doi:10.3969/j.issn.1673-9833.2019.06.003

基于 Hadoop 的鼓风机工业数据处理和存储系统

邢少波，张龙信，赵玉来，满君丰，周立前

（湖南工业大学 计算机学院，湖南 株洲 412007）

摘 要：针对现有工控网系统不能高效稳定地接收并处理来自工业现场的海量传感数据问题，基于 Hadoop 框架，采用消息队列遥测传输（MQTT）作为数据接入协议、EMQ 消息队列服务器提供订阅发布接口、Flink 作为流计算平台、Kafka 作为消息队列缓存分流以及 OpenTSDB 数据库作为数据存储，设计并实现鼓风机工业大数据处理和存储系统。实验表明，此系统消除了工业制造过程中产生的海量数据信息孤岛，实现生产设备状态动态监测及实时预警。

关键词：工业大数据；Hadoop；消息队列遥测传输；Flink；Kafka

中图分类号：TP311 **文献标志码**：A **文章编号**：1673-9833(2019)06-0015-08

引文格式：邢少波，张龙信，赵玉来，等 . 基于 Hadoop 的鼓风机工业数据处理和存储系统 [J]. 湖南工业大学学报，2019，33(6)：15-22.

Hadoop-Based Industrial Data Processing and Storage System for Blowers

XING Shaobo，ZHANG Longxin，ZHAO Yulai，MAN Junfeng，ZHOU Liqian

（College of Computer Science，Hunan University of Technology，Zhuzhou Hunan 412007，China）

Abstract：In view of the flaw found in the existing industrial controlling network which fails to effectively receive and deal with mass sensor data from the industrial field, an industrial data processing and storage system for blowers, based on the Hadoop framework, has thus been designed and implemented. This system uses the message queue telemetry transport (MQTT) as the data access agreement, with EMQ message queue server as the subscription and publishing interface, with Flink as flow computing platform, and with Kafka as message queue buffer shunt and OpenTSDB database as data storage. Experimental results show that the new system helps to eliminate the large amount of data and information islands generated in the process of industrial manufacturing, thus realizing the dynamic state monitoring and real-time warning of production equipment.

Keywords：industrial big data；Hadoop；MQTT；Flink；Kafka

1 研究背景

当前制造业正处在由数字化、网络化向智能化发展的重要阶段 [1]，其核心基于海量工业数据的全面感知，通过端到端的数据深度集成与建模分析，实现智

收稿日期：2019-06-13

基金项目：国家重点研发计划基金资助项目专项子课题（2018YFB1003401），国家自然科学基金资助项目（61702178），湖南省自然科学基金资助项目（2019JJ50123，2018JJ4063），湖南省教育厅重点基金资助项目（17A052），湖南省大学生创新基金资助项目（S201911535023）

作者简介：邢少波（1997-），男，河南洛阳人，湖南工业大学学生，主要研究方向为工业大数据处理，E-mail：1053302852@qq.com

通信作者：张龙信（1983-），男，湖南长沙人，湖南工业大学讲师，博士，硕士生导师，主要从事大数据处理，高性能计算方面的教学与研究，E-mail：longxin.zhang@163.com

能化的决策与控制指令，形成智能化生产、网络化协同、个性化定制、服务化延伸等新型制造模式。在这样的背景下，传统数字化工具已经无法满足需求，工业数据的爆发式增长呼唤新的数据处理工具。随着工业系统由物理空间向信息空间、从可见世界向不可见世界延伸，工业数据采集范围不断扩大，数据的类型和规模都呈指数级增长。这需要一个全新数据管理工具，以实现海量数据低成本、高可靠地管理和存储。

受益于自动化技术的发展与普及，制造业生产效率得到大幅提高，伴随而来的问题是如何对高速的生产流程进行有效地管理、协调与监控。这些问题的本质是如何高效地传递信息并反馈，即如何快速将生产信息与管理决策信息连接起来，消除“信息孤岛”。信息交换恰恰是互联网最大的特点，因此近几年互联网在工业领域的应用呈大幅增长态势，被称之为“工业互联网”。2013 年德国推出“工业 4.0”战略 [2]、2014 年中国提出“中国制造 2025”计划等都是对工业互联网的推广应用 [3-4]。在已有的研究中，沙乐天等 [5] 针对工业物联网高速发展过程中存在的后门隐私信息泄露问题，基于隐私数据的特征定义提取相应的语义，有效地降低了工业物联网中的隐私信息泄露风险。詹剑锋等 [6] 针对大数据系统基准测试难以满足大数据领域的多样性，研发出覆盖多系统、数据管理和体系结构的开源测试基准 BigDataBench，目前该基准已成功被应用于学术界和工业界。徐泉等 [7] 总结了工业大数据近年来的发展情况，全面阐述了工业云的发展现状、相关核心技术、存在的不足、服务的评估方式及未来的发展态势。鉴于工业大数据数量巨大、数据多源、价值分散等特点给数据分析带来的挑战，孔宪光等 [8] 提取工业数据特征时采用滑动窗口动态更新数据，提出可靠的增量主成分分析策略，有效地改善了工业数据流的提取精度。汪星等 [9] 针对工业大数据分析过程中样本维数过高导致增量判别失效的问题，引进滑动窗口和熵值法进行工业大数据的特征提取，有效地提高了数据分析时的判别能力。田保军等 [10] 针对大数据中的数据性稀疏问题，基于 Hadoop 平台，采用协同过滤策略提取大数据的特征，提高了推荐的成功率。

