Cluster Computing
https://doi.org/10.1007/s10586-018-2074-6

E-commerce big data computing platform system based on distributed computing logistics information

Junmin Hu[1]

Received: 20 January 2018 / Revised: 3 February 2018 / Accepted: 7 February 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract
E-commerce websites generate a large amount of user behavior data, with the continuous increase of the business volume of e-commerce companies. Enterprises hope to have a deep understanding of each customer through these data and expect to form a learning relationship with customers. Based on this, this paper firstly elaborates the background and significance of the system development, the status quo of related technologies at home and abroad, then carries on the overall design and gives the system realization plan, and finally designs and realizes the big data T station based on distributed real-time computing framework, Enterprises, especially Internet companies, can provide internal data support by building a hierarchical data warehouse infrastructure and real-time data computing technology, and then cluster, divide and predict more relevant data through mathematical statistics and mining algorithms. Each of us is tagged with descriptions to get each of our attributes and hobbies, etc., providing a strong support for the external needs of enterprises and fast data-driven business.

Keywords Distributed computing · E-commerce · Big data · Computing platform

1 Introduction

E-commerce websites generate a large amount of user behavior data, with the continuous increase of the business volume of e-commerce companies. Enterprises hope to have a deep understanding of each customer through these data and expect to form a learning relationship with customers [1]. Businesses can translate customer data into customer knowledge, making mining big data enable computers to learn about customer relationships, improving the overall operational efficiency of e-commerce companies, and improving GMV and other key indicators by taking advantage of the vast amount of information they have and the data they generate each time they interact with their customers [2]. Early data warehousing and recommendation engines used off-line data calculations, with inefficient internal reporting requirements and lagging site recommendation results, affecting user experience and order placement, and were yet to be addressed in real-time data calculations[3]. Users generate more and more data in their daily operations due to the continual escalation of business systems and the constantly increased number of users. Earlier data platforms are not enough to support more and more massive amounts of data due to their simple architecture [4]. At the same time, the well-constructed data system is also a huge systematic project. The entire construction period is very long and involves a long chain of ecological resources including: data collection and access, cleaning, storage computing, data mining, visualization, etc. Each link can all be built as a complex system. All the data are put together. Decoupling is very risky for future system integration [5].This paper studies e-commerce big data computing platform system based on distributed computing based on this.

2 State of the art

Many scholars have done corresponding research in the study of massive network data mining. Some scholars put forward let the massive data and mining tasks be

✉ Junmin Hu
WilhelminaaDbV@yahoo.com

[1] Party School of Guangxi District Organs of C. P. C, Guangxi, Nanning, China

Published online: 23 February 2018

decomposed into multiple servers and parallel processing to improve the efficiency of network data mining. They also proposed to use the method of database simulating linked list structure to solve large-scale tree structure and graph model of distributed computing, and realize the efficiency of data mining in Hadoop. Some scholars also proposed to apply Map Reduce idea to Naive Bayesian Classification Algorithm, K-modes Clustering Algorithm and ECLAT Frequent Item Set Mining Algorithm, and the effectiveness of this application for data mining was verified by experiments They applied the Map Reduce programming model to Ant Colony Algorithm and realized the rapid mining of web logs. These researches provide the theoretical basis for big data mining in e-commerce platform [6].There were many scholars have conducted research as for distributed computing, which is two or more software to share information with each other. These software can run on the same computer or on several different computers connected by a network [7].The vast majority of distributed computing is based on client/server models and there are two kind of main software: client software that makes requests for information or services; and server software that provides such information or services.

3 Methodology

3.1 Design and implementation of data warehouse system

Data warehouse technology: there are often a large amount of data tables in daily operations of enterprise, in which attribute fields are about 50 and the data size is more than 100 million records, for which the traditional relational database can not support, and there are a lot of good products with big data gambling and computing in the field of engineering big data technology, among which the most influential one is Hadoop, which provides a set of solution of big data storage and computing, solving the issues of storage and computing of big data perfectly. Early data warehouse used a single mysql, which quickly reached the bottleneck in the rapid development of business. So migrating from mysql to Oracle and upgrading to minicomputer high-end storage can also satisfy most of the needs. But the data volume is bigger and bigger, in addition to the needs of real-time, accuracy, search engine, etc., the long-open-sourcing Hive was elected through research. ApacheHive is a data warehouse built on top of the Hadoop architecture. When redacted hql in hive, hive dynamically compiled that into a mapreduce program, batching and calculating data in hadoop. It is a basic technology selection to build our data warehouse which can provide refining and query analysis of data.

