模拟天气 API 的参数
Table 4-7 Parameters of simulated weather API

相对地址	返回值说明
/getWeather	JSON,{ "time":[ISO8601 时刻字符串], "weather":[当日天气代码]}
/getWeatherForecast	JSON,{"time":[ISO8601 时刻字符串], "weather":[明日天气代码]}

为了使测试数据本身呈现一定的规律和动态趋势，测试数据采用周期性均值、天气和工业企业相关修正值、随机值多个值相加的方式生成，具体来说，模拟的 PM2.5 数据由日基础值、小时基础值、天气修正值、工业企业数量修正值和随机修正值这几个部分相加而成。其中小时基础值设定为每日 19 点最高 36μg/m^3，每小时下降 3μg/m^3，至早 7 点下降为 0μg/m^3 并变为每小时上升 3μg/m^3，工业企业数量修正值设定为工业企业数量的 1 倍，随机修正值设定为在-10μg/m^3 至+10μg/m^3 之间均匀分布。日基础值如表 4-8 所示，模拟真实环境，呈现出工作日高休息日低的特点。

表 4-8 模拟数据日基础值
Table 4-8 Day-related base value for simulated data

周一	周二	周三	周四	周五	周六	周日
20μg/m^3	30μg/m^3	40μg/m^3	50μg/m^3	40μg/m^3	15μg/m^3	15μg/m^3

通常来说降水能够降低 PM2.5 的浓度，据此设计天气修正值如表 4-9 所示。

表 4-9 模拟数据天气修正值
Table 4-9 Weather-related base value for simulated data

晴	多云	阴	小雨	中雨	大雨
+20μg/m^3	+5μg/m^3	0μg/m^3	-5μg/m^3	-10μg/m^3	-20μg/m^3

本文还生成了设备信息和区域工业企业数量数据，如表 4-10。

表 4-10 区域工业企业数量表 region_company 的模拟数据
Table 4-10 Simulated data of regional industrial company table region_company

id	uuid	name	count
1	10001	District1	8
2	10002	District2	20

根据前文说明的数据生成算法，本文使用 Node.js 简易脚本生成了 2015 年全年的数据并导入 HDFS 作为历史数据。还生成了 2016 年前七日数据作为模拟环境

中设备上传的数据用于案例分析。

4.4.3 区域工业企业数据和天气数据的整合

区域工业企业数据存放在 MySQL 数据库中，天气服务数据位于 HTTP API 中，这两项数据属于不适合标准化的环境数据，故使用本平台的 NiFi 进行整合。

应用案例的实现采用 NiFi 内置的处理器 QueryDatabaseTable 通过连接池 DBCPConnectionPool 获取测试 MySQL 数据库中 region_company 整表的数据，采取每日定时触发的方式。对天气服务 API 则采用 GetHTTP 处理器进行调用，采取每小时定时触发的方式。数据将使用 PutHDFS 处理器存入 HDFS 中。

配置后，在处理器之间加上适当的连接，如图 4-6 所示，QueryDatabaseTable 将定时拉取数据库内容，并序列化为二进制的 Avro[46]格式保存进 HDFS，GetHTTP 也会获取天气数据，并以 JSON 文本格式保存进 HDFS。

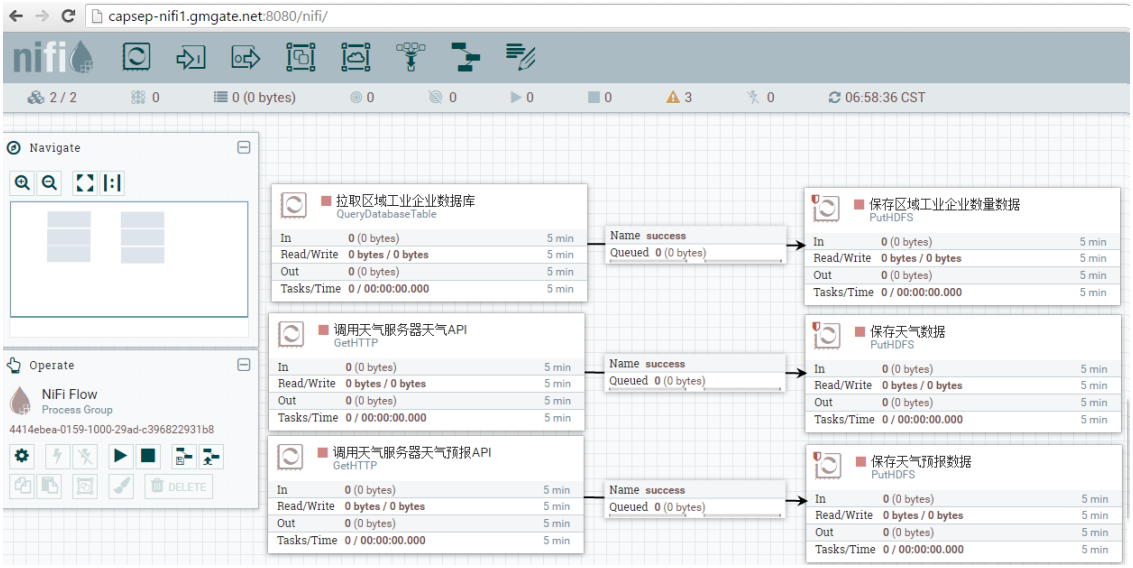

图 4-6 应用案例 NiFi 整合数据的配置

Fig.4-6 Data consolidation configuration in NiFi for the application case

4.4.4 基于 Spark 的预报模型训练任务的实现

收集到足够的数据后，需要使用基于 Spark 的预报模型训练任务来产生预报模型，本文为应用案例选择了随机森林回归模型。随机森林模型是决策树模型的一种组合。随机森林模型会产生很多相互之间没有关联的随机决策树，分别对输入进行分类，然后取结果的众数。随机森林模型具有训练和预测速度快，能处理很高维度的数据等特点。

经过本平台的整合，与 PM2.5 高度相关的数据已经保存在 HDFS 上，本任务

将区域工业企业数量、天气、基本的时间信息和前三日同小时的 PM2.5 数据合并成一个特征向量，以同时间的 PM2.5 值为回归目标值，建立随机森林回归模型。考虑到日期和天气都会作为离散变量处理，采用的超参数如表 4-11 所示。

表 4-11 随机森林模型采用的超参数
Table 4-11 Hyper-parameters for random forest model

超参数	值	说明
maxBins	32	特征最大装箱数
maxDepth	5	最大深度
numTrees	20	决策树个数

具体实现的数据流如图 4-7 所示，首先将整合的相关信息合并为一个多列的 DataFrame，然后生成特征向量列和标签列。这里特征向量列包含日期、星期几、小时数、天气编号，区域工业企业数量和前三日同小时的 PM2.5 数据，其中日期、星期几和天气编号为离散变量，区域工业企业数量、小时数和前三日同小时的 PM2.5 数据为连续变量。之后将其按照 7：3 的比例划分为训练集和交叉检验集，训练集用于训练并得到模型结果，模型结果应用在交叉检验集上进行预测，并计算均方根误差，根据误差进行人工调优。最后将预测结果较优的模型保存到 HDFS 上用于实际的预测工作。本应用主要用于验证本平台的适用性，因而没有进行进一步的人工调优。