然而，传统的数据分析和处理系统无法应对海量工业数据的全面感知、网络传输、批量处理及智能化的决策实现。针对这一问题，本文设计了基于 Hadoop 的鼓风机工业数据处理和存储系统。本研究将设计工业大数据处理系统，主要内容包含系统的结构设计和逻辑设计，以及关键的数据接入层、数据中间层设计、数据库层设计、核心算法描述等，最后将工业现场采集的数据进行测试和分析，验证系统的可行性。

2 工业大数据处理系统设计

考虑到工业大数据处理系统在工业互联网中起着承上启下的作用，因此，系统设计必须满足低耦合、可扩展、易维护等要求。课题组在系统设计时借鉴了网络协议分析中的分层思想，将不同功能解耦到不同服务层次，每一层依赖于其上一层提供的功能，如消息中间层依赖数据接入层提供的数据，应用层依赖消息中间层提供的消息等。这样各功能模块既可以专注于本层的功能实现，降低实现难度，又实现了模块间低耦合、可扩展的目标，从而提高了系统的可扩展性和兼容性。同时，也提高了系统负载均衡能力和系统的可维护性，改善了系统的服务质量。

系统服务架构主要由 MQTT（message queuing telemetry transport）服务器、Kafka 服务器集群、Flink 计算集群、数据库服务器集群等组成，各分布式服务进程在管理节点的协调下向外提供高并发、高可用的服务，不同服务在以太网中通过不同的协议和上下游交流信息，实现功能的解耦分离。

OpenTSDB 数据库依赖的 HBase[11]、Hadoop[11]，以及 HBase 和 Kafka 依赖的 Zookeeper 未列出，它们是整个系统的基石，为系统提供了多副本、高可用的功能。Flink 集群 [12] 中目前运行了 2 个 Flink 程序：一个属于数据接入层，负责订阅 MQTT 消息，经转换计算后写入 Kafka 消息队列；另一个属于应用层的异常模块，负责判断异常信息。

工业大数据处理系统的体系架构如图 1 所示。

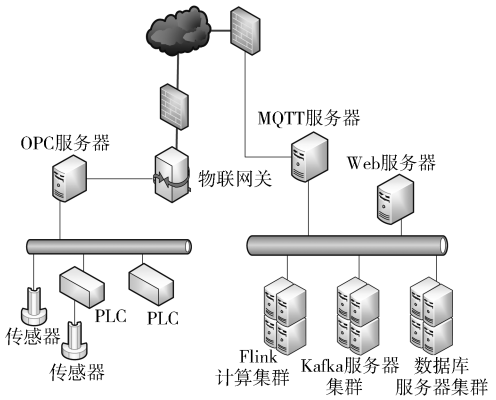

图 1 工业大数据处理系统架构图

Fig. 1 Architecture diagram of the industrial big data processing system

MQTT 服务器的实现采用的是 EMQ（elastic message queue）服务器，系统中考虑到服务器资源

紧张以及 EMQ 并发承受能力较高等因素，采用了单台部署模式。当应用场景并发量增大，系统可以采用分布式部署，扩展至多个物理节点。为描述方便，本系统将 MQTT 服务器以及 3 个 Hadoop 生态应用集群安装在同一个服务器集群中，推荐的做法是将 Zookeeper 以及 Kafka 等为多方提供服务的进程部署在单独的集群中，并部署资源监控维护进程进行资源管理。

如图 2 所示，系统整体的逻辑结构由数据接入层、消息中间件层、数据库层和应用层 4 层组成。层与层之间通过各自的协议与上下游交换重要信息，这样既没有耦合又在数据流上保障连接顺畅。数据接入层与消息中间层都向外提供接口，外部系统可以根据需要选择连接偏底层的数据接入层或以应用为主的消息中间层。应用层包含异常报警模块和扩展模块等，可以通过消息订阅 API 以及数据库查询接口任意添加应用模块，如生成报表模块、机器学习训练模型等。作为系统的可视化部分，本系统提供的数据接口采用 Web 系统集成调用。

