计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(12):79−85 [doi: 10.15888/j.cnki.csa.007181]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

Hadoop 环境下分布式物联网设备状态分析处理系统①

张瑞聪, 任鹏程, 房　凯, 张卫山

(中国石油大学 (华东) 计算机与通信工程学院, 青岛 266580)
通讯作者: 张瑞聪, E-mail: 1240686068@qq.com

摘　要: 设备故障可能会引起严重的生产事故, 对企业、社会和人身安全造成严重威胁. 因此, 对物联网设备状态分析并进行合理的处理具有重要意义. 针对物联网设备数据量大且复杂的特性, 本文提出了一种针对物联网设备的海量数据处理架构, 同时结合 Dask 分布式计算框架, 设计了基于 Hadoop 环境的分布式物联网设备状态分析处理系统. 本系统主要包括数据服务、数据分析和数据存储 3 个模块, 并通过合理的节点调度方案保证了算法的高效运行和分布式计算的稳定性. 系统运行表明能有效的处理大批量数据并实时准确预测设备状态, 满足工业智能制造过程中的实际应用.

关键词: 物联网设备状态; Hadoop; Dask 分布式计算框架; 节点调度

引用格式: 张瑞聪,任鹏程,房凯,张卫山.Hadoop 环境下分布式物联网设备状态分析处理系统.计算机系统应用,2019,28(12):79–85. http://www.c-s-a.org.cn/1003-3254/7181.html

Distributed Status Analysis and Processing System for IoT Device in Hadoop Environment

ZHANG Rui-Cong, REN Peng-Cheng, FANG Kai, ZHANG Wei-Shan

(College of Computer & Communication Engineering, China University of Pertroleum, Qingdao 266580, China)

Abstract: Equipment failures can cause serious production accidents and pose a serious threat to business, society, and personal safety. Therefore, it is important to analyze the state of the IoT device and reasonablely process. Aiming at the large and complex data of IoT devices, this study proposes a massive data processing architecture for IoT devices. At the same time, combined with Dask distributed computing framework, a distributed device state analysis and processing system for IoT based on Hadoop environment is designed. The system mainly includes three modules of data service, data analysis, and data storage, and through reasonable node scheduling scheme, the efficient operation of the algorithm and the stability of distributed computing are guaranteed. The system operation shows that it can effectively process large quantities of data and accurately predict the status of the equipment in real time to meet the practical application in the industrial intelligent manufacturing process.

Key words: status of IoT devices; Hadoop; Dask distributed computing framework; node scheduling

随着工业化、信息化的飞速发展, 设备工艺越来越复杂, 监控、传感器的广泛部署使得设备的运行状态得以有效监控[1]. 基于工业行业特性, 工业物联网大数据显现出规模大、类型杂和价值密度不一致等特征[2], 因此, 相比传统的大数据, 对具有以上特征的大数据分析处理更加困难. 而且目前设备故障以发生故障后进

① 基金项目: 国家自然科学基金 (61309024); 山东省自然科学基金 (F020509, F060604)
Foundation item: National Natural Science Foundation of China (61309024); Natural Science Foundation of Shandong Province (F020509, F060604)
收稿时间: 2019-05-07; 修改时间: 2019-05-28; 采用时间: 2019-06-04; csa 在线出版时间: 2019-12-10

行报警的方式呈现, 个别故障预测只能基于专家经验[3]判断, 过度依赖于专家知识.

本文结合当前工业领域[4]发展面临的问题, 设计了基于 Hadoop 环境的分布式物联网设备状态分析处理系统, 主要提供数据服务、数据分析和数据存储的功能. 通过分布式系统的构建, 有力地解决了当前工业物联网设备数据规模大、设备数据类型杂、设备数据价值密度不一等问题, 实现了工业物联网设备运行状态的精准诊断分析, 有效提高生产效率、降低资源能源消耗.

1 相关工作

近年来, 国内外对物联网设备状态的分析和处理方面的研究已经获得了广泛的应用, 尤其是在设备运行状态监控和故障诊断方面. 2009 年, 曾凡琳等双层监控系统框架下的带有时延和丢包的子系统故障检测方法[5], 减小了延时与丢包对故障检测性能的影响[6]. 2019 年, 焦亚军等利用 PLC 和传感器设计了一套带式运送机故障检测和预防系统, 随时检测设备故障点[7], 若出现故障信号立刻发出报警, 提示维修. 遗憾的是以上系统均未考虑对海量数据处理时性能方面的问题.