In the realization of data warehouse, the process of data cleaning can be simplified through hierarchical management of data. Because dividing the original work into multiple steps is equivalent to dividing a complicated work into simple tasks, the original black box into a white box, and each layer of the processing logic is relatively simple and easy to understand, so that we can easily ensure the accuracy of each step, when the data error occurs, we often only need to adjust a local step. The benefit of doing so is that each layer of data is decoupled and the data flows in a certain order so that the entire data warehouse system is more robust. The figure below shows the hierarchy of the data warehouse (Fig. 1).

3.2 Data knowledge and metadata management

Data knowledge and metadata management is the data of data, that is, the intermediary of data, which is similar to the administrator of the library, which can record management information, business rules and algorithms as well as data features of the warehouse and the market, and provide multi-dimensional metadata browsing and query service. It can also view the implementation of the running tasks and the complete or unsuccessful tasks in the current warehouse, and it can inform the data warehouse to develop engineer by the way of pushing notification or alerting, and three functions are mainly included, respectively are rapid retrieval of metadata information, data kinship map and rich metadata information.

Rapid retrieval of metadata information refers to locating the required metadata information quickly and accurately through the search engine and multi-dimensional labels. These metadata information can help data warehouse developers to accurately locate the desired theme

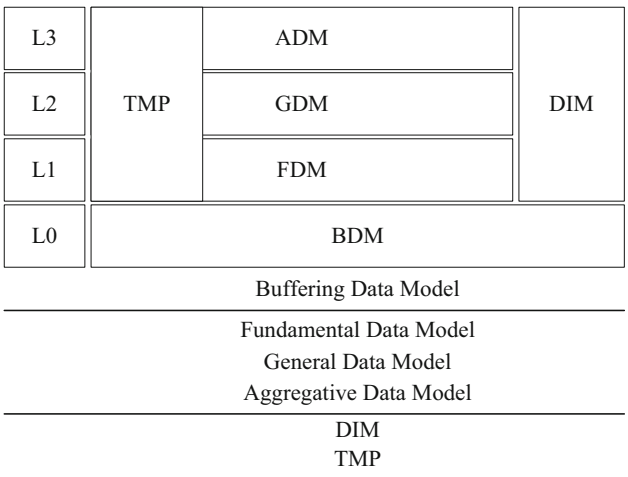

Fig. 1 Hierarchical architecture of data warehouse

data, and the newly added or modified table agencies have history, field and table on-line or off-line have a clear record, which is more conducive to develop Labor Management, preventing data problems and duplication of work caused by negligence or personnel changes; we can trace the source of the data and understand the dynamics of related tasks through the data kinship map; rich metadata information includes big data platform data warehouse and all the metadata information of the data in market, in addition to the basic information, and also includes the all-around and multi-dimensional metadata information such as partition information, loading information, etc. It as well as includes: collections, visit records and comments and other personalized services, and items that need to be paid attention to noted in the comment function, which are not only convenient for newcomers later, but also can enhance the collaborative development efficiency of data warehouse team. The figure below shows the details of the table.

3.3 Realization of user portrait market

In fact, the user portrait is a data mart built in the upper layer of the data warehouse. Setting up user portrait platform can connect the data platform which has a large amount of user data with the visual data tool platform. Let personnel of R&D and production, user research, marketing and so on analyze user characteristics of different products at any time independently according to needs to rapidly insight into users' needs with the application of value of data mining platform according to interaction scenarios of different users. Perfect user portrait platform needs to consider a comprehensive model system, usually speaking, the data required to build user portrait platform is divided into three types of entities, namely are users, product and channels. Each tab provides a perspective on how we observe, recognize and describe the user. User Portrait is a whole, which means that each dimension is not isolated and there is a link between the labels. For each type of data entity, the data dimension of the user portrait further decomposes the data dimension that can fall to form a field set. The figure below shows the data processing flow of the user portrait (Fig. 2).

First, we generate user tags to identify a user's attribute by labeling the user, and to reflect the true characteristics of the user through the user's web site browsing behavior; second, to give the definition of user's life cycle labels, we divided it into several types in accordance with the order situation based on the rules of business summary as for the user's life cycle stage.

