Article

Device Data Ingestion for Industrial Big Data Platforms with a Case Study †

Cun Ji [1], Qingshi Shao [1], Jiao Sun [1], Shijun Liu [1,2,*], Li Pan [1,2], Lei Wu [1,3] and Chenglei Yang [1,2,*]

1 School of Computer Science & Technology, Shandong University, Jinan 250101, China; jicun@sdu.edu.cn (C.J.); sdusqs@163.com (Q.S.); jiaosun_baby@163.com (J.S.); panli@sdu.edu.cn (L.P.); i_lily@sdu.edu.cn (L.W.)
2 Engineering Research Center of Digital Media Technology, Shandong University, Jinan 250101, China
3 Beijing Key Laboratory on Integration and Analysis of Large-scale Stream Data, North China University of Technology, Beijing 100144, China
* Correspondence: lsj@sdu.edu.cn (S.L.); chl_yang@sdu.edu.cn (C.Y.); Tel.: +86-137-9316-5737 (S.L.); Fax: +86-531-8839-0059 (S.L.)

† This paper is an extended version of our paper Cun, J.; Shijun, L.; Chenglei, Y.; Lei, W.; Li, P. IBDP: An Industrial Big Data Ingestion and Analysis Platform and Case Studies. In Proceedings of the 2015 International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI), Beijing, China, 22–23 October 2015; pp. 223–228.

Academic Editors: Yunchuan Sun, Antonio Jara and Shengling Wang
Received: 16 January 2016; Accepted: 18 February 2016; Published: 26 February 2016

Abstract: Despite having played a significant role in the Industry 4.0 era, the Internet of Things is currently faced with the challenge of how to ingest large-scale heterogeneous and multi-type device data. In response to this problem we present a heterogeneous device data ingestion model for an industrial big data platform. The model includes device templates and four strategies for data synchronization, data slicing, data splitting and data indexing, respectively. We can ingest device data from multiple sources with this heterogeneous device data ingestion model, which has been verified on our industrial big data platform. In addition, we present a case study on device data-based scenario analysis of industrial big data.

Keywords: big data; internet of things; industrial internet of things; device data ingestion

1. Introduction

The Internet of Things (IoT) has been defined as communication between and integration of smart objects (things) [1]. Nowadays, the rapid development of IoT has aroused increasing interest in a variety of fields [2]. In the industrial field especially, more and more sensors are being incorporated into smart products, manufacturing equipment, and production monitoring. Smart manufacturing, which has become a vital component of manufacturing in the Industry 4.0 era, is currently facing challenges related mainly to the following three issues:

1. Heterogeneous data. There are different types of smart devices in an enterprise. Each device has single-parameter or multiple-parameter sensors, resulting in the creation of heterogeneous data. The first big challenge is how to ingest heterogeneous data.
2. Multi-source data. Apart from real-time streaming data, there are also legacy applications managing existing devices in enterprises. It is unrealistic to expect enterprises to stop using such applications completely. Moreover, we can analyze real-time streaming data in real-time. As shown in Figure 1, after real-time analysis and original application, device data may be

streaming data, file data or data in relational databases. How to ingest multi-source data is the second big challenge.

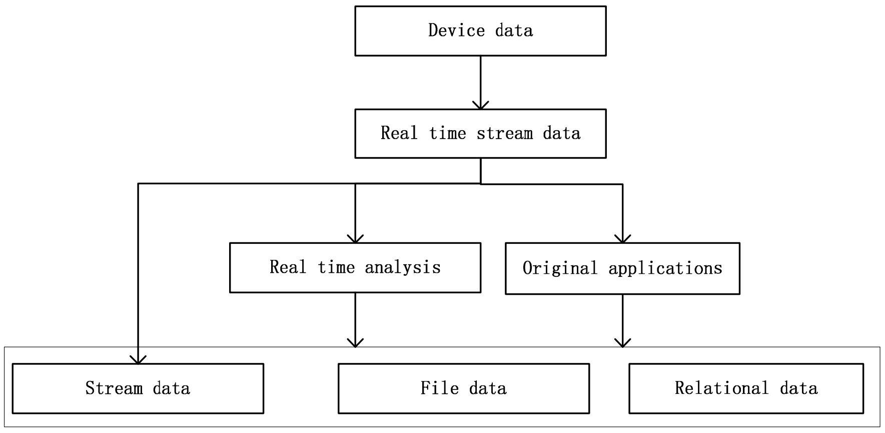


Figure 1. Multi-source data used as device data.

3. Big data. With the development of smart manufacturing technology, it can be foreseen that the IoT will increase the scale of data to an unprecedented level [3]. Recently, we have witnessed explosive growth in the variety, velocity, and volume of data [4]. The era of big data has arrived [5–8]. More effective approaches for resolving record storage and queries in a big data environment are required.

To solve the above issues, a heterogeneous device data ingestion model is urgently needed. Unfortunately, existing models do not properly address these issues. Therefore, we propose a heterogeneous device data ingestion model for our Industrial Big Data Platform (IBDP). IBDP is an industrial big data platform based on a series of open source softwares for ingestion, analysis and visualization of multi-source data [9]. The model includes device templates and four strategies for data synchronization, data slicing, data splitting and data indexing, respectively. We can ingest device data from multiple sources using this heterogeneous device data ingestion model, which is verified on our IBDP. In addition, we report on a case study for a device data-based scenario analysis on the IBDP. The main contributions of our paper are the following:

1. We propose a heterogeneous device data ingestion model, which facilitates the ingestion and fusion of heterogeneous data from multiple sources.
2. We provide four data processing strategies for data synchronization, data slicing, data splitting and data indexing, respectively.
3. We implement the model on our IBDP and present a case study of data analysis after data ingestion.

The rest of the paper is structured as follows: first, a brief overview of related work is given in Section 2. The heterogeneous device data ingestion model is proposed in Section 3. Thereafter, heterogeneous sensor data ingestion methods for multiple data sources are explored in Section 4. In Section 5, the entire platform and some case studies are discussed, and finally, our conclusions are given in Section 6.