Hadoop 是一个能够让用户轻松架构和使用的分布式计算平台. 用户可以轻松地在 Hadoop 上开发和运行处理海量数据的应用程序[8]. 它主要有高可靠性、高扩展性、高效性、高容错性的优点.

根据 Hadoop 平台的特点, 2013 年, 刘树仁等为了处理海量数据, 设计并实现了基于 Hadoop 技术的数据存储系统[9], 验证了该系统具有分布式海量存储及高效查询的优势, 适合智能电网环境下设备状态监测数据的存储; 2014 年, Duan 等通过智能化分析能提出新的云计算数据存储管理模式, 让管理的效率更高、安全性更好、维护性更强[10].

尽管 Hadoop 在离线复杂大数据处理方面表现良好, 但支持的机器学习算法却相当有限, 为了更好地利用 Python 数据处理包做大数据处理, 本文引入了 Dask 计算工具. Dask 是一个由 Python 语言支撑的灵活的分布式计算[11]工具, 主要具有以下两个特点: 第一为动态任务分配, 第二具有大数据处理能力. Dask 分布式计算提供并行化 Numpy array 和 Pandas DataFrame 类的接口, 同时提供任务分配接口, 更加便捷地将分布式计算整合到项目之中. Dask 采用原生态访问 pydata 堆栈, 在完全 Python 环境中启动分布式计算, 能够保证以最小的性能开销和较低的延迟情况下快速计算[12]. 此外 Dask 既能运行于千个节点上做分布式计算, 也能运行于单台笔记本上, 且在响应方面基于交互式设计理念, 为开发者和使用者提供快速的反馈和精准的诊断.

本文结合 Hadoop 和 Dask 分布式计算框架, 设计了一种基于 Hadoop 环境的分布式物联网状态分析和处理系统, 系统能够高效地处理大批量数据并实时准确预测设备的运行状态.

2 系统设计

2.1 系统概要设计

根据相关需求分析, 本系统采用数据层、服务层、通讯层和表现层四层架构, 具体形式如图 1 所示.

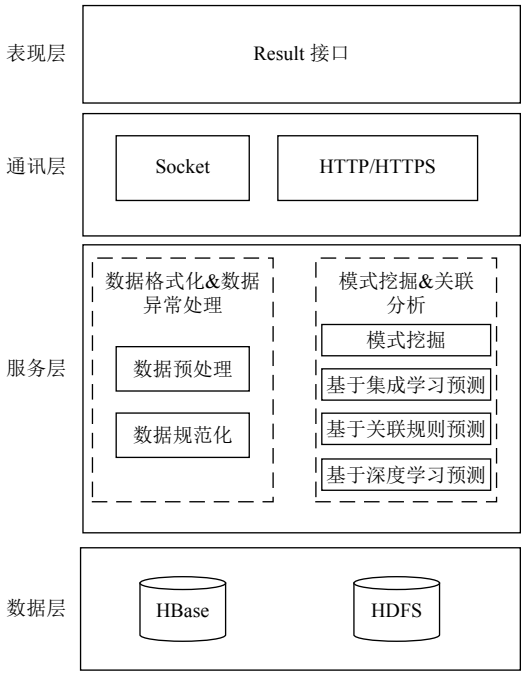


图 1 系统架构

结合四层架构的设计方式, 系统由 3 个模块组成: 数据服务模块、数据分析模块和数据存储模块.

数据服务模块采用 Web-Server, 避免一台主机多环境相互影响而导致的程序故障问题[13]. 以双向服务的方式实现数据的推拉结合, 在保证高速数据传输的同时实现数据异常处理、消息丢失重发、消息“断点续传”, 保证了双向数据传输的可靠性和稳定性. 该模块不仅从云平台拉取数据, 还可以将数据预测的结果反馈回云平台. 此外数据服务模块和数据处理层是完

成隔离, 两层中间采用 Socket 通信.