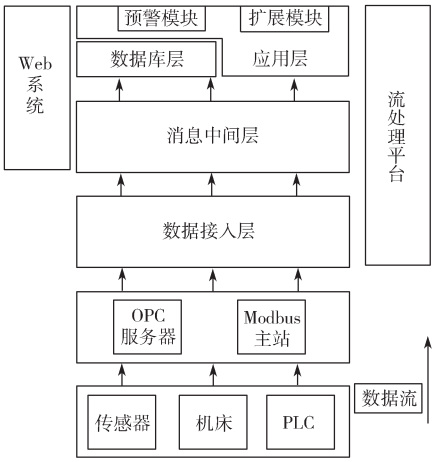

图 2　工业大数据处理系统逻辑框图

Fig. 2　Logical block diagram of industrial big data processing system

值得一提的是，数据流并非严格依照图 2 中的流向流动，例如预警模块产生的预警数据也会写入消息中间层和数据库中供预警信息消费者使用。控制信息则是由上到下从 Web 系统下达到消息中间层，再由数据接入层的程序订阅控制消息下达到物联网中。

3　关键技术研究

3.1　数据接入层设计

数据接入层作为系统最底端与物联网直接通信的部分，是整个系统的基础。系统采用 MQTT 协议作为通信协议，并使用开源的高性能 MQTT 服务器 EMQ 作为代理服务器。

MQTT 是发布订阅（publish/subscribe）模式的消息协议。与 HTTP 协议请求响应（request/response）模式不同，MQTT 发布者与订阅者之间通过“主题”（topic）进行路由。MQTT 的主题格式类似 Unix 文件路径的树形结构，如“data/region1/gateway1/”，并且支持“+”“#”等通配符。“+”通配一层，“#”通配多层，如订阅“data/#”即为订阅“data/”下所有层的主题。通配符只支持订阅者订阅时使用，发布者只能向特定主题发布消息。在本系统中，考虑到要支持较多数量的物联网关，并且可以任意增加或缩减，主题路由设置需具有伸缩性、扩展性。

主题路由共分三级。首先，将数据的 2 种类型作为主题路由的第一级：“data/”和“cmd/”。“data/”由物联网关发布采集到的数据，数据接入层订阅；“cmd/”下由数据接入层发布控制信息，物联网关订阅。第二级由区域号组成，方便了区域管理和权限控制。主题路由第三级则由唯一的网关编号来实现，如“data/region1/gateway1”是区域 1 物联网关 1 发布数据的主题，“cmd/region1/gateway2”是区域 1 物联网关 2 订阅命令的主题。

在这种主题结构下，数据接入层可以使用通配符“data/#”订阅所有区域的所有网关发布数据，同样也可以向“cmd/region2/+”或“cmd/region2/#”主题发布控制命令实现区域开关等控制，体现了本系统的伸缩性、扩展性、灵活性等特点。

消息转换格式设计如图 3 所示，结合系统中维护的映射关系和校零值等参数，计算程序将面向物联网的网关和采集点标识的数据转换为面向用户的机器和监测点（鼓风机油箱温度）标识的数据。

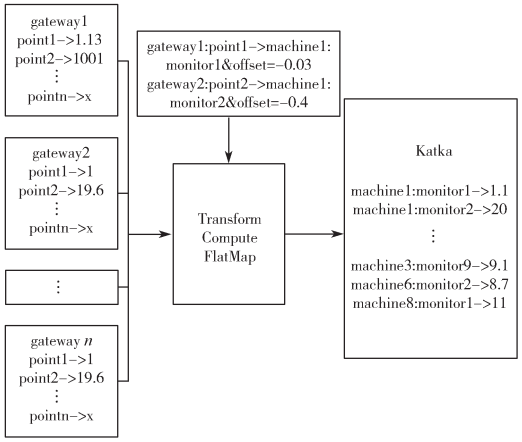

图 3　消息格式转换设计

Fig. 3　Message format transformation design

3.2 数据中间层设计

消息中间件选用 Kafka 消息队列。系统各层、各模块间的信息交流几乎都是在 Kafka 中进行，图 4 展示了各模块依赖 Kafka 发布订阅的消息。

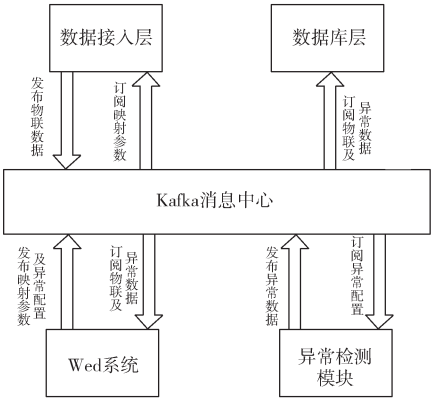

图 4 各模块与消息中间层的通信设计

Fig. 4 Communication design between individual module and message middle layer

数据接入层需要从 Kafka 中获取映射参数的更新，更重要的是向 Kafka 中发布物联数据，数据包括 PLC（programmable logic controller）寄存器的值和传感器数据。而数据库层只需要订阅其中的物联数据及异常模块产生的异常数据并进行持久化，不需要发布数据。Web 系统作为与用户交流的门户，需要向 Kafka 中发布用户配置的映射参数、异常配置等数据，同时更重要的是订阅实时的物联数据，以进行可视化。异常检测模块与需要订阅物联数据和异常配置来判别异常，并记录异常持续时间。当有异常时需要发布到 Kafka 中供数据库持久化和 Web 系统向用户展示。