In the application of user portrait, the established user portrait fair can be standardized output to the outside and synchronized to a unified data mart layer to facilitate the development of the underlying call; at the same time, setting up data cube with multi-dimensional analysis based on theme can be directly faced analysts and engineers to get through the associated data and products of upstream and downstream and to marketing users, and as for product managers and front-line sales personnel, they can filter out the predetermined population in the data cube and directly call the marketing platform for issuing coupons, EDM and other operations, reducing many intermediate links, achieving efficient operations and precision marketing, so as to greatly enhance the efficiency. Multidimensional analysis is one of the excellent applications of producibility of user portrait among them. The combination of many dimensions of user portrait and orders, goods, traffic and other indicators can quickly realize intelligent analysis and provide professional and effective advice according to data comparison and analysis. We can also establish membership growth system according to the user portrait, which, in fact, is the user's level label. It can meet the individual needs of members and provide better services for members through the membership level, so that the level label has practical application value, that is, the higher the level of membership loyalty and the frequency of consumption are, the higher the membership level will be, and the higher the exclusive rights and services will be.

4 Result analysis and discussion

4.1 Engine component with real-time computing-Infobright

Data analysts have stringent requirements on the timeliness of data queries in the scenario of AD-HOC, hoping to respond to a few urgent requests in a short time and make timely market decisions and marketing feedback. After several rounds of technical selection, we selected Infobright to solve such problem very well, which can achieve excellent performance through knowledge grid and comprehensive optimization technology for complex queries. Once imported into Infobright, the data was highly compressed and stored as "chunks", and at the same time, the knowledge grid automatically created a very compact piece of metadata that contained the relational information between the statistics and the chunks. So when a query was received, Infobright's query optimizer could intelligently decide which blocks of data were related to the query request through metadata and decompressed it. Infobright did not need to specifically divide the data and did not need to create index based on knowledge grid technology so as to save query processing time and improve response speed. Therefore, Infobright was finally selected for an ad hoc query on the identified data mart to meet the fast and

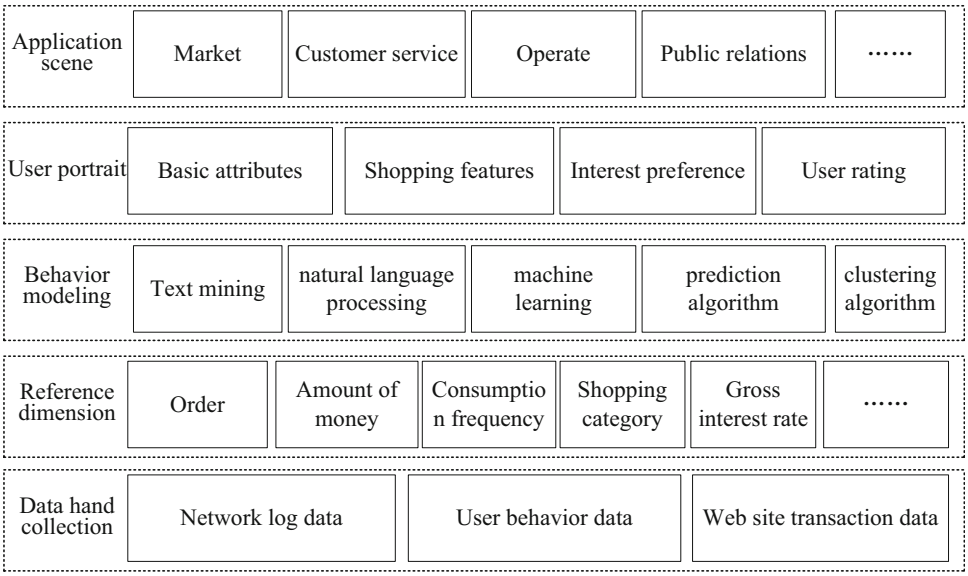

Fig. 2 The overall architecture of user portrait generation

complicated demand for analysts. InfoBright's knowledge grid data architecture is shown below (Fig. 3).

It was a visual interface to interact with users at the front end of data products. We hoped that users can select different reports forms to set up their own reports according to their own needs of measures and dimensions while they were viewing inherent reports they created. They could also create a derived measure based on the measurement provided making a simple operation when there was no suitable measure, while the business people just shared created reports under the department for departmental use through the approval of departmental leaders. In order to meet this need, we needed to make dynamic splicing of sql statements in the front-end, and users in the front-end dynamically generated sql statement and sent to the back-end for data calculations in the form of dragging and dropping (Fig. 4).