数据分析模块在服务层实现, 其中数据处理和特征提取选择基于 Dask 分布式计算框架, 预测基于 Keras 框架[14]. 数据处理的主要功能有: 数据缺失填充、数据异常检测和替换、数据归一化和数据降维处理等. 特征提取和选择模块主要功能为数据特征提取和选择. 数据预测主要功能为预测算法的训练、测试与运行. 此外该模块还支持模型更换 (支持数据归一化算法替换、数据降维算法替换和异常检测算法替换) 和数据持久化等功能.

数据存储模块负责模型保存、预测结果保存和提取特征保存等数据存储服务. 由于数据量会随着时间而不断增加为此我们必须考虑到数据量增加导致得硬盘不足问题, 结合服务实时访问需求最终采用 HBase 作为数据存储, 另外采用 HDFS[15]作为存储基础.

2.2 系统详细设计

本节将讲解数据服务模块、数据分析模块和数据存储模块的详细设计. 系统详细设计图如图 2 所示.

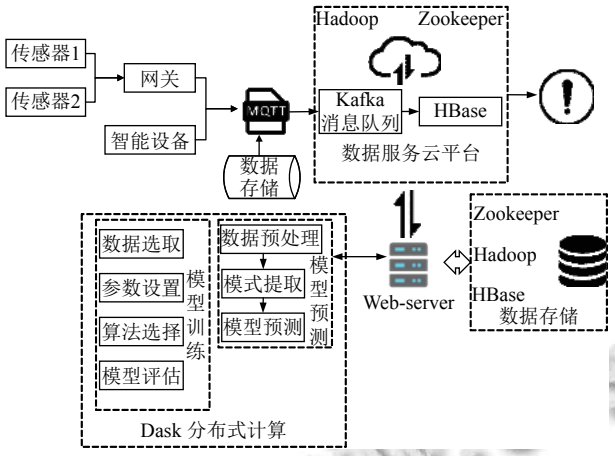


图 2 系统详细设计图

2.2.1 数据服务模块设计

数据服务模块定义了数据检查机制、数据重传协议和数据传输格式. 数据发送方会主动将每一台设备所有传感器数据实时发送给 Web-server 服务器[16], 数据格式如下:

```
{
"Hash_MD5": "39203040",
  "SensorData": {
    "设备名称": "1#机",
    "状态参数": {
      "数据时间": "2017-09-12 23:59:32",
      "冷冻泵运行指示": "1",
      ……
    },
    "设备数据": {
    "数据时间": "2017-09-12 23:59:32",
    "蒸发器侧进水温度": "12.1",
      ……
    }
  }
}
```

上述数据传送格式为 json, 数据中包含设备名称和数据 (分别是状态参数和设备数据) 以及校验码 Hash_MD5, 且 MD5 计算值是 SensorData 元素的字符串计算的 MD5 值. 当 SensorData 数据发送缺失或者数据“污染”时 MD5 计算结果将会发生变化, 因此可以确定数据稳定性和可靠性. 当数据计算 MD5 匹配/不匹配都会向数据发送方发送数据状态, 发送 Json 如下所示:

```
{
  "Megs": "0/1"
  "SensorData": {
  "设备名称": "1#机",
  "数据时间": "2017-09-12 23:59:32"
  }
}
```

如上所示, 当数据传输中数据缺少或被干扰时返回为 1, 如果数据接收且数据完整性和准确性良好则 Megs 会返回 0, 否则应表示数据已接收但数据校验出错, 请求数据重发. 同时当数据发送方在数据发送 2 秒后没有收到数据确认时, 数据发送方会立刻向数据接收端再次发送数据, 同时再次等待数据确认消息.

当网络传输出现问题时内存数据会进行暂时数据存储, 其中数据持久化为双端持久化, 以便网络恢复时能迅速重建程序, 同时确保内存中数据不会丢失, 从而避免网络故障带来的数据部分丢失问题[17]. 具体实现为数据发送端会实时保存数据传输时间点并持久化, 数据传输时间点是记录当前数据成功发送并且返回接受成功的数据时间点 (数据发送后且收到接收端数据 Megs 为 0 的确认); 数据接收端会实时持久化数据组, 具体流程为在接收端接收到数据时并不会立刻返回数据确认, 首先将数据添加到内存数据数组中 (添加之前

会进行数据是否存在判断, 排除数据接收但确认消息丢失情况) 并将其持久化到数据库中, 然后再返回数据确认消息[18].