3.3 数据库层设计

数据库层负责物联数据持久化，为下游应用提供了查询历史数据的接口。数据库层包括了写入模块、数据库模块和查询接口模块。写入模块负责订阅 Kafka 中的物联数据和异常数据，并写入数据库。数据库模块则由 OpenTSDB 承担，最后在 Web 接口模块中提供了方便下游应用的查询接口，使用 Springboot 程序提供的 RESTful（representational state transfer）API 实现。

OpenTSDB 数据库作为一款时序数据库，存储后端为 HBase，拼接时间戳与若干标签键值对作为 HBase 的行键。在进行预分区后，由于 HBase 的分区按照行键顺序进行分割存储，而写入数据时的时间戳间隔时间极短，如果不作任何处理会产生热点问题。即使 HBase 有多个分区，由于时间戳位于行键高位，行键在同一范围连续形成，会存储到同一个分区中，所以 OpenTSDB 在 2.2 版本后集成了加盐功能，即在行键高位插入一个随机数，将数据分散到不同分区中，有效地解决了热点问题。借助 HBase 存储中的树形索引表（tree table），OpenTSDB 可以快速响应查询请求 [13]。

图 5 为 OpenTSDB 在 HBase 中加盐后的行键组成示意图。

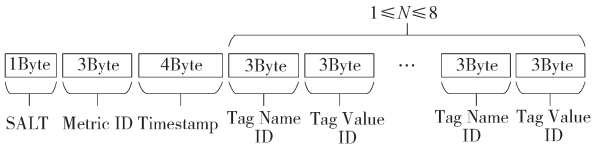

图 5 OpenTSDB 在 HBase 中加盐后的行键组成

Fig. 5 Row key composition of OpenTSDB with a salt addition to HBase

由图 5 所示的行键组成可以看出，Tag 辅助索引最多 8 个，在当前场景下，物联数据系统选择了两级辅助索引：机器编号和监测点编号，值为实时数据，可充分满足当前的需求。倘若需要扩展，如增加分区索引等，只需要动态添加，并不需要暂停服务。而对于异常记录数据，由于用户配置的异常信息中包含了监测点索引信息，该表中的两级索引构成为机器编号和异常编号。其值为异常的持续时间，而记录的时间戳为异常开始时间。

4 核心算法设计

4.1 数据处理算法

数据处理算法如算法 1 所示。

算法 1 数据处理算法

Input: MQTT messages

Output: Kafka messages

1 Connect to EMQ broker and subscribe the data topic

2 Get total parameters of calculation and identity mapping from Redis

3 **while** Kafka topic of params has message **do**

4 Update or add the parameters

5 **end while**

6 **while** topic has new message **do**

7 Get MQTT message

8 **if** message is illegal **do**

9 Throw message and record exception

10 **end if**

11 **for** each record **in** MQTT message **do**

12 **if** the record has a calculation parameter **do**

13　Get the calculation parameter of the record
14　Compute and zero correction according to the parameter
15　**end if**
16　**if** the record has an identification of user definition **do**
17　Change the identification from gateway to user definition
18　**end if**
19　Set the key of Kafka message to keep order
20　Publish the record to the data topic of Kafka
21　**end for**
22　**end while**

数据处理算法的功能主要由 3 部分组成：第一部分对应算法 1 的 Step1~5，从 Redis 全量获取用于计算和校零的参数以及用户定义的传感器数据的标识，并订阅 Kafka 参数主题实时维护该参数。在用户更改计算参数或新增传感点配置（如调整校零值的操作）时能够实时感应，以较小时延表示，使用户能及时看到调整后的数据。

第二部分为最重要的计算和更改标识，对应算法 1 的 Step 6~18。首先判断获取的 MQTT 消息中网关 Id 是否有记录，没有则为不合法，将丢弃消息并记录异常信息。然后对 MQTT 消息中的每一条传感器记录进行操作。如果参数集合中含有该记录的计算参数，则结合参数对该记录值进行计算。如果用户定义标识集合中含有该记录当前标识对应的用户定义标识，则将标识更换，为数据流下游的操作提供便利。

第三部分对应算法 1 的 Step 19~20，功能为构造指定 Key 值为记录的 Kafka 消息标识，以保证同一传感器数据落入同一 Kafka 分区中，从而保证高频数据的顺序。若将记录以单条形式发布至 Kafka 中，此举将对 Kafka 最大吞吐量造成轻微影响，但是更加贴近生产情况并可以满足下游的需求。这样虽然牺牲小部分性能，却能换取更大的收益。

4.2　数据存储算法

数据存储算法如算法 2 所示。数据库写入是单进程执行，由于构造字符串格式、拼接字符串以及网络通信耗时较多，单线程下批量订阅 Kafka 消息后，同时拼接字符串并发送数据写入报文，这很可能导致阻塞时间过长。算法中消费 Kafka 消息的速度低于生产者生产消息的速度，最终导致 Kafka 消息积压以及存储数据延时过大。算法 2 中采用订阅线程订阅 Kafka 数据记录，写入线程复制消息异步构造写入报文并发送，配合同步锁以避免两个线程出现读写数据的冲突问题。