A route calculation needed to be performed to determine whether the request could be quickly returned and ad hoc query, or to the underlying data warehouse for curing when required allocation. The routing rule was also summarized in the experiment, which based on the test environment with a single node, 2 physical cpu, each cpu8 core, 2.6GHZ dominant frequency of cpu and 128G internal storage. Due

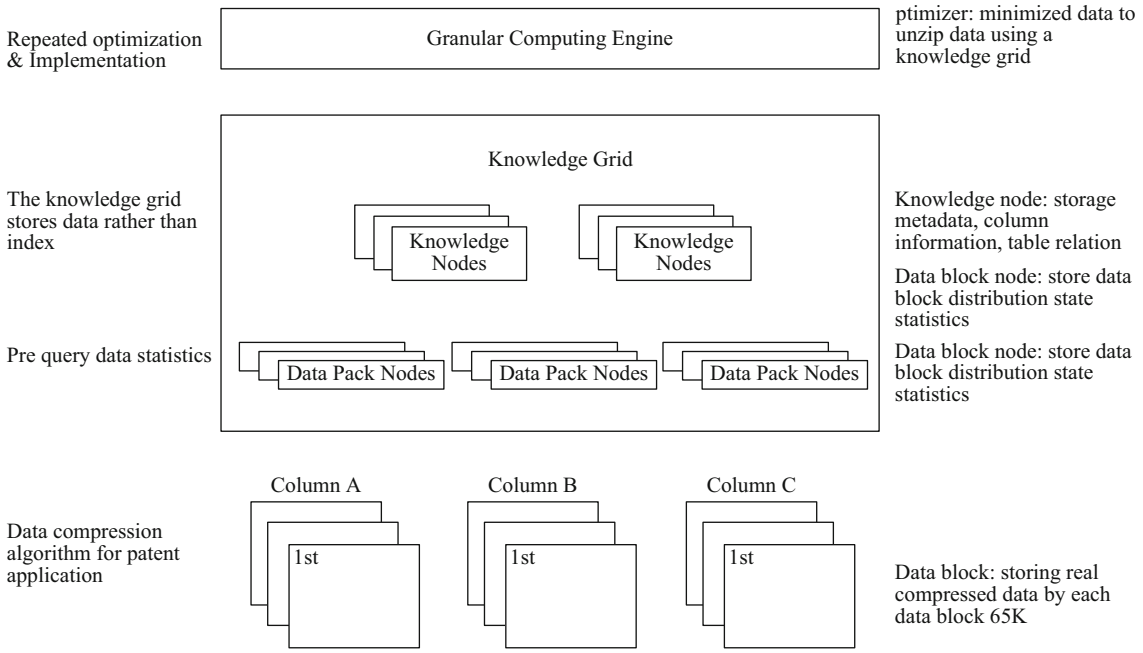

Fig. 3 InfoBright based on the architecture of the knowledge grid

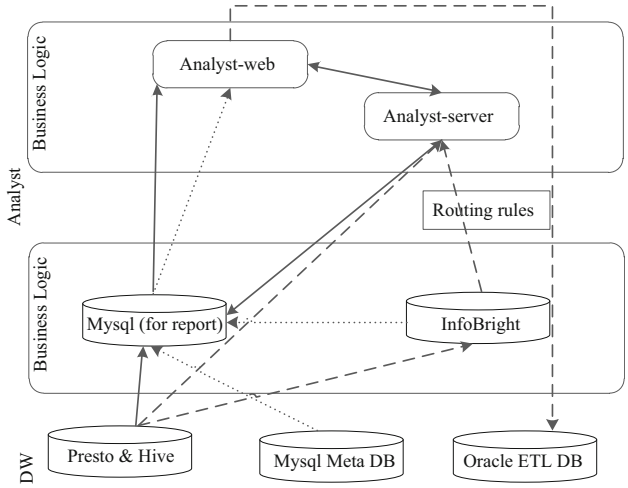

Fig. 4 SQL dynamic splicing and data routing rules architecture

to the production environment was relatively close, this time we imported 140 million data to the following tests (Figs. 5, 6).

We got the routing rules from the experiment, if the number of the join table was more than 2 or 3, could be considered taking Presto&Hive; if there had distinct, you could decide whether to take hive according to the join table data and the data volume of table; if there was orderby, you could consider taking hive; If the amount of data was too large, such as more than 300 million, you could consider taking hive; if there was case when, you could consider Presto&Hive; if there had conditions such as $\geq$, $\leq$ and so on of where condition in the query, you could decide whether or not take Presto&Hive based on the volume data of table. Through the routing rules summarized above, we gave different weights for different situations, realizing the function of dynamic routing of data query through rules and weights. Of course, we also needed to constantly adjust the parameters in production to continuously optimize the query efficiency.