2. Related Work

In recent years, we've been witnessing explosive growth in the variety, speed, and volume of data [4]. The most fundamental challenge for the big data platform is how to explore the large volumes and analyze the data to get the useful information or knowledge for future actions [10]. The traditional data platforms are not sufficient in dealing with these challenges. Hadoop is now widely used due to its cost effective, high scalability and fault-tolerance, and now it has become the basic technology for big data [11]. Currently, many big data platform-based HDFS have been developed such as CDH, HDP, Hive, Cascade, *etc.* In our platform, HDFS is also used to store data. Liu *et al.* have summarized some of the big data platforms which have been widely used for achieving real-time availability [11]. These platforms are Hadoop Online, Storm, Flume, Spark and Spark Streaming, Kafka, Scribe, S4, HStreaming, Impala, *etc.* They are leveraged in one or more situations. They are suitable for different situations so the best effect can be achieved by integrating them.

Much research on sensor data and smart data ingestion and fusion has already been reported. Llinas *et al.* provided a tutorial on data ingestion and a basis for sensor ingestion and fusion for further study and research [12]. Regarding sensor data ingestion, various researchers have proposed frameworks; for example, Lee *et al.* presented a peer-to-peer collaboration framework for multi-sensor data fusion in resource-rich radar networks [13]. In most of these frameworks, data can be exchanged among different sensors. This is different from simple sensors, where data cannot be exchanged among the different devices. Dolui *et al.* discussed two types of sensor data processing architectures, namely, on-device and on-server data processing architectures [14]. Smart devices and products in the industrial field employ the second architecture.

For unstructured data, Sawant *et al.* summarized the common data ingestion and streaming patterns, namely, the multi-source extractor pattern, protocol converter pattern, multi-destination pattern, just-in-time transformation pattern, and real-time streaming pattern [15]. At LinkedIn, Lin Qiao *et al.* proposed the far more general and extensible Gobblin, which enables an organization to use a single framework for different types of data ingestion [16]. The structure of the data they collected is unknown, but for smart devices, the structure can be obtained if we have templates for the devices.

There are also a few specialized open-source tools for data ingestion, such as Apache Flume, Aegisthus, Morphlines, and so on, but these are generally used to ingest a single type of data. For heterogeneous device data from multiple sources, we need to ingest different types of data. Thus, we created the IBDP with a heterogeneous device data ingestion model for data from multiple sources. Using this model, we can ingest various device data and store them in a unified format.

This paper is a substantial extension of [9] in some important aspects. First, we propose a heterogeneous device data ingestion model, which facilitates the ingestion and fusing of heterogeneous data from multiple sources. Second, we provide four data processing strategies for data synchronization, data slicing, data splitting and data indexing, respectively. Third, we re-implemented the ingestion layer of IBDP which proposed in [9] with the heterogeneous device data ingestion model and the data processing strategies. Last, we provide more case study details.

3. Heterogeneous Device Data Ingestion Model

Device data include not only streaming data, but also data stored in relational databases and files. We propose a heterogeneous device data ingestion model as outlined in Figure 2. The model can receive or extract heterogeneous device from multiple sources and save them in a unified format. Included in our heterogeneous device data ingestion model are device templates and four strategies based on the device templates. The strategies cover data synchronization, data slicing, data splitting and data indexing.

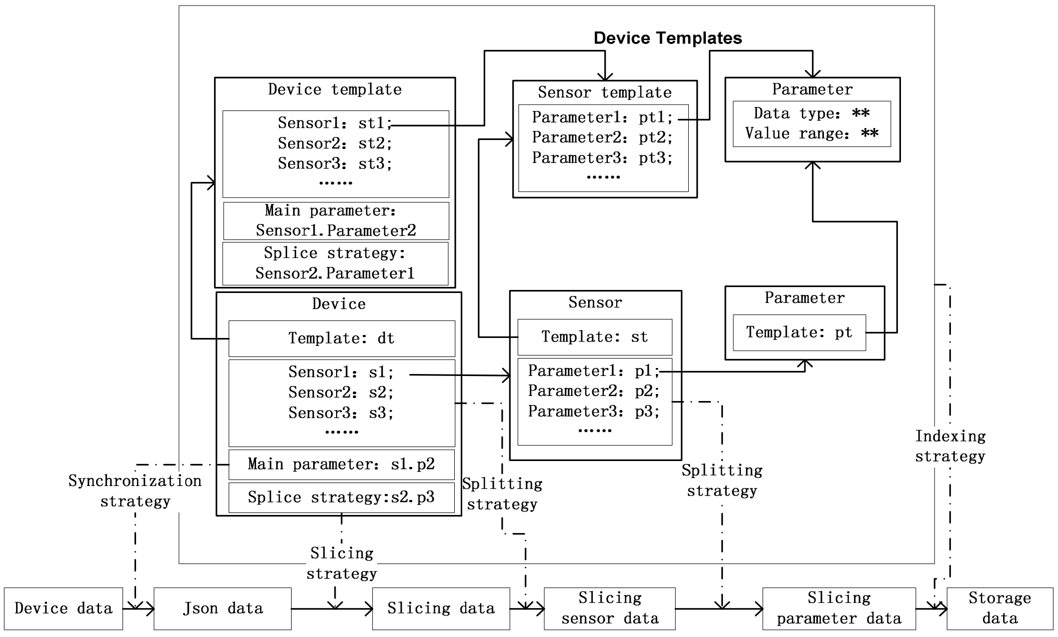


Figure 2. Heterogeneous device data ingestion model.

3.1. *Device Templates*

Each device has sensors and each sensor has parameters. Since for a single type of device the sensors and parameters are the same, we can manage each device with templates. As shown in Figure 2, there are several sensor templates in each device template and there are several parameter templates in each sensor template. For each device, sensor, or parameter template, there may be several corresponding devices, sensors, or parameters, respectively. A device in different templates may contain the same sensor, while a sensor in different sensor templates may contain the same template parameter. In device templates, we need to set the main parameter, which is used for data synchronization. A splice strategy is also needed. Since devices may be logical, we can combine some related sensors to create a virtual device, which is also supported by the device templates.

3.2. *Data Synchronization Strategy*

Different sensor data in a device may be asynchronously transferred, especially in a virtual device. Besides, data stored in a relational database or files need to be merged, if they have been split on the basis of parameter or sensor. Thus, device data need to be synchronized. We propose a data synchronization strategy based on the main parameter in a device template. The synchronization strategy is given in Algorithm 1.