首先订阅线程得到数据后写入订阅数据列表中。当达到批量写入条件（0.5 s 时间或 50 000 条记录）后，写入线程构造写入列表并加锁，然后复制记录列表。此时订阅线程检测到同步锁状态改变，暂停向记录列表中写入，并等待解锁。当写入线程复制完成，并初始化记录列表后完成解锁，以保证写入线程复制数据并初始化记录列表过程中不会有新数据写入导致丢失。解锁后订阅线程与写入线程同时开始工作。订阅线程继续订阅消息写入记录列表。写入线程构造、拼接 http 报文后向 OpenTSDB 服务器发送数据。需要注意的是，多个 OpenTSDB 实例需要使用 Nginx 或在程序中使用负载均衡策略进行轮换服务来增大吞吐量。

算法 2　数据存储算法

Input: Kafka records
Output: DB records
1　Connect to Kafka broker and subscribe the data topic and the alarm data topic
2　Create a list for record called record_list
3　Create a lock for synchronization of two threads called syc_lock
　　subscribe thread:
4　**while** have new record **do**
5　Get Kafka record
6　**if** the syc_lock **is** unlocked
7　Put the record to the list
8　**else**
9　Wait and sleep for unlocked
10　**end if**
11　**end while**
　　writing thread:
12　Create another list called write_list for writing
13　**while** length of the list achieve 50k or the time since last write achieve 0.5s **do**
14　Lock the syc_lock
15　Copy the record_list to write_list
16　Initialize the record_list
17　Unlock the syc_lock
18　**for** each record **in** write_list
19　Build the format of record for writing
20　Splice http messages
21　**end for**
22　Send writing http messages to OpenTSDB server
23　**end while**

5 实验部分

5.1 实验环境

系统服务器集群配置包含 3 个节点，每个节点使用 4 核 CPU 和 8 GB 内存。服务器运行的系统为 CentOS 7。集群中各节点的域名分别为 hadoop1、hadoop2、hadoop3。Flink 集群选择了最新的稳定版 1.8.0。系统选择将 Flink 的 master 进程 jobManager 安装在 hadoop2 中。将配置文件修改好后分发到其他主机中，并且在 hadoop2 配置目录下 FLINK_HOME/conf/slaves 添加 hadoop1、hadoop2、hadoop3。启动时系统会将 slaves 文件中所有主机通过 SSH 启动 slave 进程。

Kafka 依赖于 Zookeeper 提供的一致性服务，组件间相互依赖的要注意版本兼容问题，如果不兼容可能导致错误，之后部署的 HBase、OpenTSDB 等也存在同样的问题。Kafka 的版本选择了 2.2.0，对应的 Zookeeper 版本选择了 3.4.10。

OpenTSDB 采用最新稳定版 2.4.0，对应的 HBase 版本为 1.3.0，Zookeeper 版本为 3.4.10。需要注意的是，系统部署的时候关闭 HBase 自带的 Zookeeper 服务。

5.2 系统测试

测试程序模拟物联网关向 EMQ 发送 MQTT 消息，每条消息中包含 5 个采集点值。在压力测试中，由于单线程发送速度有限，测试程序使用三线程发送数据，每个线程的发送速度为 6 000~7 000 条 /s。在本地四核 CPU 机器上，3 个线程均以 20 000 条 /s 记录左右的速度发送，进而达到 10 万采集点次 /s。需要注意的是发送过程中不仅要变换消息时间戳的值，也需要更改采集点标识的值。否则，消息单线程发送速度超过 1 000 条 /s 后，精确到毫秒级的时间戳会发生重复，导致数据库记录被覆盖。

程序发送约 15 min 后查看 EMQ 管理界面中的消息数，如图 6 所示。

图 6 EMQ 管理界面中的消息数截图

Fig. 6 Screenshot of EMQ message number

由图 6 可知，EMQ 接收的消息数为 2 052 万。同时对比 OpenTSDB 数据库中数量：“SELECT COUNT (*) FROM measurement1”。数据库中的记录数为 1.03 亿，考虑检索数据库记录数花费时间较长，并且此时数据仍在插入，导致数据库记录数目多于 2 052*5。可见在 15 min 左右的时间内，系统接收处理了超过 1 亿采集点次，承载并发的能力达到了 10 万 /s 的要求。