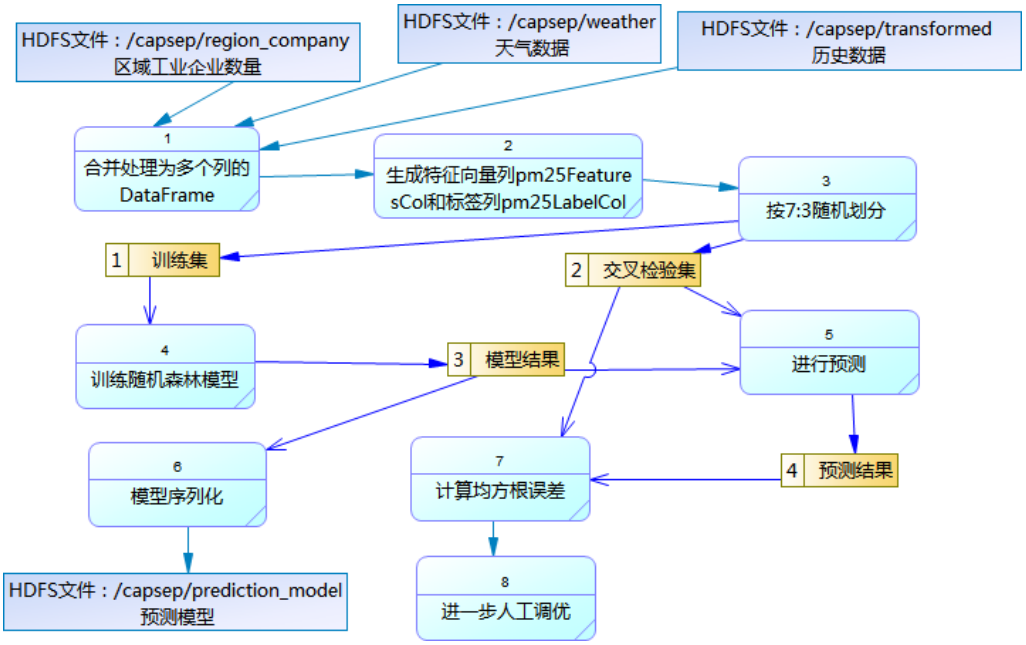

图 4-7 基于 Spark 的预报模型训练任务的数据流图
Fig.4-7 Data flow diagram of Spark-based prediction model training job

4.4.5 基于 Spark Streaming 的实时预报生成任务的实现

通过训练获得一个预测模型并保存下来后，就可以利用这个模型产生实时预测，每当有新的数据进入平台，就会在经过清洗、分类后被发布到 Kafka 消息队列的 data.transformed 主题中。通过基于 Spark Streaming 的任务可以从该主题中取出最新的实时数据，并与历史数据和天气预报合并作为预测模型的输入，从而产生 24 小时后的预测结果。

其基本流程的数据流图如图 4-8 所示，首先将整合的信息合并为一个 DataFrame，然后从中取出和模型训练时相同的列构成特征向量，从 HDFS 中读出序列化的模型并且反序列化，利用该模型和特征向量产生预测，并直接以监测点唯一标识符加字符串.pm25forecast 为键，将预测结果存入 Redis 数据库的一个 List 结构中， 该 List 大小封顶，只存放最近的 30 个值，即每次执行完 PUSH 命令后都会使用 LTRIM 命令删掉多余的值。

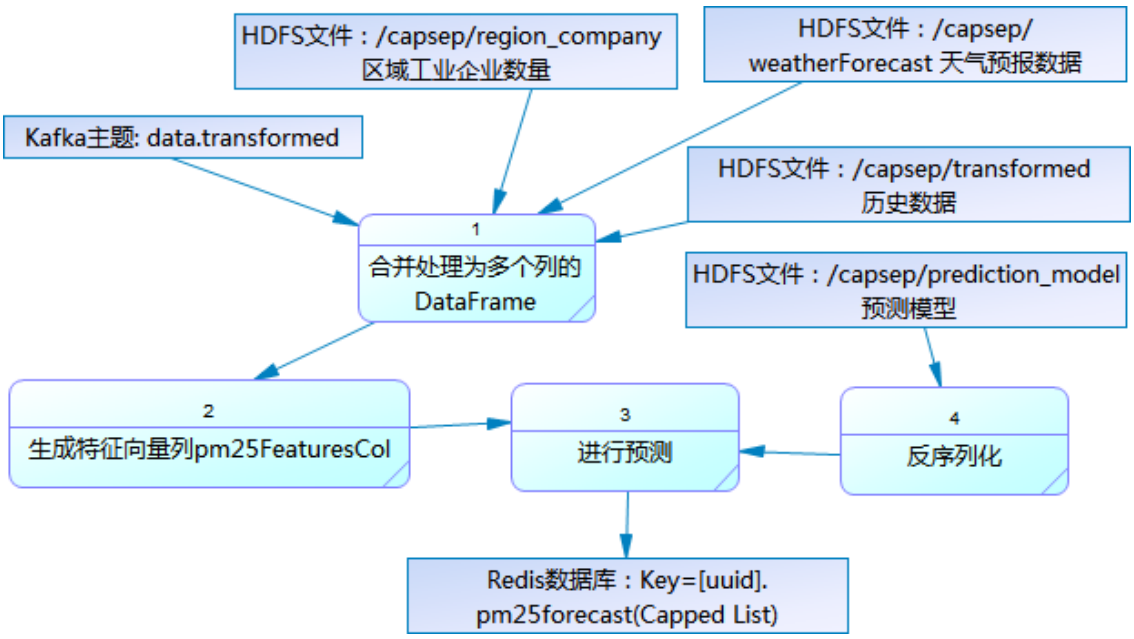

图 4-8 基于 Spark Streaming 的实时预报生成任务的数据流图
Fig.4-8 Data flow diagram of real-time prediction job based on Spark Streaming

4.4.6 预报可视化的实现

经过实时任务的处理，未来 24 小时的预报数据已经存放在 Redis 暂存数据库中，只需编写一个 Web 应用从本平台的内置数据展现 API 中读取该数据并绘制图像即可。

具体来说，该应用的预报可视化部分采用基于 HTML 和 JavaScript 的 Web 应

用来实现，通过地址“http://capsep-di/lib/plotly.js”引入了本平台内置数据展现 API 服务器的图形化库，通过访问地址“http://capsep-di/redis/[uuid].pm25forecast”获得了预报 PM2.5 的数据序列并结合当前时间画图，其展示效果如图 4-9。