Algorithm 1 Data Synchronization Strategy

```
1.  Receive data.
2.  If the data are asynchronously transferred
3.      If the data contain the main parameter
4.          Create a new JSONObject.
5.          Store the data in the JSONObject according to device template.
6.      Else
7.          Store the data in the JSONObject, whose time is the closest to the time of the data.
8.      End If.
9.  Else
10.     Transform data into JSONObject according to device template.
11. End If
12. Output JSONObjects once a certain number of these objects have been created.
```

The main functions of the data synchronization strategy are data synchronization and data format conversion. To use the data synchronization strategy, we need to set the main parameter when creating the device template. When a device is added to the platform, the main parameter is automatically generated according to the templates.

For data format conversion, we need to set the data mapping relationship from the data received to the device format. For example, the format of data stored in relational databases is {*deviceId*, *sensorId*, *time*, *p1*, *p2*, *p3*}. In the process of data ingestion, each parameter value in the data item is placed in the corresponding position of the JSONObject. The format of a JSONObject is shown in Table 1.

Table 1. Format of JSONObject.

```
{
“deviceId”: “A000000100000001”;
“deviceTemplateId”: “A0000001”;
“time”: 2016-01-31 18:30:06.0;
“sensorData”:[
{
  “sensorId”: “B000000000000001”;
  “parameterData”:[
    { “parameterId”: “C000000000000001”; “value”: 10.3},
    { “parameterId”: “C000000000000002”; “value”: 21.4},
    ....
  ]
},
{
  “sensorId”: “B000000000000002”;
  “parameterData”:[
   { “parameterId”: “C000000000000003”; “value”: 14.2},
   { “parameterId”: “C000000000000004”; “value”: 12.2},
   ....
  ]
},
 ...
]
}
```

3.3. Data Slicing Strategy

Since the device data are continuously updated, they should not be placed in a file. We propose a data slicing strategy for the device templates. There are three slicing methods in this strategy, the first of which is timing slicing. This method requires setting a string in the device templates, the format of which and the included parameters are given in Table 2.

Table 2. Format of string and included parameters for timing slicing.

Parameter	Description
String format	String containing five integer parameters, m h d M dw, in order, for example, 0 4 −1 −1 −1.
m	Minute, ranging from −1 to 59 (−1 means that minute has not been set)
h	Hour, ranging from −1 to 23 (−1 means that hour has not been set)
d	Day of month, ranging from −1,1 to 31 (−1 means that day of month has not been set)
M	Month, ranging from −1,1 to 12 (−1 means that month has not been set)
dw	Day of week, ranging from −1 to 6 (−1 means that day of week has not been set; Sunday = 0, Monday = 1, ..., Saturday = 6)

For example, the string *0 4 −1 −1 −1* means that the data need to be sliced at 4:00 am every day. The second method is periodic slicing, which requires setting an integer in the device templates, denoting the length in seconds of the slicing interval. The third method involves slicing according to a specific parameter, which needs to be set in the device templates. When the parameter changes, the data are sliced. For instance, some devices need to slice data based on the value of a switch sensor. The data slicing strategy is set in the device template. When a device is added to the IBDP, it is automatically equipped with a slicing strategy. The algorithm for the data slicing strategy is given below (Algorithm 2).

Algorithm 2 Data Slicing Strategy

```
1.  Obtain the data slicing strategy by deviceId in JSONObject.
2.  If slicing strategy is timing slicing,
        create a timer.
        When the timer reaches the specified time, create a new file and transmit the old file;
        Add data to the new file when the timer is sleeping.
      End If.
3.  If slicing strategy is periodic slicing,
        create a timer.
        At periodic intervals according to the timer, create a new file and transmit the old file;
        add data to the new file when the timer is sleeping.
      End If.
4.  If slicing strategy is slicing according to specific parameter,
        When the value of the specific parameter changes, create a new file and transmit the old file;
        add data to the new file.
      End If.
```

3.4. Data Splitting Strategy

In the slicing device sensor file, each data item is a JSONObject. Since the device data are heterogeneous, it is difficult to get specific parameters to analyze despite the fact that the parameters can only contain integer, float, Boolean or binary data. The device data could be used and analyzed more easily if they were split into parameters. We propose a splitting strategy based on the templates. Splitting involves two steps, the first of which is to split the file into sensor data items. Each data item in the sensor data files is listed in Table 3. The second step is to split the sensor data files into parameter data files. Each data item in the parameter data file is listed in Table 4.

Table 3. Format of data items in sensor data files.

```
{
"deviceId": "A000000100000001";
"deviceTemplateId": "A0000001";
"time": 2016-01-31 18:30:06.0;
"sensorId": "B000000000000001";
"parameterData": [
    { "parameterId": "C000000000000001"; "value": 10.3},
    { "parameterId": "C000000000000002"; "value": 21.4},
    ....
]
}
```

Table 4. Format of data items in parameter data files.

```
{
"deviceId": "A000000100000001";"deviceTemplateId": "A0000001";
"time": 2016-01-31 18:30:06.0;"sensorId": "B000000000000001";
"parameterId": "C000000000000001";  "value": 10.3
}
```

3.5. Data Indexing Strategy

To speed up data ingestion, we propose a data indexing strategy for data saved in the Hadoop Distributed File System (hereafter, HDFS). In each file, the format of a data item is $<t, v>$, where t is the time of data generation and v is the value of the datum. When the files are created, they are indexed using the data indexing strategy described in Figure 3, where *dtID* is the unique identity of the device template, *dID* is the unique identity of the device, *sID* is the unique identity of the sensor, *pID* is the unique identity of the parameter, and t_{begin} is the start time in the file. *dtID*, *dID*, *sID*, *pID*, and t_{begin} can be obtained from the parameter data files. To save space, we use the directories of HDFS to implement the index structure. Now, we give an example to show how to use the index. Suppose that we want to get the data of one parameter whose identity is pID_1 from t_{begin} (t_{begin} is earlier than t_{begin2} and is later than t_{begin1}) to t_{end} (${}_{end}$ is earlier than t_{begin3} and is later than t_{begin2}) which is shown in Figure 3. First, we search the identities of the sensor and device which contain the parameter pID_1. The identities are sID_1 and dID_1, respectively. Second, we get the device template $dtID_1$ corresponding to device dID_1. Third, we use directory */the_root_directory_to_store_device_data/stID*$_1$*/dID*$_1$*/sID*$_1$*/pID*$_1$ to get data of parameter pID_1. Next, we get the files $file_1$ and $file_2$ (Note that the files not only contain data items between t_{begin} and t_{end}, but also contain some data items earlier than t_{begin} or later than t_{end}). Finally, we read data items from the files, align the data items $< t,v >$ by t and eliminate the data item which t is earlier than t_{begin} or later than t_{end}. After getting the data items, we can analyze the data of Section 4.4.