在终端中使用客户端查询 OpenTSDB 最新的 20 条记录：“SELECT * FROM measurement1 ORDER BY time desc limit 20”，结果如图 7 所示，其中记录较多的是真实的温度传感器数据，为 24~28 ℃；湿度传感器数值，为 56.1%；气压为 100.22 kPa。

time	GW_id	IOP_id	value
1557477195000000000	KSD0186701	klha_pre:pre	100.22
1557477195000000000	KSD0186701	klha_gw:channel2	121.63
1557477195000000000	KSD0186701	klha_gw:channel1	24.49
1557477195000000000	KSD0186701	klha_gw2:channel4	0
1557477195000000000	KSD0186701	klha_gw2:channel3	27.7
1557477195000000000	KSD0186701	klha_gw2:channel2	24
1557477195000000000	KSD0186701	klha_gw2:channel1	23.2
1557477195000000000	KSD0186701	klha:temp	24.3
1557477195000000000	KSD0186701	klha:humi	56.1
1557477195000000000	KSD0186701	klha:DewPoint	15
1557477195000000000	KSD0186701	fx3u:M1	0
1557477195000000000	KSD0186701	fx3u:M0	0
1557477195000000000	KSD0186701	fx3u:D101	0
1557477195000000000	KSD0186701	fx3u:D100	0
1557477194000000000	KSD0186701	klha_pre:pre	100.22
1557477194000000000	KSD0186701	klha_gw:channel2	121.65
1557477194000000000	KSD0186701	klha_gw:channel1	24.45
1557477194000000000	KSD0186701	klha_gw2:channel4	0
1557477194000000000	KSD0186701	klha_gw2:channel3	28
1557477194000000000	KSD0186701	klha_gw2:channel2	25.2

图 7 终端查看数据库实时记录

Fig. 7 Real-time record of database in terminal view

在终端上使用“SELECT * FROM alarm_measurement ORDER BY time DESC”查看异常记录，结果如图 8 所示。时间戳为开始时间，duration 为持续时间，单位为 s。

time	GW_id	IOP_id	alarm_code	alarm_msg	duration	level	
1552872670000000000	KSD0186711	Smart200:vd	9f08ee59fe7c4b03	0219测试设备后主轴承温度异常	122	2	
1552821620000000000	KSD0186701	klha:humi	db2017ec3d3a41c5	室外温度异常	3705185	1	
1552312107000000000	KSD0186711	fx3u:d103	82300c331a2a42d9	11设备前主轴承温度异常	560685	1	
1552033622000000000	KSD0186711	Smart200:vd	9f08ee59fe7c49d3	0219测试设备后主轴承温度异常	61350	2	
1551785282000000000	KSD0186711	fx3u:d103	82300c331a2a42d9	11设备前主轴承温度异常	526776	1	
1551785044000000000	simulation1	fx3u:d124	zhcg2026_simulation1	Z振动异常	13	1	
1551777325000000000	simulation1	fx3u:d124	zhcg2026_simulation1	Z振动异常	10	1	
1551766968000000000	simulation1	fx3u:d124	zhcg2026_simulation1	Z振动异常	10	1	
1551766496000000000	simulation1	fx3u:d124	zhcg2026_simulation1	Z振动异常	10	1	
1551765409000000000	simulation1	fx3u:d124	zhcg2026_simulation1	Z振动异常	10	1	
1551756890000000000	simulation1	fx3u:d124	zhcg2026_simulation1	Z振动异常	10	1	
1551733835000000000	simulation1	fx3u:d124	zhcg2026_simulation1	Z振动异常	10	1	
1551731867000000000	simulation1	fx3u:d124	zhcg2026_simulation1	Z振动异常	12	1	
1551724814000000000	simulation1	fx3u:d124	zhcg2026_simulation1	Z振动异常	10	1	
1551704292000000000	simulation1	fx3u:d124	zhcg2026_simulation1	Z振动异常	12	1	
1551693371000000000	simulation1	fx3u:d124	zhcg2026_simulation1	Z振动异常	15	1	
1551680032000000000	simulation1	fx3u:d124	zhcg2026_simulation1	Z振动异常	12	1	
1551684592000000000	simulation1	fx3u:d124	zhcg2026_simulation1	Z振动异常	10	1	
1551678197000000000	simulation1	fx3u:d124	zhcg2026_simulation1	Z振动异常	16	1	
1551676065000000000	simulation1	fx3u:d124	zhcg2026_simulation1	Z振动异常	10	1	
1551674622000000000	simulation1	fx3u:d124	zhcg2026_simulation1	Z振动异常	10	1	
1551672174000000000	simulation1	fx3u:d124	zhcg2026_simulation1	Z振动异常	10	1	

图 8 终端查看异常记录

Fig. 8 Exception record in the terminal view

使用开源图表工具 Grafana 连接 OpenTSDB 查询一个采集点（如 gw1:pt1）的历史记录（按秒取样），查看 0.5 h 内记录；如图 9 所示，生成的模拟数据为 0~22 的随机数。