In practice, in order to be able to respond quickly to user requests for computation, we have built a data mart facing more than one themes for column database on the topic of data warehouse technology. When in the interactive interface, the user dynamically added dimensions and indicators through drag-and-drop and submitted the request to the system, which generated a SQL statement dynamically according to the request submitted by the user, and the routing rule would determine which calculation engine the generated SQL statement would sent to.

4.2 Engine component with real-time computing-Preresto

Presto is a distributed query and execution engine fully internal-storage-based, so the hardware choice for the Presto cluster must meet the requirements of large internal storage, 10GbE and high computational power. Presto cluster is the topology of master–slave, because that Presto is divided into two kinds of node services, namely are Coordinator and Worker. In addition, due to Presto has two clients: Cli Client: the server deployed Presto command line client. Application Client: JDBC driver which developers can use Presto to query and calculate big data using Java code. Therefore, the hardware architecture of the entire Presto cluster includes four kinds of servers: Coordinator, Worker, Cli client and application client. The realization of hardware system was shown in the figure (Fig. 7).

Presto's support for multiple data sources, data source decoupling, and ease of scalability are attributed to Presto's Connector design. Each data source corresponds to a Connector in Presto, you can discretionarily develop your own desired data source. Presto is a distributed big data query and execution engine fully internal-storage-based, with all queries and calculations executing in internal storage. Based on PiPeLine(pipeline)design, each query

```
mysql> select s1.index_value_g1,s1.index_value_g2,s1.index_value_g3
    -> from
    -> (
    -> select
    -> t3.month_of_quarter
    -> ,sum(t1.order_count) as index_value_g1
    -> ,sum(t1.order_amt) as index_value_g2
    -> ,count(distinct t1.opp_customer_no) as index_value_g3
    -> from test.a05_cust_trade_stat t1
    -> left join test.dim_cfg_date t3
    -> on t1.event_create_dt=t3.calendar_date
    -> left join test.dim_cfg_biztype t4
    -> ON t1.biz_type = t4.biztype_cd
    -> where t3.year_of_calendar='2015' and t3.month_of_quarter ='1' and t4.biztype_cd ='01'
    -> group by t3.month_of_quarter
    -> ) s1 ;
+----------------+-----------------+----------------+
| index_value_g1 | index_value_g2  | index_value_g3 |
+----------------+-----------------+----------------+
|        6060183 | 37165139810.8316 |         780478 |
+----------------+-----------------+----------------+
1 row in set (8.89 sec)
```

Fig. 5 Single table query 3 indexes take 8.9 s

```
mysql> select s1.index_value_g1,s1.index_value_g2,s1.index_value_g3,
    -> s2.index_value_g4,s2.index_value_g5,s2.index_value_g6
    -> from
    -> (
    -> select
    -> t3.month_of_quarter
    -> ,sum(t1.order_count) as index_value_g1
    -> ,sum(t1.order_amt) as index_value_g2
    -> ,count(distinct t1.opp_customer_no) as index_value_g3
    -> from test.a05_cust_trade_stat t1
    -> left join test.dim_cfg_date t3
    -> on t1.event_create_dt=t3.calendar_date
    -> left join test.dim_cfg_biztype t4
    -> ON t1.biz_type = t4.biztype_cd
    -> where t3.year_of_calendar='2015' and t3.month_of_quarter ='1' and t4.biztype_cd ='01'
    -> group by t3.month_of_quarter
    -> ) s1
    -> right join
    -> (
    -> select
    -> t3.month_of_quarter
    -> ,sum(t1.order_count) as index_value_g4
    -> ,sum(t1.order_amt) as index_value_g5
    -> ,count(distinct t1.opp_customer_no) as index_value_g6
    -> from test.a05_cust_trade_stat t1
    -> left join test.dim_cfg_date t3
    -> on t1.event_create_dt=t3.calendar_date
    -> left join test.dim_cfg_biztype t4
    -> ON t1.biz_type = t4.biztype_cd
    -> where t3.year_of_calendar='2015' and t3.month_of_quarter >='1' and t3.month_of_quarter<='12'
    -> and t4.biztype_cd ='01'
    -> group by t3.month_of_quarter
    -> ) s2
    -> on s1.month_of_quarter=s2.month_of_quarter;
+---------------+-----------------+---------------+---------------+-----------------+---------------+
| index_value_g1 | index_value_g2  | index_value_g3 | index_value_g4 | index_value_g5  | index_value_g6 |
+---------------+-----------------+---------------+---------------+-----------------+---------------+
|        6060183 | 37165139810.8316 |         780478 |       6060183 | 37165139810.8316 |         780478 |
|           NULL |            NULL |          NULL |       4801011 | 31190642913.7099 |         679120 |
|           NULL |            NULL |          NULL |       4499856 | 27811269977.1278 |         613360 |
+---------------+-----------------+---------------+---------------+-----------------+---------------+
3 rows in set (22.67 sec)
```