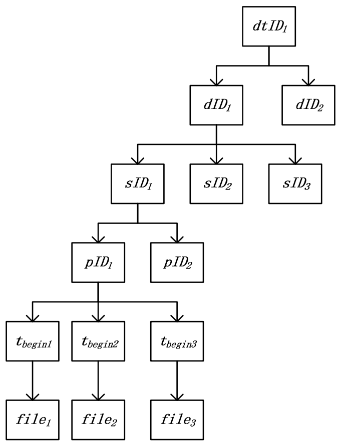


Figure 3. Hierarchical index of device data: *dtID* is the identity of device template, *dID* is the unique identity of device, *sID* is the unique identity of sensor, *pID* is the unique identity of parameter, and t_{begin} is the start time in the file.

4. Heterogeneous Sensor Data Ingestion Methods

The heterogeneous device data ingestion model includes five processes to ingest device data as shown in Figure 4. These processes are data reception or extraction, data synchronization and format transformation, data slicing, data splitting, and data indexing. The data synchronization and format transformation process corresponds to the data synchronization strategy, the data slicing process to the data slicing strategy, the data splitting process to the data splitting strategy, and the data indexing process to the data indexing strategy. The four processes for different types of data sources are the same. The data reception or extraction process differs for different types of data sources.

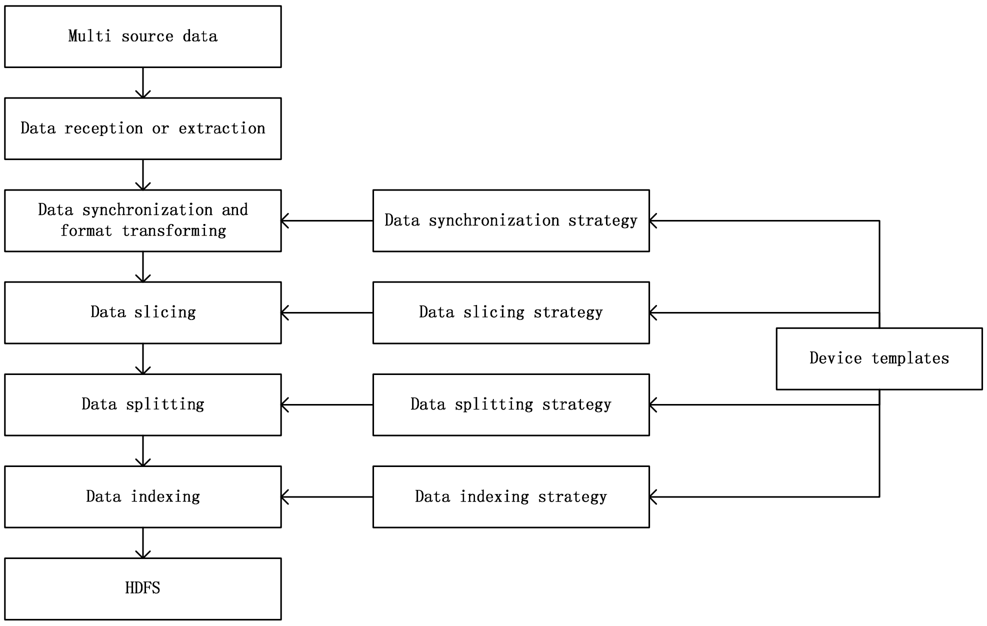


Figure 4. Five processes used to ingest device data.

4.1. Ingestion of Device Streaming Data

Most device data comprise streaming data, which is the basic form to ingest. We do not need to extract streaming data; instead, they are received through Flume, which is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data. The ingestion algorithm for streaming data is given in Alogrithm 3.

Algorithm 3 Streaming data ingestion algorithm

```
1. Receive data from the monitoring port of Flume.
2. Receive data and convert data to a JSONObject using data synchronization strategy (described in Section 3.2,
   Algorithm 1).
3. Slice the data using data slicing strategy (described in Section 3.3, Algorithm 2).
4. Split device data files into parameter data files using data splitting strategy (described in Section 3.3).
5. Index and store files using data indexing strategy (described in Section 3.4).
```

4.2. Ingestion Files of Device Data

Some device data are saved in the form of files after real-time analysis or original application processing. If so, we first obtain the files through Flume and process these into pseudo streaming data. Thereafter, we can view and handle the data as streaming data. The ingestion algorithm for files is given in Algorithm 4.

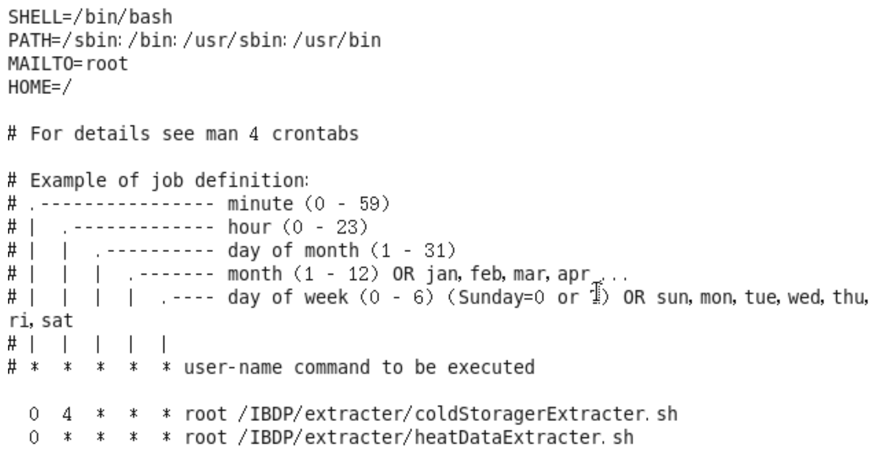
Algorithm 4 Files data ingestion algorithm

```
1. Receive files from monitoring directories of Flume.
2. Read the data and send the data to the monitoring port of Flume.
3. Process the pseudo streaming data using streaming data ingestion algorithm (described in Section 4.1,
   Alogrithm 3).
```

4.3. Ingestion of Device Data from Relational Database

Some device data are stored in relational databases after real-time analysis or original application processing. To ingest the data, we first need to extract them from the relational database using JDBC and Crontab.