在订阅本系统实时数据的 Web 系统（外部系统）中查看两个实时参数变化的动态图表，如图 10 所示，Web 系统订阅本系统 Kafka 中特定主题数据后筛选用

户正在查看的数据发送给前端进行显示。

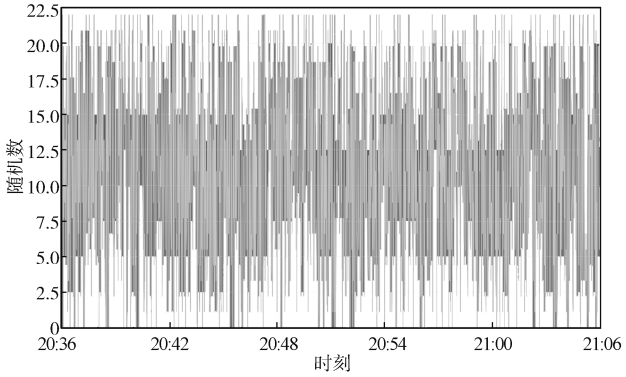

图 9　Grafana 取样查看

Fig. 9　Sampling view with Grafana

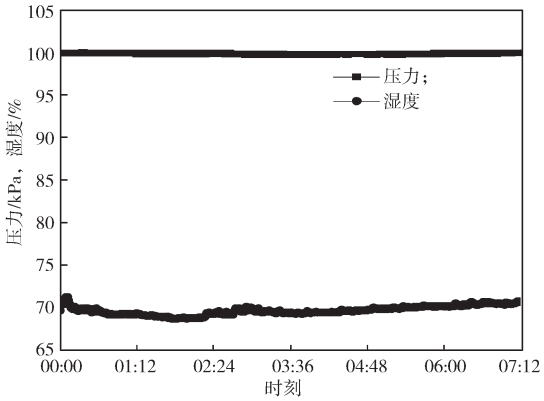

图 10　Web 系统实时参数查看

Fig. 10　Real-time parameters viewing in the web system

一条传感记录 (如温度传感记录)，从生成到展现给用户经历的传输历程如下：传感器生成记录后，PLC 通过模拟量接口或 Modbus 协议不断请求数据保存至寄存器中，物联网关通过 OPC 服务器定时向 PLC 查询寄存器值，然后将若干寄存器值构造 JSON 数据后通过 MQTT 协议发送给 EMQ 服务器。至此数据从工控网传输至因特网。此后，系统消息中间层的 Flink 程序通过订阅 MQTT 特定主题从 EMQ 服务器拉取消息，结合用户配置的校零值等参数计算后写入 Kafka 中，接入本系统的 Web 系统通过订阅 Kafka 特定主题获取到数据后发送至前端图表中展示。为用户展示的同时，数据还将并行写入数据库以及检测是否异常。

实验测试后发现，影响系统并发处理量较大的因素是外部物联网关发送 MQTT 报文每条消息中的记录数。其原因是数据接入程序订阅 MQTT 消息为实时处理，而化零为整的批处理是大数据领域提高吞吐量的常用手段，但批量过大同样会降低整体吞吐量（系统吞吐量定义为在不落后 MQTT 生产的前提下接收、处理 MQTT 消息并发布至 Kafka，然后存入数据库的最大每秒处理监测点数量）。

图 11 展示了 MQTT 消息负载记录数与系统吞吐量的关系。

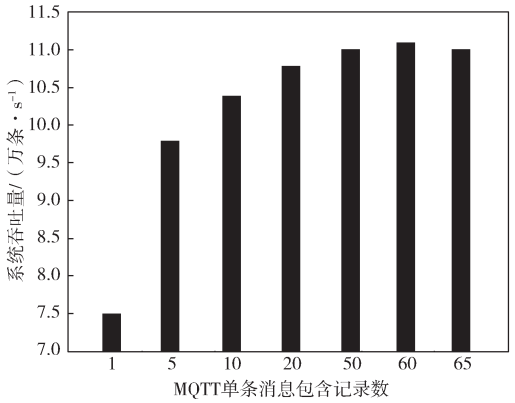

图 11　MQTT 单条消息包含记录数与系统吞吐量的关系

Fig. 11　Relationship between the number of records contained in a single message and the system throughput

假设每条记录为一个传感器的数据 (实际也可能是 PLC 中某个寄存器值等)，通过模拟物联网关构造不同记录数的 MQTT 消息发布至 EMQ 服务器来测试系统整体性能与物联网关发布消息的关系。由图 11 可以看到，当 MQTT 消息仅包含单条记录时，系统吞吐量最低时约 7.5 万条 /s；MQTT 单条消息提升至包含 5 条记录时吞吐量显著提升至 9.8 万条 /s；而提升至 60 条记录前吞吐量一直随着记录条数增加而升高，在 60 条记录左右达峰值 11.1 万条 /s，之后缓慢下降。不难得出，当每条 MQTT 报文中携带 60 条左右消息记录、消息大小为 3 kB 时，系统的吞吐量达到最大值。