将预测结果反馈回云平台时, 由于 Web_server 和预测模块不是运行于同一台服务器, 为降低由数据写入和读取带来的延迟, 引入模块之间通信. 本论文采用 Socket 通信方法[19], 主要实现两个方法 Socket_server 和 Socket_client. Socket_server 为 Socketserver 端负责将验证正确的数据实时发送, 而预测模块为 Socket_client 负责接收数据, 以及将预测结果返回给 Web_server. Socket 通信 JSON 格式设计如下所示, 传输数据只有设备名称和设备数据 (设备状态数据未使用). 预测数据返回的 JSON 格式传输数据只有设备名称、数据时间和预测状态.

```
{
  "设备名称": "1#机",
  "设备数据": {
  "数据时间": "2017-09-12 23:59:32",
  "蒸发器侧进水温度": "12.1",
  ……
  }
}
```

2.2.2 数据分析模块设计

数据分析模块主要基于 Dask 分布式计算框架, 该分布式计算框架能有效支持大量机器学习算法并通过高效的并行计算缩短计算时间. 同时引入了计算资源调度 DRB 方法, 保证数据预处理算法和预测算法的高效运行和分布式计算的稳定性. 下面主要介绍分布式计算框架的搭建和节点调度方案的实现.

本分布式计算平台基于 Dask 技术来搭建, 通过搭建一台主节点多台从节点和多个备用节点的组成最终的分布式计算平台, 每个节点之间采用 TCP 通信, 高速并行化计算[20]每次主节点分发的计算请求. 主节点名称、从节点名称和备用节点名称分别为 Scheduler、Worker 和 Temp_Worker.

本分布式计算将由 5 台机器组成, 其中每台机器包含 4 个节点. 因为 Dask 分布式计算能在运行时刻加入节点, 且当节点加入后可以快速的从 Dask 主节点获得计算任务, 平台的设备信息都可以从设备的主机点获取, 信息读取地址为: http://192.168.1.101:端口/workers, 端口号根据实际信息修改访问端口号.

节点调度方案如图 3 所示, 当设备计算资源过于匮乏时 Temp_Worker 将持续启动子线程并加入分布式计算系统中, 其中计算资源匮乏判断条件为整体资源使用率情况[21].

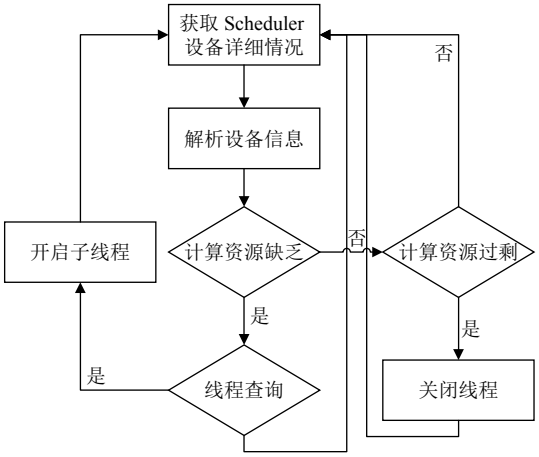


图 3 DRB 节点调度方案

计算资源匮乏具体的判断条件如式 (1) 和式 (2) 所示, C表示分布式计算平台整体 CPU 计算资源大小, c_i表示其中一台机器某个节点的 CPU 使用率, M 表示分布式计算平台整体内存大小, m_i表示其中一台子节点的内存使用率. m_t表示在 t 时刻分布式计算平台整体内存使用均值即如下公式所示:

$$c_t = \frac{\sum_{i}^{n} c_i}{C} \tag{1}$$

$$m_t = \frac{\sum_{i}^{n} m_i}{M} \tag{2}$$

式中, n 表示设备总的节点数量不包含主节点. c_t在时间点 t 时刻分布式计算平台的 CPU 整体使用率[22]. 当且仅当任何一个整体使用率 c 超过 0.8(计算机超频, 不同机器可能不一样) 或者 m 超过 0.8 时且该值维持 10 秒以上, 待加入节点将启动 Worker 并加入 Scheduler. 此外当设备资源过剩时我们将关闭部分启动节点, 具体资源判断当c_t和m_t同时在持续 5 秒内资源利用率都低于在 0.3 时备用节点进程将终止.