Fig. 6 Two tables associated query 6 indexes take 22 s

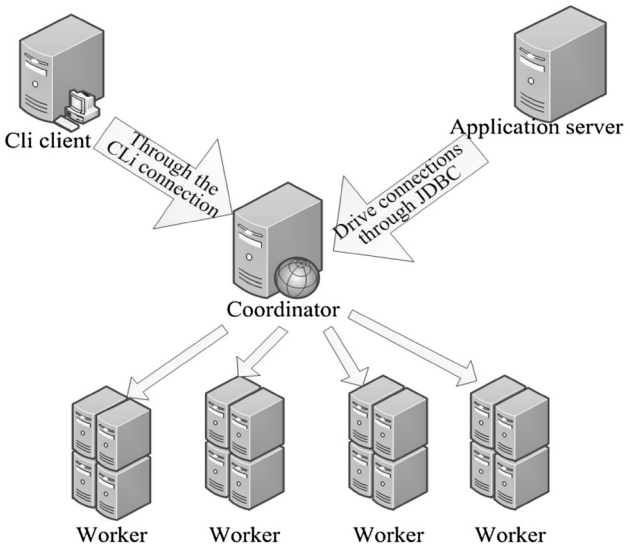

Fig. 7 Hardware system architecture diagram

will be broken down into multiple Tasks distributed in the Worker, each Task exists dependencies with its upstream and downstream Task in the data stream, and each Task will be further subdivided into multiple operators, each of them represents an operation on the data stream. Whenever the query starts, it will continue to perform Task on each Worker, and each processing a Buffer size of the data, it will transfer the results to the Task in the downstream phase, so that the transmission of data real-time dynamic can be basically guaranteed is on the client side. The Presto client submits a query statement to the Coordinator, which decomposes the query into tasks and executes them by every worker. The Coordinator obtains the final query result from the Worker in SingleStage and returns the query result to the client. The realization of Presto's software system is shown in the figure (Fig. 8).

Performing a query was divided into seven steps in Presto which can be seen from the figure, so we could import the data from one data source into another data source using a simple SQL statement as Presto can easily support multiple data sources and can support mixed computing of multiple data sources. And data in Mysql or Postgre databases can be imported into Hive via the createtable…asselect…statement.And then realized that imported the data in Hive into Mysql through the secondary development. In practical applications, with the use of azkaban scheduling system, scheduled or fixed-frequency scheduling Presto execution scripts can be implemented to accomplish related ETLs.This use of Presto provided a new ETL solution that was more convenient and faster than the traditional ETL solution. In real-time data computing, quasi-real-time data flow analysis mainly refers to the use of SQL through presto-kafka message queue to analyze and calculate the data flow in Kafka. There were two scenarios in the actual use of the process, respectively were retaining historical data and query data. Specific system architecture as shown below (Fig. 9).