The extractor reads data from a relational database through JDBC and sends every line of data to the monitoring port of Flume. When to start the extraction is controlled by Crontab. Crontab, the configuration file for which is shown in Figure 5, is a command provided by the operating system. When we create a timing job, a new line is added to the end of the file. Each line in Crontab, the description of which is shown towards the middle of Figure 5, is a command to start an extractor. Two command examples are given at the bottom of Figure 5. The first command, *0 4 * * * root /IBDP/extracter/coldStoragerExtracter.sh* means that the system will start an extractor as described in file *coldStoragerExtracter.sh* at 4:00 am.



```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,
ri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed

  0  4  *  *  * root /IBDP/extracter/coldStoragerExtracter.sh
  0  *  *  *  * root /IBDP/extracter/heatDataExtracter.sh
```

Figure 5. Configuration file for Crontab.

The data after ingestion by the extractor take the form of pseudo streaming data, which can be viewed and handled as streaming data. The ingestion algorithm for relational data is given in Algorithm 5.

Algorithm 5 Relational data ingestion algorithm

```
1. Start up an extractor using Crontab.
2. Read the data in relational database through JDBC and send the data to the monitoring port of Flume.
3. Process the pseudo streaming data using streaming data ingestion algorithm (described in Section 4.1,
   Alogrithm 3).
```

4.4. Analysis of Device Data

The device data are stored in a unified format after ingestion. However, in most cases, at least one item of the device data is meaningless. Recent empirical evidence strongly suggests that we should analyze the data as a time series. Moreover, the device data need to be fused with business data. In most cases, we analyze the data according to the process outlined in Figure 6.

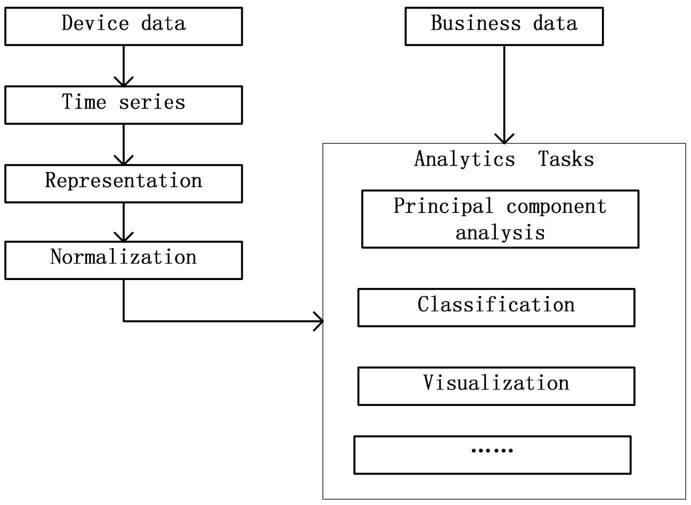


Figure 6. Data analysis process.

Though the device data are in a unified format after ingestion, there are still some difficulties in analyzing them, for example, the data frequency may not be consistent. Time series with different time intervals should not be compared directly. To support data for enterprise analysis, we provide a library of algorithms on IBDP, containing various common algorithms for time series including:

(1) Representation algorithms. In IBDP, there are several time series representation algorithms, such as sampling, piecewise aggregate approximation [17], discrete Fourier transforms [18], discrete wavelet transforms [19], piecewise linear representation [20] and piecewise linear representation based on series importance point [21]. After choosing a representation, the time series can be compared with each other.
(2) Normalization algorithms. The time series obtained are generally accompanied by much noise. The IBDP contains the min-max and Z-score normalization algorithms. After normalization, the impact of noise is greatly reduced.
(3) Principal component analysis algorithms. We do not need to analyze all the parameters of each type of device; instead, only key parameters are extracted for analysis. The IBDP includes a principal component analysis [22] algorithm to analyze the principal components of a time series.
(4) Visualization algorithms. To better display the features to enterprises, there are line chart [23] and ThemeRiver [24] algorithms for time series visualization. Using these visualization algorithms, enterprises can see the features directly.
(5) Classification algorithms. The IBDP includes some classification (as well as early classification) algorithms. The main classification algorithms are shapelets (perfectly accurate shapelets from [25], and fast shapelets from [26]), 1NN, and early classification of time series [27]. Using the classification algorithms, enterprises can classify time series for different analytical tasks.

In the analytical tasks, we use not only device data, but also business data. By fusing the data, the analytical result is more accurate and more useful.

5. Case Study of Industrial Data Analysis

5.1. IBDP: An Industrial Big Data Platform

The high-level architecture of a big data platform consists of three main layers: the data ingestion layer, data analytic layer, and data storage layer [4]. We implemented the heterogeneous device data ingestion model and methods in the IBDP [9], the architecture of which is shown in Figure 7.