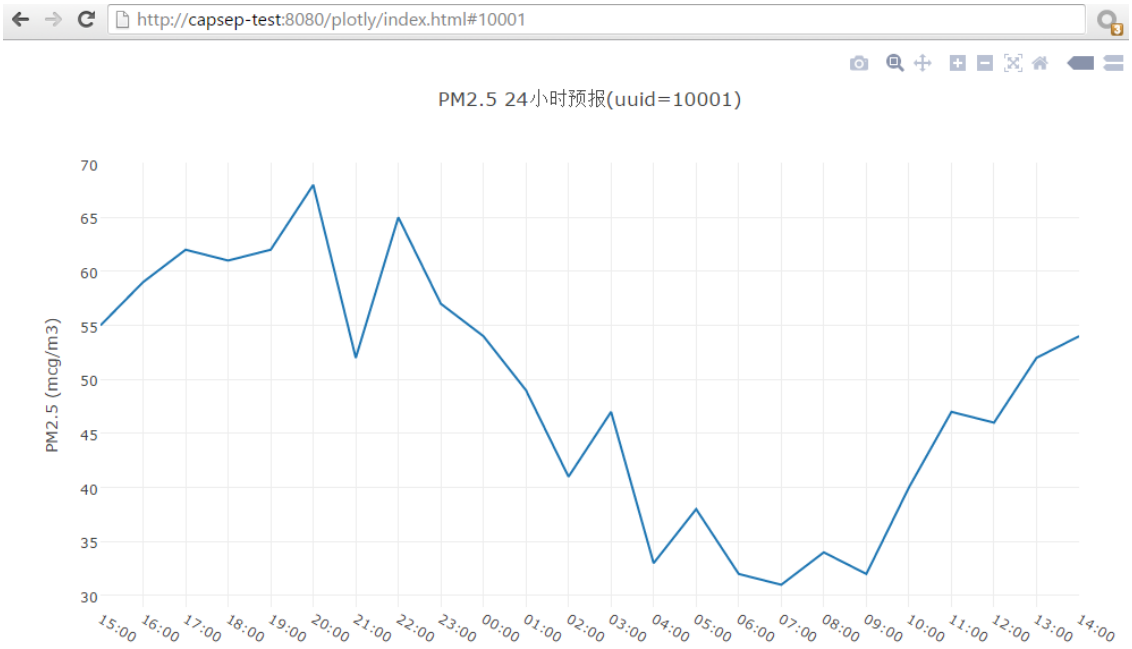

图 4-9 应用案例 PM2.5 预报的可视化效果
Fig.4-9 Visualization style of the PM2.5 prediction in the application case

4.5 本章小结

本章主要介绍了本平台及其应用案例的实现细节。首先介绍了本平台运行环境的配置，在虚拟机上通过搭建运行环境模拟了本平台在真实生产环境下的搭建方法，包括其运维系统的搭建和相关服务器软件的部署。然后介绍了本平台中关键系统的实现方法，即数据网关、基于 Spark Streaming 的清洗与分类任务和内置数据展现 API 的实现方法。最后介绍了用于分析和评估本平台功能的应用案例的实现，包括其总体实现方案、应用环境的模拟、测试数据的生成方法、数据的整合方法、预报模型的训练方法、实时预报的生成方法以及预报可视化的实现方法。

第五章 分析与评估

5.1 应用案例“区域空气 PM2.5 污染 24 小时预报”的案例分析

5.1.1 案例分析的意义

本平台的设计初衷是通过实现智慧环保中技术难度较高、功能较繁杂但通用性较强的环境数据整合和分析功能，从而支撑智慧环保相关应用的开发，进而助力智慧环保的实施。其主要设计目标包括为智慧环保相关应用提供环境数据的整合和分析功能。其中，环境数据的整合功能既包括整合环境监测设备上传的实时环境数据，也包括整合其他第三方数据库和应用系统中的非实时的环境相关数据。环境数据的分析既包括流式数据的分析，也包括批量数据的分析，还包括数据的展现。

为了评估本平台是否达到设计目标，本文设计并实现了智慧环保的典型应用案例“区域空气 PM2.5 污染 24 小时预报”，为了在该运行环境中再现其实际应用场景，本文模拟了其应用场景并生成了相关的测试数据。该应用案例涉及整合多种设备的历史数据和实时数据，同时涉及对数据进行批量分析和流式分析，最后还包含分析结果的展现，具有较高的典型性和代表性。通过评估本平台对该应用案例的适用性，可以有效地评估本平台支持智慧环保相关应用进行环境大数据整合和分析的功能。

5.1.2 通过案例分析评估数据整合功能

在该应用案例中，该应用需要获得当前监测点的历史监测数据用于预测模型的训练，还需要获得当前监测点的实时数据用于实时预报的产生。模拟应用环境中，与真实应用环境类似，两种使用不同协议的环境监测设备分别连接到不同的数据网关上。其中一种使用基于 HJ/T212 的字符流协议上传监测数据，另一种使用 HTTP JSON 协议直接提交本平台的环境数据标准中间格式。这两种设备虽然采用了不同的协议，但是经过本平台内置数据网关的整合后能够转化为统一的标准数据格式并存储在 HDFS 中成为历史数据，同时发送到 Kafka 相应的主题上成为实时数据。这些历史数据和实时数据能够被该应用读取、识别，并作为后续计算的重要输入数据。可见，本平台能够对设备上传的标准实时数据进行整合，满足该应用的需要。

除此之外，该应用还需要获得工业企业信息和天气信息。在模拟应用环境中，本文模拟了真实应用环境，工业企业信息存储于外部的一个 MySQL 数据库中，而天气信息则由第三方 API 提供，数据来源具有一定的复杂性。这两种数据虽然来源各异，但该应用可以通过配置 NiFi 集群对这两种数据进行整合，并存储在 HDFS 上，以用于后续的计算。由此可见，本平台能够对非实时的环境相关数据进行整合，满足该应用在这方面的需求。

综上，示意图如图 5-1 所示，本平台能够满足该应用整合实时环境数据和整合非实时环境相关数据的需求。总结来说，本平台能够较好地向该应用提供整合环境数据的功能，具备向智慧环保应用提供数据整合功能的能力，达到了支持环境数据整合的设计目标。

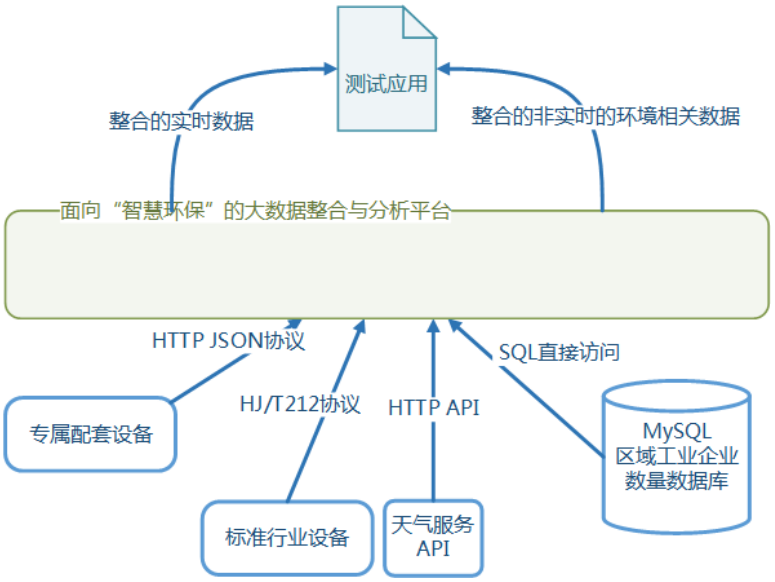

图 5-1 应用案例中数据整合的案例分析

Fig.5-1 Case study of data consolidation in the application case

5.1.3 通过案例分析评估数据分析功能

在该应用案例中，该应用需要运用整合后的批量历史数据训练一个用于后续预报的模型。批量历史数据存储在 HDFS 中，该应用通过基于 Spark 的预报模型训练任务，从 HDFS 中取出了历史数据并用随机森林模型进行了训练，并将模型保存在 HDFS 上。该训练任务由 YARN 集群提供计算资源进行执行。可见，本平台能够支持该应用进行批量数据分析。

该应用在有了预报模型的基础之上，还需要通过结合实时数据产生实时预报。该应用通过基于 Spark Streaming 的实时预报生成任务，从 Kakfa 获得实时数据，