此外在多次实验过程中发现部分节点可能会出现死机或者内存使用率抵达使用上限, 导致节点瘫痪或节点进程阻塞问题. 为此根据分布式计算的任务备份的基本原理 (Dask 的 Scheduler 在管理节点时, 如果节

点丢失时, 主节点会将丢失节点任务分发给其它节点执行), 节点进程阻塞判断方案将根据m_i单个节点内存使用率和 c_i 单个节点 CPU 使用率. 判断条件为当且当c_i一直大于 1.2 时且维持利用率不变 20 s 以上时节点线程将会被终止, 并立刻再次启动该节点进程.

2.2.3 数据存储模块设计

数据存储主要包括设备数据存储、预测数据存储和中间模式存储. 为保证计算的高效性和稳定性, 存储服务器采用基于 Hadoop 分布式数据存储的 HBase 数据库[23], 在尽可能保证数据安全的同时提供便捷的存储扩展.

数据存储在 HBase 中有两种表结构设计模式, 宽表设计模式和高表设计模式[24]. 宽表设计模式和高表设计模式根据数据量大小的时间消耗如表 1 所示, 其中高表设计模式的时间消耗平均低于宽表设计模式百分之五十以上. 因此本文选择高表设计模式, 通过减少列簇, 将查询信息插入到 Rowkey 中来降低查询时间.

根据某著名物联网公司的数据分析, 最终数据建表通过构建三个数据列簇和一个 Rowkey. Rowkey 为数据识别的唯一标识符, 且数据查询时间相对较快, 因此 Rowkey 将采用设备名称和时间组成 Rowkey 设计, 在数据查询时只需设备名称+(开始时间–结束时间) 即可查询某一设备的多个时间段数据. 预测数据将存储于第三个数据列簇中, 预测数据列将采用 *yc* 作为标识. 中间模式数据存储构建方式和预测数据存储类似, 只需将列标识修改为 *ts* 存储于第四列簇中. 表列簇设计如表 2 所示.

表 1 宽表高表对比

行数	宽表设计模式 (s)	高表设计模式 (s)
五百万行数据	0.248	0.083
一千万行数据	0.438	0.122
两千万行数据	0.88	0.298

表 2 数据库表设计

Rowkey	Column-family1				Column-family2				Column-family3
ssdjd_20180105125000_1	S1	S2	…	S137	TS1	TS2	…	TS8503	P1

3 个列簇分别表示为原始数据存储、转换数据存储和预测设备状态存储. Rowkey 为索引只有一列 (ssdjd_20180105125000_1 中最后_1 是为了避免时间轴既 20180105125000 出错从而无法通过加 30 秒定位上一个时间或下一个时间点数据), 原始数据列簇包含 137 列, 转换数据包含 8503 列, 预测设备状态存储设备的当前状态, 当预测出设备状态时, 该预测值为半小时后的设备状态, 因此为维持预测 p1 与当前时间轴一致性, 引入内存变量和持久化方法, 在预测出数据时先暂且保存在该变量中, 每当数据超过 30 个时会进行自动数据存储.

HBase 数据库存储软硬件环境如表 3 所示, HBase 存储服务采用 5 台服务器作为数据存储服务器, 其中 152 作为数据存储的主节点 Master 同时也作为资源管理节点, 153~156 作为子节点 datanode 为数据存储节点.

表 3 HBase 数据库软硬件环境

IP	硬件配置	软件配置 1	软件配置 2
192.168.2.152	24 GB, 1 处理器, 6 核心	ResourceManager, Namenode, historyserver, Master, Hmaster	ResourceManager, ameNode, RegionServer, JobHistoryServer, aster, Hmaster
192.168.2.153	8 GB, 1 处理器, 2 核心	NodeManager, datanode, slave, HRegionServer, server.1	ResourceManager, ameNode, RegionServer, JobHistoryServer, aster, Hmaster
192.168.2.154	8 GB, 1 处理器, 2 核心	NodeManager, datanode, slave, HRegionServer, server.2	NodeManager, ataNode, RegionServer, obHistoryServer, orker, uorumPeerMain
192.168.2.155	8 GB, 1 处理器, 2 核心	NodeManager, datanode, slave, HRegionServer, server.3	NodeManager, ataNode, HRegionServer, obHistoryServer, Worker, QuorumPeerMain
192.168.2.156	24 GB, 1 处理器, 6 核心	NodeManager, datanode, slave, HRegionServer, erver.0, hive, sqoop, oozie	NodeManager, ataNode, RegionServer, RunJar, RunJar, JobHistoryServer, Worker, uorumPeerMain