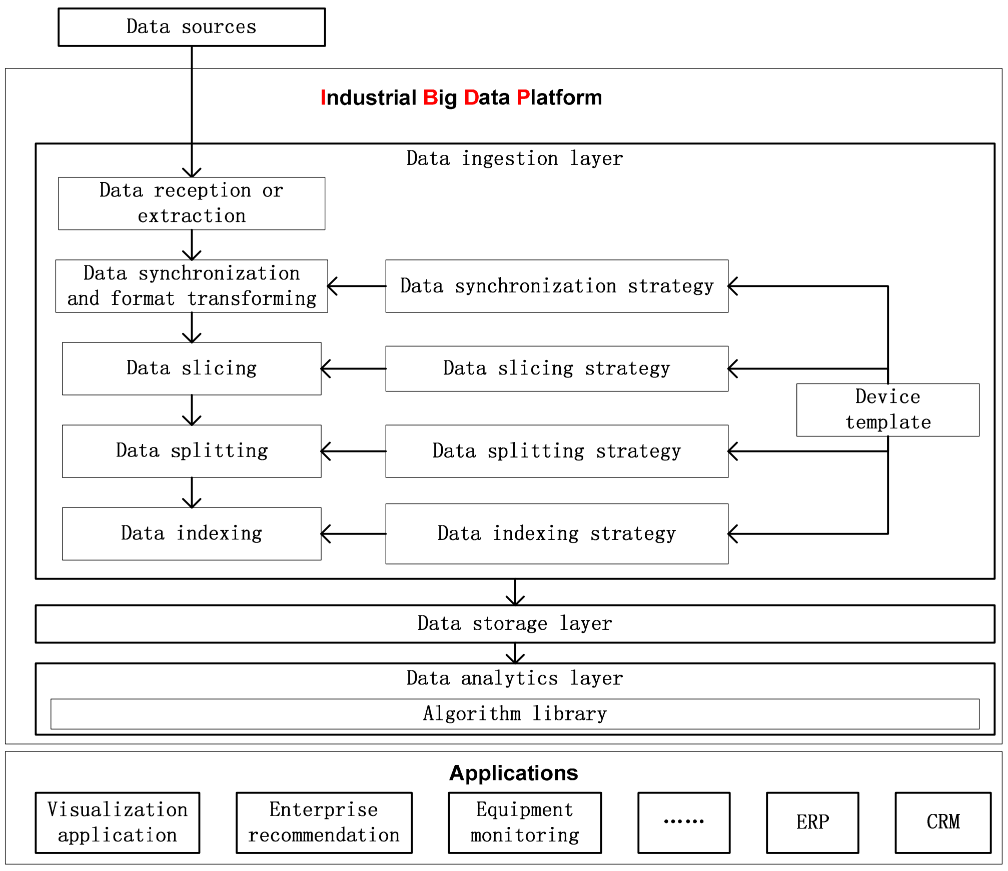


Figure 7. Architecture of IBDP.

Data are saved in the data storage layer and analyzed in the data analytic layer. There are a series of applications based on the IBDP and supported by analysis results.

The platform and applications are deployed on the cloud resource center, which allows us to adjust the storage capacity and computation ability at any time according to the need [28–30]. The extractors, Flume agents, and other processes for data ingestion can be extended, as can the storage capacity of the HDFS, and the computing ability of Spark and MapReduce.

We used our platform to ingest data from the Longda Foodstuff Group Co., Ltd. (Jinan, China, hereafter refrerred to as Longda,) [31] and the Jinan District Heating Limited Company (Jinan, China hereafter JDH,) [32]. In Longda, there are nearly 200 production devices, from which we obtained the operating data. We also created 22 virtual devices for cold storage. JDH monitors the heating in 14,237 apartments and we created one virtual device for all the apartments. All the virtual heating monitoring devices point to the same template. The devices are summarized in Table 5.

Table 5. Devices currently ingested by IBDP.

Device	Number	Type of Data	Company
production devices	almost 200	streaming data	Longda
virtual cold storage devices	22	file	Longda
virtual heating monitoring devices	14,237	relational database	JDH

In the near future, we will also ingest production device data from Zhongtong Bus Holding Co., Ltd. (Jinan, China, hereafter, ZTB) [33] as well as the operational data of buses they manufacture. We will also ingest heating data from Jinan Theral Power Co., Ltd. (Jinan, China, hereafter JTP) [34]. In addition, we would like to use the platform for other companies' data.

5.2. Case Study Analyzing Temperature Sensor Data

In Longda, products are kept in several cold storage rooms, each of which has 20 refrigeration units to maintain the temperature. Each refrigeration unit has a sensor, which is used to monitor the temperature in the cold storage (T) room and electric power consumption (W). We use these sensors in the cold storage room to construct a virtual smart device as shown in Figure 8. Note that only Sensor 1 (S1) is shown as sensors S1 to S20 are all the same.

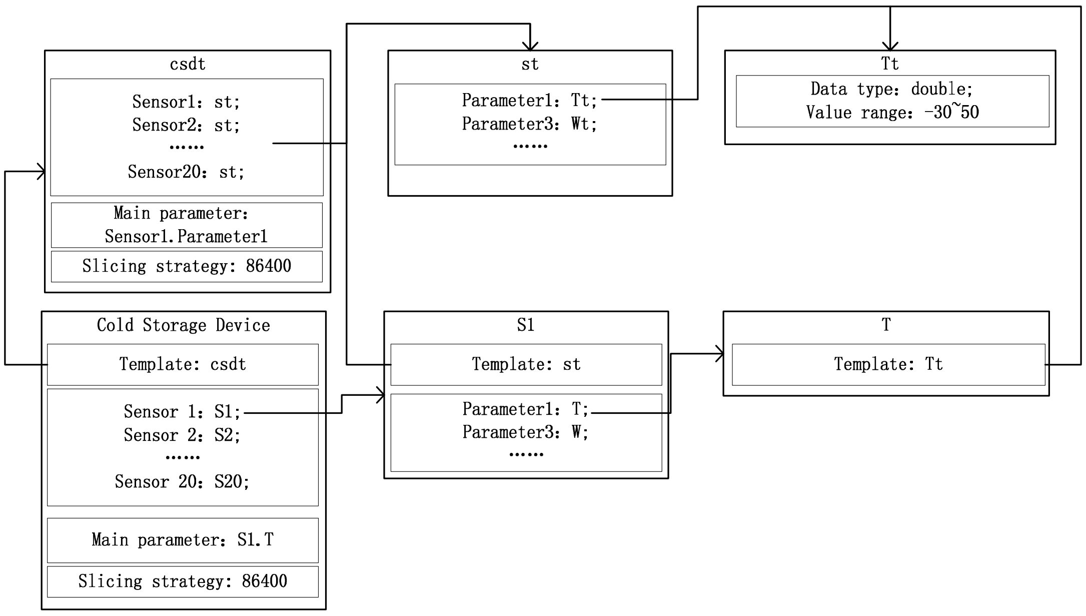


Figure 8. Virtual cold storage device.

In each cold storage room, the main parameter is T in Sensor 1 and the slicing strategy is 86,400, which means we slice data every 86,400 s (once a day).

After the data had been stored, 2500 data items of T were obtained by indexing. We used a threshold detection algorithm for detection because this would affect the quality of the products when the temperature was too high or too low. The high threshold is 0 and the low threshold is −30 as the two horizontal lines in Figure 9. The anomaly detection results are shown in Figure 9, where the points below the horizontal line of −30 are abnormal owing to too low temperatures.