从 HDFS 读取模型数据、天气预报数据和历史数据，并利用随机森林模型进行预测，最后将结果保存在暂存服务器 Redis 上。该预报生成任务运行在 YARN 集群之上。可以看出，本平台能够支持该应用进行流式数据分析。

该应用最后还需要将预测结果进行展现。该应用通过一个 Web 应用访问本平台的内置数据展现 API，从而取得之前存放于 Redis 上的预报数据。最后，该应用通过引用该 API 服务器提供的第三方图形化库对数据进行可视化，并画出图表。由此可见，本平台能够支持该应用进行数据展现。

由这些分析可见，如图 5-2 所示，本平台能够满足该应用进行批量分析，流式分析以及展现分析结果的需求。总结来说，本平台能够较好地向该应用提供分析环境数据的功能，能够支持智慧环保应用进行数据分析，达到了支持环境数据分析的设计目标。

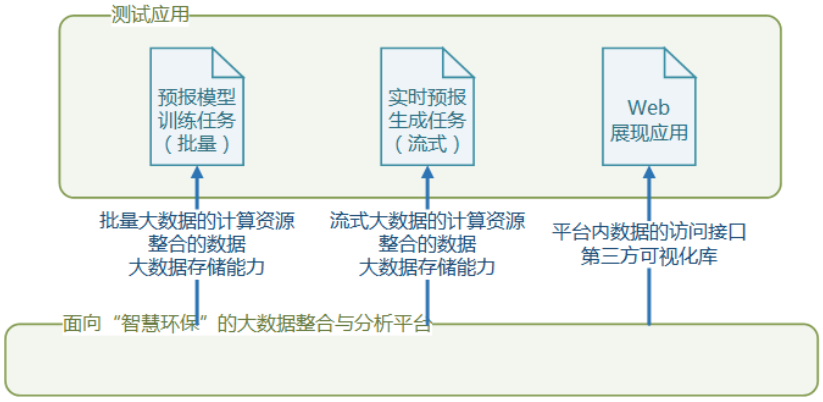

图 5-2 应用案例中数据分析的案例分析

Fig.5-2 Case study of data analysis in the application case

5.2 平台可用性、可伸缩性和可运维性的分析

5.2.1 平台可用性的分析

作为支持智慧环保的关键基础系统，本平台的可用性对于智慧环保的实施至关重要。本平台在设计过程中就考虑到了可用性，尽量避免了因为单点故障导致整个系统不可用的设计。本文首先对本平台的关键组成系统和服务器软件的可用性进行分析，然后对整个平台的可用性进行分析。

分系统和软件的可用性分析如表 5-1 所示，HDFS、YARN、Kafka、ZooKeeper、NiFi 和数据网关不会受到单点故障的影响，而 Ambari、SaltStack、MongoDB、Redis 和数据展现 API 服务器会受到单点故障的影响。

表 5-1 本平台关键组成部分的可用性分析
Table 5-1 Availability analysis of key components in this platform

平台组成部分	是否存在单点故障	保持可用的最低条件
HDFS 集群	否	NameNode 存活一机，QJM 服务器多数存活，ZooKeeper 服务器多数存活，块成功复制比例大于 99%（不失去过多的 DataNode 服务器）
YARN 集群	否	ResourceManager 存活一机，NodeManager 有足够多存活（满足计算资源）
Kafka 集群	否	ZooKeeper 服务器多数存活，Broker 服务器最多失效一机（与最小的主题复制因数相关）
ZooKeeper 集群	否	ZooKeeper 服务器多数存活
NiFi 集群	否	ZooKeeper 服务器多数存活，NiFi 节点至少一机存活
Ambari	是	Ambari 服务器存活
SaltStack	是	SaltStack 主节点存活
数据网关	否	每种数据网关至少一机存活
MongoDB	是	MongoDB 服务器存活
Redis	是	Redis 服务器存活
数据展现 API 服务器	是	数据展现 API 服务器存活

进而本文分析了本平台功能的可用性，如表 5-2 所示，可见主要的数据整合和分析功能的可用性较好，不存在单点故障的问题。数据展现功能和运维功能存在单点故障的问题，但是相对来说，运维功能不涉及业务数据，数据展现功能涉及的业务数据仅为暂存的数据，这两个功能的失效往往是临时的，对系统的影响相对较小。总体来说，本平台的主要功能具有较好的可用性。

表 5-2 本平台主要功能的可用性
Table 5-2 Availability analysis of key functions of this platform

关键功能	涉及组成部分	是否存在单点故障
实时数据整合功能	HDFS 集群、YARN 集群、Kafka 集群、ZooKeeper 集群	否
非实时数据整合功能	HDFS 集群、NiFi 集群、ZooKeeper 集群	否
流式数据分析功能	HDFS 集群、YARN 集群、Kafka 集群、ZooKeeper 集群	否
批量数据分析功能	HDFS 集群、YARN 集群、Kafka 集群、ZooKeeper 集群	否
数据展现功能	MongoDB、Redis、数据展现 API 服务器	是
运维功能	Ambari 服务器，SaltStack 主节点	是

5.2.2 平台可伸缩性的分析

环境监测数据的规模增长非常迅速，智慧环保相关的应用也在快速的发展。保持一定的可伸缩性，是本平台的一个重要设计目标。本文首先分析了本平台可能存在的伸缩方向，然后对本平台在这些方向上的可伸缩性进行了分析。

本平台的伸缩需求主要以扩容为主，受限于单台机器的性能，同时考虑到高性能的单机成本非常高，仅有纵向扩容能力无法完全解决平台长期扩容的问题，平台还需要有较好的横向扩容能力，即通过增加新的机器进行扩容的能力。本平台的扩容需求主要包括提高环境设备接入量、提高数据存储量、增加流式分析任务数量和计算量、增加批量分析任务数量和计算量、增加非实时数据整合量和增加数据展现应用数量等。为了满足这些扩容需求，本平台需要对平台内相应的系统进行扩容，其对应关系如表 5-3 所示。

表 5-3 本平台主要扩容方向
Table 5-3 Main directions of scaling of this platform

扩容需求	需要扩容的组成部分
提高环境设备接入量	数据网关，Kafka 集群
提高数据存储量	HDFS 集群
增加流式分析任务数量和计算量	Kafka 集群，YARN 集群
增加批量分析任务数量和计算量	YARN 集群
增加非实时数据整合量	NiFi 集群
增加数据展现应用	数据展现 API 服务器，暂存数据库 MongoDB 和 Redis

可见，本平台的扩容主要通过扩容数据网关、Kafka 集群、HDFS 集群、Kafka 集群、YARN 集群、NiFi 集群、数据展现 API 服务器、MongoDB 和 Redis 来实现。这些组成部分的横向扩容方式如表 5-4 所示，可见，除了增加数据展现应用的需求无法直接由横向扩容满足以外，其他扩容需求都能较好地以横向扩容的方式实现。数据展现应用通常对性能的要求较低，纵向扩容能在一定程度上满足需求，而较大规模的展现应用可以通过自行缓存部分数据来实现扩容。总体来说，就本平台主要的伸缩方向和需求而言，本平台具有较好的伸缩性。

表 5-4 本平台主要组成部分的横向扩容方式分析
Table 5-4 Analysis of ways to scale-up key components in this platform

平台组成部分	横向扩容方式
HDFS 集群	增加 DataNode 节点可以扩充存储容量
YARN 集群	增加 NodeManager 节点可以扩充可供调度的 CPU 和内存等计算资源