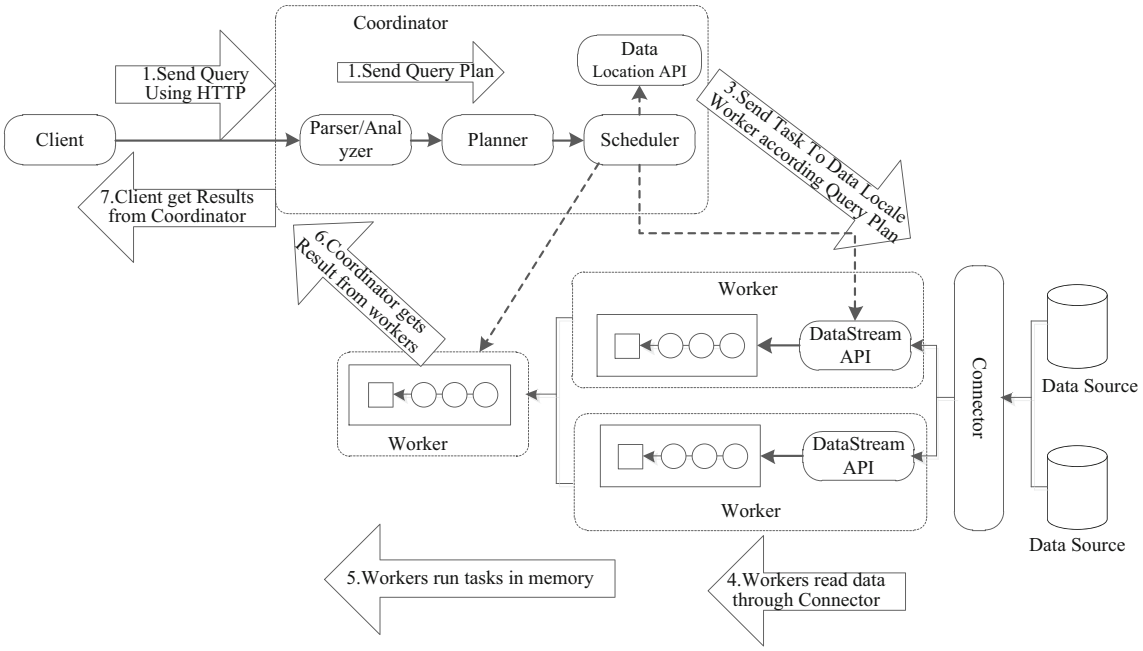

Fig. 8 Software system architecture diagram

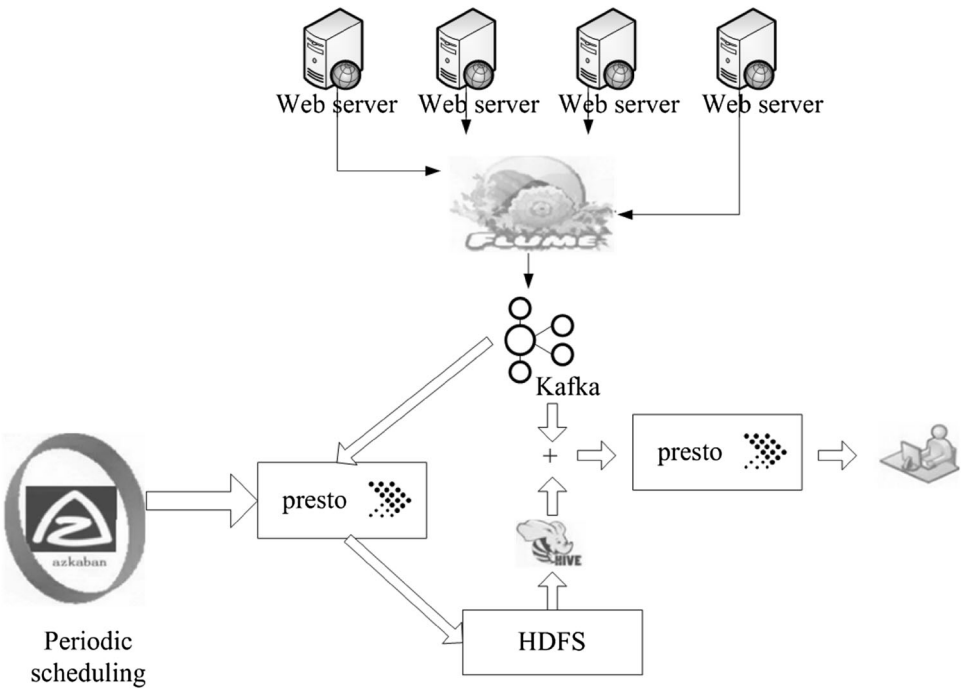

Fig. 9 Quasi real time calculation of data flow chart

As can be seen from the figure above: we needed to cooperate with presto-hive and presto-kafka to complete the calculation for the needs of analysis and calculation of kafka historical data.