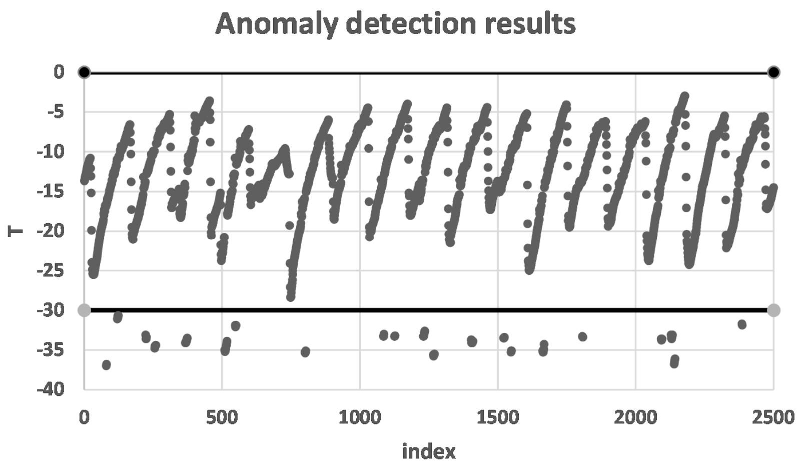


Figure 9. Anomaly detection results.

Thus, we obtained T parameters, which were affected by the temperature of the cold storage and the working status of the refrigeration units. Then, we analyzed the correlation between them using a near correlation coefficient. The correlation analysis results are shown in Figure 10.

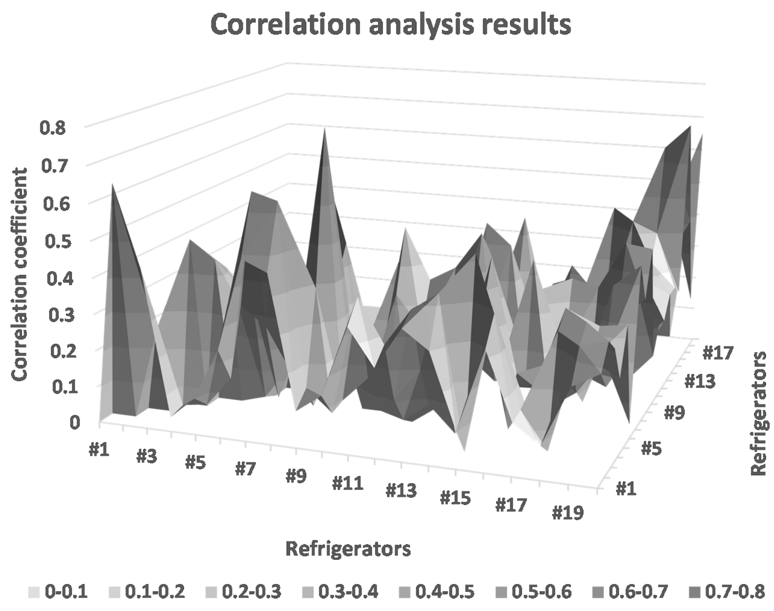


Figure 10. Correlation analysis results of some cold storage rooms.

From Figure 10, we can see the working status of the refrigeration unit affecting the value of T the most. The working status of refrigerators #1 and #2, #3 and #4, . . . , #19 and #20 is similar.

The price of electricity differs at different times. The price at night is much lower than that during the day. The prices are shown in Table 6. To reduce cost, we expect that cooling happens between 23:00 and 07:00.

Table 6. Price of electricity at different times.

Time	Price
10:30–11:30, 19:00–21:00	1.2773
08:30–10:30, 18:00–19:00, 21:00–23:00	1.2068
07:00–08:30, 11:30–18:00	0.7838
23:00–07:00	0.3608

We analyzed the relation between electric power consumption and temperature. When the temperature is between 0 and −30, the fitting functions are as shown in Table 7.

Table 7. Fitting functions when the door of the cold storage is closed.

Status	Temperature (T) and Time (t)	Power (W) and Time (t)
No refrigeration and door of cold storage is closed	$T' = T + 1.5 \times t$	$W' = W + 11 \times t$
No refrigeration and door of cold storage is open	$T' = T + 5.4 \times t$	$W' = W + 11 \times t$
Refrigeration and door of cold storage is closed	$T' = T - 10.95 \times t$	$W' = W+3225 \times t$

Finally, we formulated the strategy for refrigeration, which begins at 23:00 and lasts for $3 + (t_{open})/2$ h, where t_{open} is the total open time.

6. Conclusions

With the development of digitized manufacturing, data ingestion and analysis have become more and more important. In this paper, we presented a data ingestion model for heterogeneous devices, which consists of device templates and four strategies for data synchronization, data slicing, data splitting, and data indexing, respectively. Next, we introduced heterogeneous sensor data ingestion methods to ingest device data from multiple sources. Finally, we presented a case study describing a scenario in which device data were analyzed on the IBDP. We showed that the heterogeneous device data ingestion model can be used to ingest, analyze, and store data. It also makes it easier and more efficient for enterprises to analyze data.

Acknowledgments: The authors would like to acknowledge the support provided by the National Natural Science Foundation of China (61402263), the National High Technology Research and Development Program of China (2014AA01A302), special funds from the Taishan Scholar Construction Project, the Independent Innovation Projects of Shandong Province (2013CXC30209, 2014ZZCX08102), the Science & Technology Development Projects of Shandong Province (2014GGX101028), and the Open Fund of Beijing Key Laboratory on Integration and Analysis of Large-scale Stream Data, North China University of Technology.

Author Contributions: The work was conducted with the cooperation of all authors. The main idea was proposed by Cun Ji, Shijun Liu and Chenglei Yang. Lei Wu and Li Pan analyzed the data model and provided many valuable suggestions. Cun Ji, Qingshi Shao and Jiao Sun implemented the model and methods. Cun Ji wrote the manuscript. All authors participated in revising the manuscript.

Conflicts of Interest: The authors have no conflicts of interest to declare.

Abbreviations

The following abbreviations are used in this manuscript:

IBDP	Industrial Big Data Platform
IoT	Internet of Things
Longda	Longda Foodstuff Group Co., Ltd.
JDH	Jinan District Heating Co., Ltd.
ZTB	Zhongtong Bus Holding Co., Ltd.
JTP	Jinan Theral Power Co., Ltd.