表 5-4（续）

平台组成部分	横向扩容方式
Kafka 集群	增加 Broker 节点可以进行并行的消息收发
NiFi 集群	增加 NiFi 节点可以进行并行的数据整合处理
数据网关	增加服务器数量可以提升接入数据的处理能力
数据展现 API 服务器	无法直接进行横向扩容
MongoDB	无法直接进行横向扩容
Redis	无法直接进行横向扩容

5.2.3 平台可运维性的分析

大数据的整合与分析平台往往需要长期运行，而平台的长期运行离不开运维工作，因而可运维性成为本平台的一项关键指标。本文分析了本平台的常见运维需求及其实现方式，从而揭示了本平台的可运维性。

平台的运维需求包括服务器软件配置更新、操作系统配置更新、服务器软件更新、操作系统软件包更新、部署新机器、文件备份、批量远程命令执行、计划任务等。本文选取其中具有代表性的一些运维操作进行分析，如表 5-5 所示，可见常见运维操作均可由 Ambari 和 SaltStack 进行有效的执行。总体来说，本平台具有良好的可运维性。

表 5-5 本平台主要运维操作实现方式
Table 5-5 Ways of performing major operational actions in this platform

运维操作	实现方式
服务器软件配置更新	Ambari 可以对 HDFS、YARN、ZooKeeper、Kafka 的配置进行版本化管理，其他服务器的配置可以由 SaltStack 的 file 模块配合 salt 文件系统进行管理
服务器软件更新	Ambari 可以对 HDFS、YARN、ZooKeeper、Kafka 的版本进行滚动更新，其他服务器的停机更新可以由 SaltStack
操作系统配置更新	可以由 SaltStack 的 file 模块配合 salt 文件系统进行管理
部署新机器	可以由 SaltStack 的 high state 系统进行管理
文件备份	可以由 SaltStack 的 file 模块、rsync 模块等执行
批量远程执行命令	可以由 SaltStack 筛选主机并使用 cmd 模块执行
计划任务	可以由 SaltStack 的 schedule 模块执行

5.3 平台性能测试与评估

5.3.1 平台性能测试概述

本文对本平台中的关键组成部分进行了性能测试，测试了 Kafka 和 Spark 的性能。其中，Kafka 的性能与本平台数据整合的性能有着直接的关系，Spark 的性

能则基本反映了本平台数据分析的性能。因此，进行这两个方面的性能测试，能够在很大程度上揭示本平台的整体性能。

本平台的运行环境部署在虚拟机中，且整个平台的虚拟机共享一台物理机，因此本文中的性能测试不能完全反映本平台在真实环境中的性能。但是考虑到真实环境中的物理机通常性能要优于本文运行环境中的虚拟机，所以该性能测试的结果仍然具有一定的参考意义。

5.3.2 Kafka 消息队列收发性能测试

Kafka 消息队列在数据整合和流式处理中扮演了重要角色，而相比延迟来说，消息的吞吐量对本平台的意义更重要，因而本文对其消息收发的吞吐量进行了测试。测试客户端在另一台物理测试机上，其硬件配置如表 5-6 所示，该测试机与本平台运行环境的宿主物理机通过千兆网络相连，测试工具采用 Kafka 自带的生产者性能测试脚本和消费者性能测试脚本。

表 5-6 Kafka 收发性能测试机硬件配置

Table 5-6 Hardware configuration of the test machine in Kafka send/receive benchmark

硬件	配置
处理器	Intel E3 1230 v3 @ 3.3Ghz（四核）
内存	16G DDR3
网络	Intel I210 千兆网络

本平台运行环境中的 Kafka 采用双机部署，故我们采用每个主题 2 个分区，复制比为 2，消费者 2 个接收线程的参数进行测试。考虑到贴近常用场景，生产者设置为等待响应但不等待复制完成，即异步复制。

环境数据采用 JSON 格式在 Kafka 上发送，消息的长度各有区别，通常在 100 字节到 2000 字节之间，故我们选择测试多种消息长度下的性能，分别测试 100 字节，500 字节，1000 字节和 2000 字节长度的消息，均测试 500MB 数据。首先为每个长度的消息设置对应的主题，然后先执行生产者测试脚本发送消息，再执行消费者测试脚本接收消息，分别测出独立收发消息的性能，最后同时启动生产者和消费者测试同时进行收发的性能。独立收发性能的测试结果如表 5-7。同时收发的性能测试结果如表 5-8。

经过整理，可以绘制出 Kafka 消息吞吐量（每秒消息数）与消息长度的关系图，如图 5-3 所示。可以看出，在环境数据的消息长度范围内，Kafka 表现出了较高的性能，能够满足数据整合和流式数据分析的性能需求。

表 5-7 独立收发 Kafka 主题的性能测试结果
Table 5-7 Kafka topic independent send/receive benchmark result

消息长度	发送（生产者）吞吐量		接收（消费者）吞吐量	
	消息数	带宽	消息数	带宽
100 字节	464554/s	44.30MB/s	773275/s	73.75MB/s
500 字节	117440/s	56.00MB/s	193611/s	92.32MB/s
1000 字节	55910/s	55.32MB/s	102304/s	97.57MB/s
2000 字节	28464/s	54.29MB/s	51610/s	98.44MB/s

表 5-8 同时收发 Kafka 主题的性能测试结果
Table 5-8 Kafka topic concurrent send/receive benchmark result

消息长度	发送（生产者）吞吐量		接收（消费者）吞吐量	
	消息数	带宽	消息数	带宽
100 字节	382907/s	36.52MB/s	662629/s	63.19MB/s
500 字节	89993/s	42.91MB/s	171199/s	81.63MB/s
1000 字节	48938/s	46.67MB/s	87872/s	83.80MB/s
2000 字节	20779/s	39.63MB/s	42636/s	81.32MB/s

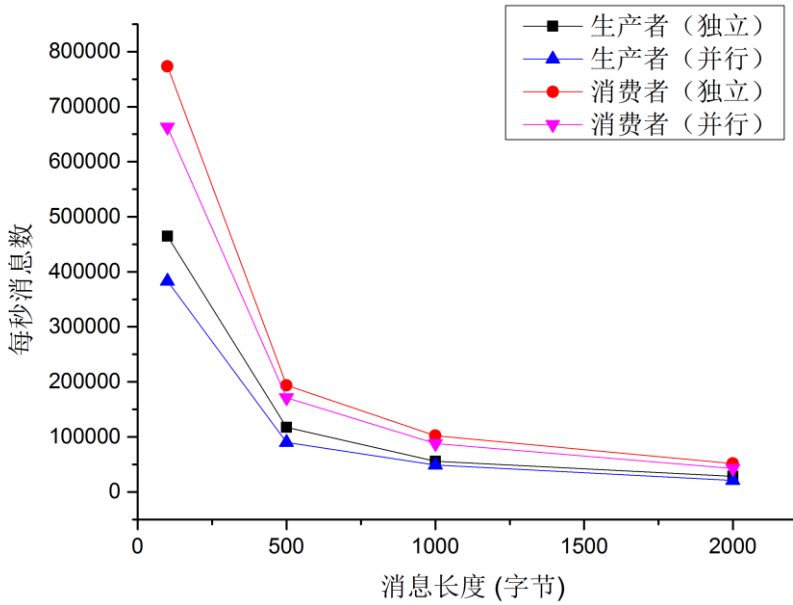

图 5-3 测试中 Kafka 吞吐量与消息长度的关系图
Fig.5-3 Relation between Kafka throughput and message size in tests