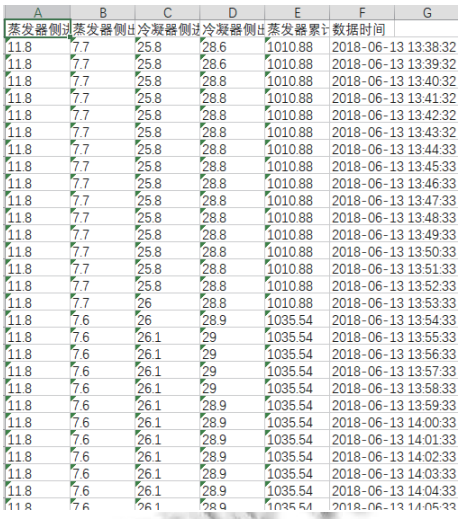
3 实验分析

3.1 系统预测功能验证

为了验证系统的有效性, 本文选择某企业商业空调的传感器数据进行实验, 部分数据展示如图 4 所示.

系统可以得到蒸发器侧进水温度、蒸发器侧出水温度、冷凝器侧进水温度、冷凝器侧出水温度等

44 个不同特征的商业空调传感器数据. 系统通过数据服务模块从企业云平台的得到维度为 45 的矩阵数据, 其中最后一列数据为设备的运行状态标识.



A	B	C	D	E	F	G
蒸发器侧进	蒸发器侧出	冷凝器侧进	冷凝器侧出	蒸发器累计	数据时间	
11.8	7.7	25.8	28.6	1010.88	2018-06-13 13:38:32	
11.8	7.7	25.8	28.6	1010.88	2018-06-13 13:39:32	
11.8	7.7	25.8	28.8	1010.88	2018-06-13 13:40:32	
11.8	7.7	25.8	28.8	1010.88	2018-06-13 13:41:32	
11.8	7.7	25.8	28.8	1010.88	2018-06-13 13:42:32	
11.8	7.7	25.8	28.8	1010.88	2018-06-13 13:43:32	
11.8	7.7	25.8	28.8	1010.88	2018-06-13 13:44:33	
11.8	7.7	25.8	28.8	1010.88	2018-06-13 13:45:33	
11.8	7.7	25.8	28.8	1010.88	2018-06-13 13:46:33	
11.8	7.7	25.8	28.8	1010.88	2018-06-13 13:47:33	
11.8	7.7	25.8	28.8	1010.88	2018-06-13 13:48:33	
11.8	7.7	25.8	28.8	1010.88	2018-06-13 13:49:33	
11.8	7.7	25.8	28.8	1010.88	2018-06-13 13:50:33	
11.8	7.7	25.8	28.8	1010.88	2018-06-13 13:51:33	
11.8	7.7	25.8	28.8	1010.88	2018-06-13 13:52:33	
11.8	7.7	26	28.8	1010.88	2018-06-13 13:53:33	
11.8	7.6	26	28.9	1035.54	2018-06-13 13:54:33	
11.8	7.6	26.1	29	1035.54	2018-06-13 13:55:33	
11.8	7.6	26.1	29	1035.54	2018-06-13 13:56:33	
11.8	7.6	26.1	29	1035.54	2018-06-13 13:57:33	
11.8	7.6	26.1	29	1035.54	2018-06-13 13:58:33	
11.8	7.6	26.1	28.9	1035.54	2018-06-13 13:59:33	
11.8	7.6	26.1	28.9	1035.54	2018-06-13 14:00:33	
11.8	7.6	26.1	28.9	1035.54	2018-06-13 14:01:33	
11.8	7.6	26.1	28.9	1035.54	2018-06-13 14:02:33	
11.8	7.6	26.1	28.9	1035.54	2018-06-13 14:03:33	
11.8	7.6	26.1	28.9	1035.54	2018-06-13 14:04:33	
11.8	7.6	26.1	28.9	1035.54	2018-06-13 14:05:33	

图 4 部分实验数据

数据分析模块数据预处理功能主要包括数据的缺失填充、异常检测和替换、数据归一化和降维处理等. 特征提取和选择模块主要功能为数据特征提取和选择. 数据预测主要功能为预测算法的训练、测试与运行.