References

1. Rani, S.; Talwar, R.; Malhotra, J.; Ahmed, S.H.; Sarkar, M.; Song, H. A novel scheme for an energy efficient Internet of Things based on wireless sensor networks. *Sensors* **2015**, *15*, 28603–28626. [CrossRef] [PubMed]
2. Lin, Y.; Yang, J.; Lv, Z.; Wei, W.; Song, H. A self-assessment stereo capture model applicable to the Internet of Things. *Sensors* **2015**, *15*, 20925–20944. [CrossRef] [PubMed]
3. Fan, W.; Bifet, A. Mining big data: Current status, and forecast to the future. *ACM SIGKDD Explor. Newsletter* **2013**, *14*, 1–5. [CrossRef]
4. Ranjan, R. Streaming big data processing in datacenter clouds. *IEEE Cloud Comp.* **2014**, *1*, 78–83. [CrossRef]
5. Campbell, P. Community cleverness required. *Nature* **2008**, *455*. [CrossRef]
6. Mervis, J. Agencies rally to tackle big data. *Science* **2012**, *336*, 22. [CrossRef] [PubMed]
7. Labrinidis, A.; Jagadish, H.V. Challenges and opportunities with big data. In Proceedings of the VLDB Endowment, Istanbul, Turkey, 27–31 August 2012; Volume 5, pp. 2032–2033.
8. Gantz, J.; Reinsel, D. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView: IDC Analyze the Future* **2012**, *2007*, 1–16.
9. Ji, C.; Liu, S.; Yang, C.; Wu, L.; Pan, L. IBDP: An Industrial Big Data Ingestion and Analysis Platform and Case Studies. In Proceedings of the 2015 International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI), Beijing, China, 22–23 October 2015. (in press).
10. Rajaraman, A.; Ullman, J.D. *Mining of Massive Datasets*; Cambridge University Press: Cambridge, UK, 2012.
11. Liu, X.; Iftikhar, N.; Xie, X. Survey of real-time processing systems for big data. In Proceedings of the 18th International Database Engineering & Applications Symposium, Porto, Portugal, 7–9 July 2014; pp. 356–361.

12. Llinas, J.; Hall, D.L. An introduction to multi-sensor data fusion. In Proceedings of the 1998 IEEE International Symposium on Circuits and Systems (ISCAS'98), Monterey, CA, USA, 31 May–3 June 1998; pp. 537–540.
13. Lee, P.; Jayasumana, A.P.; Bandara, H.D.; Lim, S.; Chandrasekar, V. A peer-to-peer collaboration framework for multi-sensor data fusion. *J. Netw. Comput. Appl.* **2012**, *35*, 1052–1066. [CrossRef]
14. Dolui, K.; Mukherjee, S.; Datta, S.K. Smart device sensing architectures and applications. In Proceedings of the 2013 International Computer Science and Engineering Conference (ICSEC), Nakorn Pathom, Thailand, 4–6 September 2013; pp. 91–96.
15. Sawant, N.; Shah, H. Big Data Ingestion and Streaming Patterns. In *Big Data Application Architecture Q & A*; Apress: New York, NY, USA, 2013; pp. 29–42.
16. Qiao, L.; Li, Y.; Takiar, S.; Liu, Z.; Veeramreddy, N.; Tu, M.; Botev, C. Gobblin: Unifying data ingestion for Hadoop. In Proceedings of the 41st International Conference on Very Large Data Bases, Kohala Coast, HI, USA, 31 August–4 September 2015; Volume 8, pp. 1764–1769.
17. Keogh, E.; Chakrabarti, K.; Pazzani, M.; Mehrotra, S. Dimensionality reduction for fast similarity search in large time series databases. *Knowl. Inf. Syst.* **2001**, *3*, 263–286. [CrossRef]
18. Agrawal, R.; Faloutsos, C.; Swami, A. *Efficient Similarity Search in Sequence Databases*; Springer: Berlin, Heidelberg, 1993.
19. Chan, K.P.; Fu, A.W.C. Efficient time series matching by wavelets. In Proceedings of the 15th International Conference on Data Engineering, Sydney, Australia, 23–26 March 1999; pp. 126–133.
20. Keogh, E. Fast similarity search in the presence of longitudinal scaling in time series databases. In Proceedings of the Ninth IEEE International Conference on Tools with Artificial Intelligence, Newport Beach, CA, USA, 3–8 November 1997; pp. 578–584.
21. Zhou, D.Z.; Li, M.Q. Time series segmentation based on series importance point. *Comp. Eng.* **2008**, *34*, 14–16. (In Chinese).
22. Yang, K.; Shahabi, C. On the stationarity of multivariate time series for correlation-based data analysis. In Proceedings of the Fifth IEEE International Conference on Data Mining, Houston, TX, USA, 27–30 November 2005.
23. Playfair, W. *The Commercial and Political Atlas and Statistical Breviary*; Cambridge University Press: Cambridge, UK, 2005.
24. Havre, S.; Hetzler, B.; Nowell, L. ThemeRiver: Visualizing theme changes over time. In Proceedings of the IEEE Symposium on Information Visualization (InfoVis 2000), Salt Lake City, UT, USA, 9–10 October 2000; pp. 115–123.
25. Mueen, A.; Keogh, E.; Young, N. Logical-shapelets: An expressive primitive for time series classification. In Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, 21–24 August 2011; pp. 1154–1162.
26. Rakthanmanon, T.; Keogh, E. Fast shapelets: A scalable algorithm for discovering time series shapelets. In Proceedings of the 13th SIAM Conference on Data Mining (SDM), Austin, TX, USA, 2–4 May 2013.
27. Xing, Z.; Pei, J.; Philip, S.Y. Early classification on time series. *Knowl. Inf. Syst.* **2012**, *31*, 105–127. [CrossRef]
28. Armbrust, M.; Fox, A.; Griffith, R.; Joseph, A.D.; Katz, R.; Konwinski, A.; Zaharia, M. A view of cloud computing. *Comm. ACM* **2010**, *53*, 50–58. [CrossRef]
29. Barroso, L.A.; Clidaras, J.; Hölzle, U. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Syn. Lect. Comp. Archit.* **2013**, *8*, 1–154. [CrossRef]
30. Wang, L.; Ranjan, R.; Chen, J.; Benatallah, B. *Cloud Computing: Methodology, Systems, and Applications*; CRC Press: Boca Raton, FL, USA, 2011.
31. Longda Foodstuff Group Co. Ltd. Available online: http://www.longda.com.cn (accessed on 15 December 2015).
32. Jinan District Heating Co. Ltd. Available online: http://www.jnreli.com (accessed on 15 December 2015).
33. Zhongtong Bus Holding Co. Ltd. Available online: http://www.zhongtong.com/en (accessed on 15 December 2015).
34. Jinan Theral Power Co. Ltd. Available online: http://www.jnrdyxgs.com (accessed on 15 December 2015).