5.3.3 Spark 综合性能测试

Spark 为本平台提供了大数据处理能力，流式数据的分析和批量数据的分析都依赖于 Spark。本文利用 SparkBench[47]测试集的默认测试数据测试了常见 Spark

应用的执行时间，以评估本平台的性能。本文选择的用于测试的 SparkBench 负载、测试数据的主要生成参数和总计算时间见表 5-9。

表 5-9 用于性能测试的 SparkBench 负载及参数
Table 5-9 SparkBench workload and parameter in performance test

SparkBench 负载	测试数据主要生成参数	总计算时间
LinearRegression	nExmaples=40000, nFeatures=4	16s
LogisticRegression	nExamples=20000, nFeatures=20	20s
KMeans	nPoints=1000, nClusters=10	26s
LabelPropagation	nV=200	47s
MatrixFactorization	m=50, n=20, rank=10, maxIteration=3	20s
PregelOperation	nV=500	27s
ConnectedComponent	nV=50000	38s
ShortestPath	nV=1000	29s
PCA	nSamples=1000, nFeatures=1000	27s

经过整理，其总计算时间的柱状图如图 5-4 所示，可见，本平台表现出了合理的 Spark 性能，能够满足数据分析的性能需求。

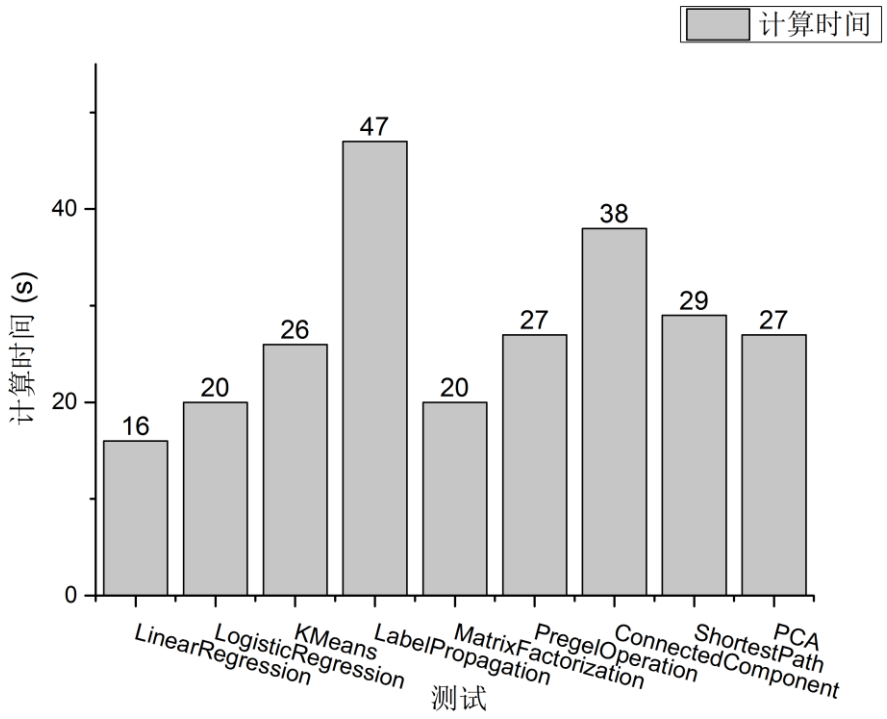

图 5-4 常见 SparkBench 测试集的性能测试结果
Fig.5-4 Benchmark result of common SparkBench test suite

5.4 本章小结

本章主要介绍了本平台的分析与评估。首先通过应用案例的案例分析对本平台的功能进行了分析和评估，评估显示本平台达到了支持数据整合与分析功能的设计目标。然后对本平台的可用性、可伸缩性和可运维性进行了分析，分析得出本平台具有较好的可用性、可伸缩性和可运维性。最后在配置的运行环境中对本平台的关键组成部分进行了性能测试，测试显示本平台表现出了较好的性能，能够满足大数据整合和分析的性能要求。

第六章 总结与展望

6.1 全文总结

随着人类的迅速发展，环境问题越发凸显，因而环境保护正在得到越来越多的重视。另一方面，信息技术的迅速进步，又使得智慧环保成为可能。智慧环保通过充分运用信息技术来感知和管理城市的环境，以求大幅提升一个城市的环境治理水平，是解决环境问题的有效手段。

然而，智慧环保的实现存在诸多技术挑战。智慧环保实施的关键之一在于分析城市中的环境数据，而环境数据存在着海量、规模增长迅速、异构程度高、来源广泛、形式多样等特点，其整合和分析都具有很大的技术难度，现有的一些解决方案并不能很好地解决智慧环保面临的这些问题。

本文以智慧环保为背景，以支持相关应用进行环境数据整合，支持相关应用进行大数据分析，和保证长期的实用性为目标，设计了一个面向智慧环保的大数据整合与分析平台。为了研究本平台对智慧环保相关应用的适用性，本文还对本平台进行了实现、分析和评估。

本文首先对本平台进行了总体设计，本平台包含数据整合平台、数据分析平台和运维支持系统三个主要部分，分别负责环境数据的整合，大数据的分析和运维支持功能。数据整合平台需要整合的数据主要包括环境设备采集的实时数据和其他环境相关数据两种。针对环境设备采集的实时数据，本文设计了一种基于 JSON 的环境数据中间格式，同时设计了数据网关层的架构用于将不同格式的设备数据转换为这种中间格式。这些数据被转换为中间格式后发送到 Kafka 消息队列中，再由一个基于 Spark Streaming 的流式任务取出进行清洗和分类，最后发送回 Kafka 消息队列并保存在 HDFS 上，从而完成整合。针对其他环境相关数据，本平台采用 NiFi 对其进行管理和整合。数据分析平台主要负责分析批量数据和流式数据，并提供展现支持。本平台的设计中采用 HDFS 集群提供大数据的存储能力，采用 YARN 集群提供大数据的计算能力，基于 Spark 和 Spark Streaming 的应用运行在 YARN 集群上，起到分析批量数据和流式数据的作用。为了支持数据展现，本文设计了暂存数据库的架构并设计了数据展现 API 层，用于将平台内的数据暴露给外部应用。运维支持系统主要负责执行运维操作，本平台的运维支持系统主要由 Ambari 和 SaltStack 组成。为了评估本平台的功能，本文还对智慧环保应用

案例“区域空气 PM2.5 污染 24 小时预报”在本平台上的实现方式做出了设计。

为了对本平台的可行性和适用性进行评估，本文配置了本平台的运行环境，对本平台进行了实现，同时也模拟了实际应用场景，对应用案例进行了实现。本文在虚拟机上配置了本平台的运行环境，搭建了 Ambari 和 SaltStack 作为运维支持系统，搭建了 HDFS、YARN、ZooKeeper、Kafka、NiFi、Redis、MongoDB、Node.js 等服务器软件和环境用于支持数据整合平台和数据分析平台。本文实现了本平台的关键系统，包括分别支持 HJ/T212 协议和 HTTP JSON 协议的数据网关，基于 Spark Streaming 的清洗与分类任务，和内置的数据展现 API 服务器。本文还模拟了实际应用场景并实现了应用案例，其主要由 NiFi 的数据整合配置、基于 Spark 的预报模型生成任务、基于 Spark Streaming 的实时预报生成任务和可视化 Web 应用组成。