将系统分析结果进行可视化如图 5 所示, 图中横坐标表示时间序列, 纵坐标表示发生故障的概率. 上方折线图为预测结果, 下方折线图为真实设备状态值. 当预测概率值大于 0.5 时系统将会发出警报, 提示设备将出现非正常运行状态. 从图中可以看出, 系统精确的预测出了设备的状态, 并且这一预测为实时的预测.

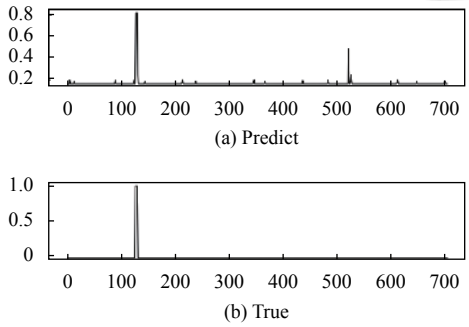


图 5 结果可视化展示

3.2 分布式计算测试

本次实验数据输入基本单位为 (4×2880)×13 的矩阵数据即一个数据样本. 分布式计算包括数据预处理、特征提取和特征选择过程, 本测试将采用两种测试条件; 第一种为同一时间向分布式计算平台发送多次任务请求, 且任务请求计算内容相同; 第二种为同一时间向分布式计算平台发送多次任务请求, 且任务请求计算内容不同.

第一种测试条件如表 4 所示, 相同时间提交所有任务计算内容相同.

表 4 分布式计算测试结果 1

指标	并发数						
	1	2	3	4	5	6	10
节点启动数	13	13	13	13	13	13	13
平台总耗时 (s)	3.3954	3.4643	3.5409	3.5507	3.5308	3.4982	3.6534

从表 4 中可看出, 同一时段提交任务数量有 1、2、3、4、5、6、10, 实验结果时间均为运行 100 次耗时平均值. 从表中可看出当提交相同任务时计算平台能合理的优化计算任务, 并且不增加计算耗时即平台能避免重复计算.

第二种测试条件如表 5 所示, 相同时间提交所有任务计算内容不同.

表 5 分布式计算测试结果 2

指标	并发数						
	1	2	3	4	5	10	11
节点启动数	13	13	13	14	15	17	17
平台总耗时 (s)	3.3648	6.2856	8.7683	10.686	12.8895	27.354	30.0862

从表 5 中可看出当提交任务数量上升时耗时也开始逐渐上升, 根据表中数据可看出当该平台可以并发处理多个任务, 且平均任务耗时为 3 s 左右. 为保证任务能在有效时间内获得计算结果 (数据采集间隔为 30 s), 该分布式计算平台至多同时提交 10 个不同任务.

4 总结

本文设计了一种 Hadoop 环境下的分布式物联网设备状态分析处理系统, 能够快速准确地进行设备状态预测. 为了提高数据处理效率, 数据预处理和特征提取方法完全基于 Dask 分布式计算框架, 并根据实际分布式计算中遇到的节点阻塞问题提出了 DRB 节点优化方案, 实现了分布式节点的智能启动、关闭和重启. 本系统的完成, 实现了工业物联网设备运行状态的精准诊断分析, 有效提高生产效率、降低资源能源消耗.