6　结语

信息技术的高速发展，工业 4.0 标准的提出，以及《中国制造 2015》的推广，核心特征都指向物联网的智能、互联和数据分析处理。工业大数据处理的主旨都在于充分利用现代智能信息采集、信息传输，通过主流的大数据分析平台进行处理，并保存提取的特征信息，分析规律，进行健康诊断和故障预测。本文设计并实现了基于 Hadoop 框架的鼓风机工业大数据处理和存储系统，分析接入工业现场数据的需求、并发量及接入方式，设计 MQTT、Kafka 两个订阅发布模型的主题设计，处理后的数据使用开源 OpenTSDB 工具，从 Kafka 实时写入数据库中，并进行设备的异常检测和预警分析。设计的系统已经在株洲市鼓风机厂上线使用，有效地检测出鼓风机在生产过程中潜在的故障，大大提高了企业的生产效率，使得工业生产过程更加智能化和现代化。

参考文献：

[1] 卫凤林，董 建，张 群 .《工业大数据白皮书 (2017 版)》解读 [J]. 信息技术与标准化，2017(4)：13−17.
WEI Fenglin，DONG Jian，ZHANG Qun. Interpretation of Industrial Big Data White Paper (2017)[J]. Information Technology & Standardization，2017(4)：13−17.

[2] 宁振波，吴元良 . 从先进制造业的发展看实施工业 4.0 的前提条件 [J]. 航空制造技术，2014(18)：37−40.
NING Zhenbo，WU Yuanliang. Precondition of Industrie 4.0 Implementation Based on the Development of Advanced Manufacturing Industry[J]. Aeronautical Manufacturing Technology，2014(18)：37−40.

[3] 王建民 . 工业大数据技术综述 [J]. 大数据，2017，3(6)：3−14.
WANG Jianmin. Survey on Industrial Big Data[J]. Big Data Research，2017，3(6)：3−14.

[4] 逯 衍 . 大数据平台数据采集系统的设计与实现 [D]. 北京：北京交通大学，2018.
LU Yan. Design and Implementation of Data Acquisition System of Big Data Platform[D]. Beijing：Beijing Jiaotong University，2018.

[5] 沙乐天，肖 甫，陈 伟，等 . 面向工业物联网环境下后门隐私泄露感知方法 [J]. 软件学报，2018，29(7)：1863−1879.
SHA Letian，XIAO Fu，CHEN Wei，et al. Leakage Perception Method for Backdoor Privacy in Industry Internet of Things Environment[J]. Journal of Software，2018，29(7)：1863−1879.

[6] 詹剑锋，高婉铃，王 磊，等 . BigDataBench：开源的大数据系统评测基准 [J]. 计算机学报，2016，39(1)：196−211.
ZHAN Jianfeng，GAO Wanling，WANG Lei，et al. BigDataBench：An Open-Source Big Data Benchmark Suite[J]. Chinese Journal of Computers，2016，39(1)：196−211.

[7] 徐 泉，王良勇，刘长鑫 . 工业云应用与技术综述 [J]. 计算机集成制造系统，2018，24(8)：1887−1901.
XU Quan，WANG Liangyong，LIU Changxin. Industrial Applications，Technologies，Challenges and Trends for Industrial Cloud[J]. Computer Integrated Manufacturing Systems，2018，24(8)：1887−1901.

[8] 孔宪光，章 雄，马洪波，等 . 面向复杂工业大数据的实时特征提取方法 [J]. 西安电子科技大学学报 (自然科学版)，2016，43(5)：70−74，152.
KONG Xianguang，ZHANG Xiong，MA Hongbo，et al. Real Time Feature Extraction Method for Complex Industrial Big Data[J]. Journal of Xidian University (Natural Science)，2016，43(5)：70−74，152.

[9] 汪 星，黄小瑜，刘瑄璞，等 . 面向工业大数据的多层增量特征提取方法 [J]. 西安电子科技大学学报 (自然科学版)，2018，45(4)：106−111.
WANG Xing，HUANG Xiaoyu，LIU Xuanpu，et al. Multi-Layer Incremental Feature Extraction Method for Industrial Big Data[J]. Journal of Xidian University (Natural Science)，2018，45(4)：106−111.

[10] 田保军，胡培培，杜晓娟，等 . Hadoop 下基于聚类协同过滤推荐算法优化的研究 [J]. 计算机工程与科学，2016，38(8)：1615−1624.
TIAN Baojun，HU Peipei，DU Xiaojuan，et al. Optimization of the Collaborative Filtering Recommendation Algorithm Based on Clustering Under Hadoop[J]. Computer Engineering and Science，2016，38(8)：1615−1624.

[11] WEN X，Bei Z D，XU C Z，et al. ATH：Auto-Tuning HBase's Configuration Via Ensemble Learning[J]. IEEE Access，2017，5：13157−13170.

[12] CHEN C，LI K L，OU A J，et al. GPU-Accelerated Parallel Hierarchical Extreme Learning Machine on Flink for Big Data[J]. IEEE Transactions on Systems，Man，and Cybernetics：Systems，2017，47(10)：2740−2753.

[13] 夏茂森 . 流程工业智能工厂建设技术的研究 [J]. 信息技术与信息化，2013(6)：46−52.
XIA Maosen. Research on Intelligent Plant Construction Technology of Process Industry[J]. Information Technology & Informatization，2013(6)：46−52.

（责任编辑：申 剑）