最后，本文对本平台进行了分析与评估。本文通过对应用案例进行案例分析来评估本平台的能力，表明本平台能够支持智慧环保相关应用进行环境数据整合和大数据分析。本文还分析了本平台的可用性、可伸缩性和可运维性，并对本平台的性能进行了测试，结果显示本平台具有较好的可用性、可伸缩性和可运维性，性能能够满足需求，具有长期的实用性。

6.2 研究展望

随着社会和经济的发展，环境保护正在得到越来越多的关注，智慧环保作为实现环境保护的有力手段之一，蕴含着巨大的商业潜力和社会效益，具有较高的研究价值，本文的平台也有一些值得进一步研究的方向。

第一，平台可以进一步增加对智慧环保应用的功能支持，目前的平台主要以提供基础设施和基本数据为主，应用还需要自行开发大多数功能。如果能够对应用在本平台上的一些常用操作和分析算法进行整理、包装，形成智慧环保应用的开发库，将有利于进一步降低开发成本，提高智慧环保的实施水平。

第二，本平台的设计没有考虑到开放性，如果平台需要向第三方开发者开放，还需要增加安全性、公平性和可管理性等方面的设计。同时，为了建立起开放平台上的智慧环保应用生态，平台还需要提供更多的应用系统，例如数据市场等。

第三，基于现实资源和数据限制，本文主要在模拟应用场景下对本平台进行了分析和评估，而实际生产环境下的表现可能会有区别。未来在生产环境中对本平台进行分析和评估，将有助于进一步揭示本平台的能力，并改善设计。

参 考 文 献

[1] Martine G, Population Fund. Unleashing the potential of urban growth[M]. UNFPA, 2007.

[2] Neirotti P, De Marco A, Cagliano A C, et al. Current trends in Smart City initiatives: Some stylised facts[J]. Cities, 2014, 38: 25-36.

[3] O'Grady M, O'Hare G. How smart is your city?[J]. Science, 2012, 335(6076): 1581-1582.

[4] Chourabi H, Nam T, Walker S, et al. Understanding smart cities: An integrative framework[C]//System Science (HICSS), 2012 45th Hawaii International Conference on. IEEE, 2012: 2289-2297.

[5] Albino V, Berardi U, Dangelico R M. Smart cities: Definitions, dimensions, performance, and initiatives[J]. Journal of Urban Technology, 2015, 22(1): 3-21.

[6] Mitton N, Papavassiliou S, Puliafito A, et al. Combining Cloud and sensors in a smart city environment[J]. EURASIP journal on Wireless Communications and Networking, 2012, 2012(1): 1.

[7] Zanella A, Bui N, Castellani A, et al. Internet of things for smart cities[J]. IEEE Internet of Things Journal, 2014, 1(1): 22-32.

[8] Kitchin R. The real-time city? Big data and smart urbanism[J]. GeoJournal, 2014, 79(1): 1-14.

[9] Vilajosana I, Llosa J, Martinez B, et al. Bootstrapping smart cities through a self-sustainable model based on big data flows[J]. IEEE Communications Magazine, 2013, 51(6): 128-134.

[10] Hashem I A T, Chang V, Anuar N B, et al. The role of big data in smart city[J]. International Journal of Information Management, 2016, 36(5): 748-758.

[11] Sanchez L, Galache J A, Gutierrez V, et al. Smartsantander: The meeting point between future internet research and experimentation and the smart cities[C]//Future Network & Mobile Summit (FutureNetw), 2011. IEEE, 2011: 1-8.

[12] Cheng B, Longo S, Cirillo F, et al. Building a Big Data Platform for Smart Cities: Experience and Lessons from Santander[C]//2015 IEEE International Congress on Big Data. IEEE, 2015: 592-599.

[13] Nam T, Pardo T A. Smart city as urban innovation: Focusing on management, policy, and context[C]//Proceedings of the 5th International Conference on Theory

and Practice of Electronic Governance. ACM, 2011: 185-194.

[14] Zhang Y, Li N, Zhang Y. The Research of China’s Urban Smart Environmental Protection Management Mode[M]//Geo-Informatics in Resource Management and Sustainable Ecosystem. Springer Berlin Heidelberg, 2015: 415-423.

[15] Zheng Y, Liu F, Hsieh H P. U-Air: when urban air quality inference meets big data[C]//Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2013: 1436-1444.

[16] Feng X, Li Q, Zhu Y, et al. Artificial neural networks forecasting of PM 2.5 pollution using air mass trajectory based geographic model and wavelet transformation[J]. Atmospheric Environment, 2015, 107: 118-128.

[17] Baker P, Agius R. Air pollution exposure and adverse pregnancy outcomes in a large UK birth cohort: use of a novel spatio-temporal modelling technique[J]. Scandinavian journal of work, environment & health, 2014, 40(5): 518.

[18] Wan B, Ma R, Zhou W, et al. Smart City Development in China: One City One Policy[J]. ZTECOMMUNICATIONS, 40.

[19] 吴勇, 张红剑. 基于大数据和云计算的智慧环保解决方案[J]. 信息技术与标准化, 2013, 11: 025.

[20] Wu Q, Fang W, Wu X J. Smart Environment Protection in Wuxi[R]. Working group of IEEE smart city of Wuxi, 2016.

[21] Sharma S. Big data landscape[J]. International Journal of Scientific and Research Publications, 2013, 3(6): 1.

[22] Demchenko Y, De Laat C, Membrey P. Defining architecture components of the Big Data Ecosystem[C]//Collaboration Technologies and Systems (CTS), 2014 International Conference on. IEEE, 2014: 104-112.

[23] Ranjan R. Streaming big data processing in datacenter clouds[J]. IEEE Cloud Computing, 2014, 1(1): 78-83.

[24] Shaw S, Vermeulen A F, Gupta A, et al. Loading Data into Hive[M]//Practical Hive. Apress, 2016: 99-114.

[25] Kreps J, Narkhede N, Rao J. Kafka: A distributed messaging system for log processing[C]//Proceedings of the NetDB. 2011: 1-7.

[26] Singh D, Reddy C K. A survey on platforms for big data analytics[J]. Journal of Big Data, 2014, 2(1): 1.

[27] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107-113.

[28] Zaharia M, Chowdhury M, Franklin M J, et al. Spark: cluster computing with working sets[J]. HotCloud, 2010, 10: 10-10.

[29] Vavilapalli V K, Murthy A C, Douglas C, et al. Apache hadoop yarn: Yet another resource negotiator[C]//Proceedings of the 4th annual Symposium on Cloud Computing. ACM, 2013: 5.

[30] Zaharia M, Das T, Li H, et al. Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters[C]//Presented as part of the. 2012.

[31] Sanchez L, Muñoz L, Galache J A, et al. SmartSantander: IoT experimentation over a smart city testbed[J]. Computer Networks, 2014, 61: 217-238.

[32] Hunt P, Konar M, Junqueira F P, et al. ZooKeeper: Wait-free Coordination for Internet-scale Systems[C]//USENIX Annual Technical Conference. 2010, 8: 9.

[33] HJ/T 212-2005, 污染源在线自动监控（监测）系统数据传输标准[S].

[34] GB 3095-2012, 环境空气质量标准[S].

[35] GB 3838-2002, 地表水环境质量标准[S].

[36] GB 15618-1005, 土壤环境质量标准[S].

[37] GB 3096-2008, 声环境质量标准[S].

[38] GB 6763-86, 建筑材料用工业废渣放射性物质限制标准[S].