参考文献

1 孟小峰, 慈祥. 大数据管理: 概念、技术与挑战. 计算机研

究与发展, 2013, 50(1): 146–169. [doi: 10.7544/issn1000-1239.2013.20121130]
2 左宪章, 康健, 李浩, 等. 故障预测技术综述. 火力与指挥控制, 2010, 35(1): 1–5. [doi: 10.3969/j.issn.1002-0640.2010.01.001]
3 韩东, 杨震, 许葆华. 基于数据驱动的故障预测模型框架研究. 计算机工程与设计, 2013, 34(3): 1054–1058. [doi: 10.3969/j.issn.1000-7024.2013.03.061]
4 王桂兰, 赵洪山, 米增强. XGBoost 算法在风机主轴承故障预测中的应用. 电力自动化设备, 2019, 39(1): 73–77, 83.
5 曾凡琳, 宗群, 刘文静, 等. 带有时延和丢包的分布式网络化控制系统故障检测方法. 东南大学学报 (自然科学版), 2011, 41(S1): 173–178.
6 尹东辉, 崔伟. 大范围超短波通信网络故障基站检测系统设计. 计算机测量与控制, 2013, 21(6): 1437–1438, 1442. [doi: 10.3969/j.issn.1671-4598.2013.06.010]
7 焦亚军. 煤矿带式运送机故障检测和预防系统研究. 机械工程与自动化, 2019, (2): 218–219.
8 Wang YG, Wang S. Research and implementation on spatial data storage and operation based on Hadoop platform. Proceedings of the 2010 2nd IITA International Conference on Geoscience and Remote Sensing. Qingdao, China. 2010. 275–278.
9 刘树仁, 宋亚奇, 朱永利, 等. 基于 Hadoop 的智能电网状态监测数据存储研究. 计算机科学, 2013, 40(1): 81–84. [doi: 10.3969/j.issn.1002-137X.2013.01.019]
10 段军红, 张小东, 史庆华. 基于 Hadoop 的海量数据存储平台设计与开发. 电子技术与软件工程, 2017, (16): 162.
11 Mikkilineni R, Morana G. Infusing cognition into distributed computing: A new approach to distributed datacenters with self-managing services on commodity hardware (virtualized or Not). Proceedings of the 2014 IEEE 23rd International WETICE Conference. Parma, Italy. 2014. 131–136.
12 张维, 王玥, 罗珅. 基于分布式计算框架的大数据机器学习. 数字技术与应用, 2018, 36(9): 27–28.
13 嵇小飞. Web 服务器集群系统的自适应负载均衡调度. 中小企业管理与科技, 2016, (6): 153–154. [doi: 10.3969/j.issn.1673-1069.2016.06.120]
14 许治国. 利用 Keras 构建神经网络在空气质量预测中的应用. 环境监控与预警, 2018, 10(5): 18–21. [doi: 10.3969/j.issn.1674-6732.2018.05.004]
15 Rao Chandakanna V. REHDFS: A random read/write enhanced HDFS. Journal of Network and Computer Applications, 2018, 103: 85–100. [doi: 10.1016/j.jnca.2017.11.017]
16 Zhang LL, Yu SS, Ding XQ, *et al*. Research on IOT RESTful web service asynchronous composition based on BPEL. Proceedings of the 2014 6th International Conference on Intelligent Human-Machine Systems and Cybernetics. Hangzhou, China. 2014. 62–65.
17 安仲奇, 杜昊, 李强, 等. 基于高性能 I/O 技术的 Memcached 优化研究. 计算机研究与发展, 2018, 55(4): 864–874. [doi: 10.7544/issn1000-1239.2018.20160890]
18 Avni H, 王鹏. 面向数据库的持久化事务内存. 计算机研究与发展, 2018, 55(2): 305–318. [doi: 10.7544/issn1000-1239.2018.20170863]
19 罗亚非. 基于 TCP 的 Socket 多线程通信. 电脑知识与技术, 2009, 5(3): 563–565, 598. [doi: 10.3969/j.issn.1009-3044.2009.03.020]
20 杨双涛, 马志强, 窦保媛, 等. 一种 Yarn 框架下的异步双随机梯度下降算法. 小型微型计算机系统, 2017, 38(5): 1070–1075. [doi: 10.3969/j.issn.1000-1220.2017.05.031]
21 潘勇, 彭省临, 毛先成, 等. 无线自组网基于服务曲线的 TCP 流公平性改进. 小型微型计算机系统, 2010, 31(12): 2413–2417.
22 刘奕醇. CPU 多线程计算的瓶颈. 中国新通信, 2019, 21(3): 87–88. [doi: 10.3969/j.issn.1673-4866.2019.03.075]
23 Wang J, Yi Q, Wang Y, *et al*. Distributed data storage solution under sink failures in wireless sensor networks. The Journal of China Universities of Posts and Telecommunications, 2017, 24(2): 72–82, 102. [doi: 10.1016/S1005-8885(17)60201-2]
24 葛微, 罗圣美, 周文辉, 等. HiBase: 一种基于分层式索引的高效 HBase 查询技术与系统. 计算机学报, 2016, 39(1): 140–153. [doi: 10.11897/SP.J.1016.2016.00140]