[39] ISO 8601:2004, Data elements and interchange formats -- Information interchange -- Representation of dates and times[S].

[40] Jeong H J, Piao X F, Choi J H, et al. Efficient Integration Method of Large-Scale Heterogeneous Security Logs Using NoSQL in Cloud Computing Environment[J]. 網際網路技術學刊, 2016, 17(2): 267-275.

[41] Han J, Haihong E, Le G, et al. Survey on NoSQL database[C]//Pervasive computing and applications (ICPCA), 2011 6th international conference on. IEEE, 2011: 363-366.

[42] Rai R. Socket. IO Real-time Web Application Development[M]. Packt Publishing Ltd, 2013.

[43] Hosmer B. Getting started with salt stack--the other configuration management system built with python[J]. Linux journal, 2012, 2012(223): 3.

[44] Svetnik V, Liaw A, Tong C, et al. Random forest: a classification and regression tool for compound classification and QSAR modeling[J]. Journal of chemical information and computer sciences, 2003, 43(6): 1947-1958.

[45] Kivity A, Kamay Y, Laor D, et al. kvm: the Linux virtual machine monitor[C]//Proceedings of the Linux symposium. 2007, 1: 225-230.

[46] Maeda K. Performance evaluation of object serialization libraries in XML, JSON and binary formats[C]//Digital Information and Communication Technology and it's Applications (DICTAP), 2012 Second International Conference on. IEEE, 2012:

177-182.

[47] Li M, Tan J, Wang Y, et al. Sparkbench: a comprehensive benchmarking suite for in memory data analytic platform spark[C]//Proceedings of the 12th ACM International Conference on Computing Frontiers. ACM, 2015: 53.

附录 1 常用监测项目标准化数据

常用监测项目标准化数据见附表 1。

附表 1 常用监测项目标准化数据

监测项目标识符	名称	监测类型	标准单位标识符	数据类型	数据范围
pm25	PM2.5	空气	mcg/m3	数值	0-999
pm10	PM10	空气	mcg /m3	数值	0-999
so2	SO_2	空气	mcg /m3	数值	0-9999
no2	NO_2	空气	mcg /m3	数值	0-9999
co	CO	空气	mg/m3	数值	0-999
o3	O^3	空气	mcg /m3	数值	0-9999
tsp	TSP	空气	mcg /m3	数值	0-999
nox	NO_x	空气	mcg /m3	数值	0-999
pb-a	Pb	空气	mcg /m3	数值	0-99
bap	BaP	空气	mcg /m3	数值	0-1
temp-w	水温	水	deg-c	数值变化	0-99
ph-w	pH	水	1	数值	0-14
o2	溶解氧	水	mg/L	数值	0-9
cod	COD	水	mg/L	数值	0-99
bod5	BOD_5	水	mg/L	数值	0-99
nh3n	NH_3-N	水	mg/L	数值	0-9
p	总磷	水	mg/L	数值	0-9
tn	总氮	水	mg/L	数值	0-99
cu-w	Cu	水	mg/L	数值	0-99
zn-w	Zn	水	mg/L	数值	0-99
f-	F^-	水	mg/L	数值	0-99
se	Se	水	mg/L	数值	0-9

附表 1（续）

监测项目标识符	名称	监测类型	标准单位标识符	数据类型	数据范围
as-w	As	水	mg/L	数值	0-99
cd-w	Cd	水	mg/L	数值	0-9
cr6+	Cr^{6+}	水	mg/L	数值	0-99
pb-w	Pb	水	mg/L	数值	0-99
cyanide	cyanide	水	mg/L	数值	0-99
phenol	挥发酚	水	mg/L	数值	0-99
sulfide	硫化物	水	mg/L	数值	0-99
fcoli	粪大肠菌群	水	cfu/L	数值	0-9999999
so42-	SO_4^{2-}	水	mg/L	数值	0-9999
cl-	Cl^-	水	mg/L	数值	0-9999
n	硝酸盐	水	mg/L	数值	0-9999
fe	Fe	水	mg/L	数值	0-99
mn	Mn	水	mg/L	数值	0-99
cd-s	Cd	土壤	mg/kg	数值	0-999
hg	Hg	土壤	mg/kg	数值	0-999
as-s	As	土壤	mg/kg	数值	0-999
cu-s	Cu	土壤	mg/kg	数值	0-9999
pb-s	Pb	土壤	mg/kg	数值	0-9999
cr	Cr	土壤	mg/kg	数值	0-9999
zn-s	Zn	土壤	mg/kg	数值	0-9999
ni	Ni	土壤	mg/kg	数值	0-9999
noise	噪音	声环境	dB	数值	0-999
rate	频率	电磁辐射	MHz	数值	0-99999
elecint	电场强度	电磁辐射	V/m	数值	0-999
magint	磁场强度	电磁辐射	A/m	数值	0-9
pwrdens	功率密度	电磁辐射	W/m2	数值	0-999
gamma	γ 照射量率	核辐射	mcR/h	数值	0-9

致　谢

还记得怀着激情和梦想开始研究生学业的那个时候，我是多么希望能够早日走出这里，用我的所想所学，成就一番事业。转眼间论文已经完成，我才终于知道对这片土地有多么不舍，而最为不舍的正是在这里的人们，是你们帮助我、关心我、鼓励我、支持我，为了带来了真正的成长，我要向你们表示由衷的感谢。

首先我要衷心地感谢我的导师吴刚老师。本文是在吴老师的悉心指导下完成的，从论文选题工作开始，他就定期抽出时间与我讨论，解决我在课题推进中遇到的问题，吴老师在学术领域、工程领域深刻的洞见开阔了我的思路，拓宽了我的视野。在论文的实际进展阶段，吴老师也耐心地给出了很多建设性的指导意见，指出了论文研究问题的核心，解答了我的很多困惑。在研究生学业的两年半时间里，吴老师专业严谨的态度和专注务实的作风都深刻地影响了我，不但教会了我如何在专业领域不断求索，更教会了我做人的道理，成为我人生中一笔宝贵的财富。吴老师在生活上也给予我们关怀，经常组织实验室的活动，给我们留下了很多美好的回忆。

感谢在 ADC 实验室的每位同学。感谢各位师兄在项目、生活和工作上对我的帮助，感谢大家的支持和鼓励。感谢大家在这两年半时间里的陪伴，和你们一起努力、进步、玩耍的日子成为了我的青春中不可磨灭的一部分，使我受益匪浅，我会好好珍惜。

我还要感谢我的家人，是你们一直默默地在身后支持着我。正因为有你们的支持，我才能全身心地投入到学习、研究中去。正因为有你们作为我坚强的后盾，我才能更加勇敢地面对挑战和坎坷。

最后，我还要感谢上海交通大学，感谢电子信息与电气工程学院，感谢软件学院，感谢你们提供的优秀环境和对我的培养。感谢参与论文评审的各位专家和老师、感谢所有为我传道授业解惑的课程老师，感谢所有为我的学业顺利进行提供帮助的领导和工作人员。即将走出校园的我，不会忘记大家的陪伴，会努力以丰厚的成果，来答谢母校、老师、同学和家人！

攻读硕士学位期间已发表或录用的论文

[1] Rundong F, Gang W. SCAPE: An Application Platform for Environmental Big Data Analysis in Smart Cities[C]//2016 International Conference on Modeling, Simulation and Optimization Technologies and Applications. Atlantis Press, 2016.（已录用）