4.3 Engine component with real-time computing-storm

In the above real-time computing scenarios, the processed data was prepared in the data warehouse in advance. There was another demand in the actual application scenario was that the data dynamically access the real-time response to the traditional online transaction processing system, which

was unable to support the real-time flow of massive data processing. Storm, based on Twitter-open-source, was elected as a technical selection after investigation and survey. We embed points in the user's front page and read the user's operation log as a data source access to the message queue, and Storm calculation engine to pull the data computing from the message queue, and displayed to the front page. Here we made a architecture for the entire real-time data flow taking a hierarchical data architecture idea (Fig. 10).

In the system realization of real-time data acquisition and analysis, achieving real-time capture and resolve work for data log of mainstream data source and achieving real-time capture and resolution for the operation log of sqlserver, mysql, oracle and hbase through the development of data collection system based on the log system. After the data was collected, the data would be passed to the message queue, which in this mainly played the role of data buffer. To consider the real-time data push for Storm, we used Kafka and a distributed message queue as the data transfer pipeline at the front of the Storm computing engine. Kafka is a distributed message queue system that forms the basis of the Data Transfer Processing Pipeline, including production and consumer terminals, and is now used by many different types of companies as multiple types of data pipelines and messaging systems (Fig. 11).

As shown in the figure, the overall flow is that the data source used push mode to post the message to the message queue server, and the data consumer terminal used the pulled mode to subscribe and consume messages from the message queue server. Kafka cluster of our system contains several Producers, such as browsing records generated by spotting in the web front-end, or server logs, etc.,a number of servers of broker core and Kafka's support for horizontal expansion, the more the number of broker was, the higher the throughput rate of the cluster would be. Producers push the data to the Broker in the form of data push. ConsumerGroup, the data consumer, pulled the data in the form of data pullout according to the business demand with a certain frequency or scheduling task. Intermediate would make configuration management for the overall message queue cluster through a Zookeeper cluster. Kafka managed the cluster configuration through Zookeeper, elected leaders, and redistributed data as ConsumerGroup changed. In this way we have realized the function of message queue module, and provided the foundation for the upper real-time data calculation. In Stom's flow calculation engine, the flow of calculation data is as shown in the figure below (Fig. 12).

From the picture above, the Spout component read the data from the external data source kafka according to a certain frequency and sent a message to Topology after the message queue getting the message: the level of frequency of t knock Le « was also pre-set,for example, the front page required a refresh every 30 s, then we could set the data transmission interval with 30 s. Spout was divided into reliable and unreliable modes and could be setting by configuring. If this message was not successfully processed by Stann, the reliable Spout could re-transmit the message, while the unreliable Spout component would not resend the message once it had been sent. There were two important methods of Spout, namely were ack and fail, ack method was called after Storm checking the successful implementing of the message,otherwise it would call the fail method, only reliable spout would call these two methods. The Bolt component then received the message sent by the Spout component of the data source. The Bolt component processed the message logically. All business operations processed by message were encapsulated in Bolt, in which we could filter, aggregate and store the result and progress other operations. Bolt received the data from the Spout and processed it in parallel, and the processing of complex streams could be done by multiple Bolt components. In the figure above,we defined the calculation of the parallel transaction amount for multiple orders of CountBolt. Multiple CmmtBolts were computed in parallel and then summarized in MergeBolt.We pre-set the rules, such as real-time statistics to show the total amount of men's wear's day sales.We could filter in CountBolt in parallel first and initial aggregated for those Bolt in line with the men's clothing, and then assigned into MergeBolt through the StreamGrouping data stream task mentioned above for

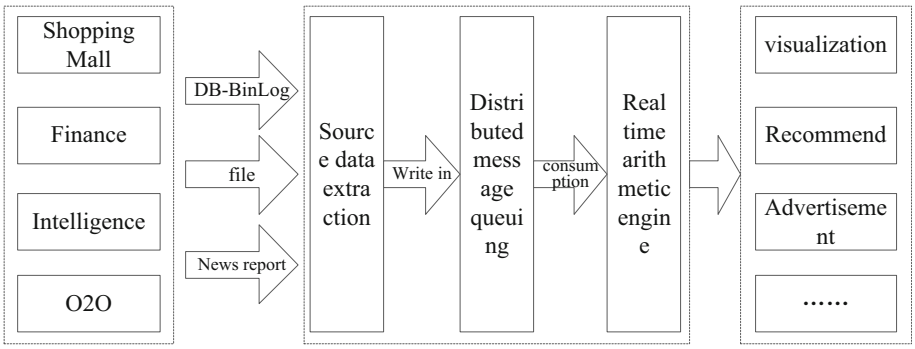

Fig. 10 Real-time data flow chart based on Storm

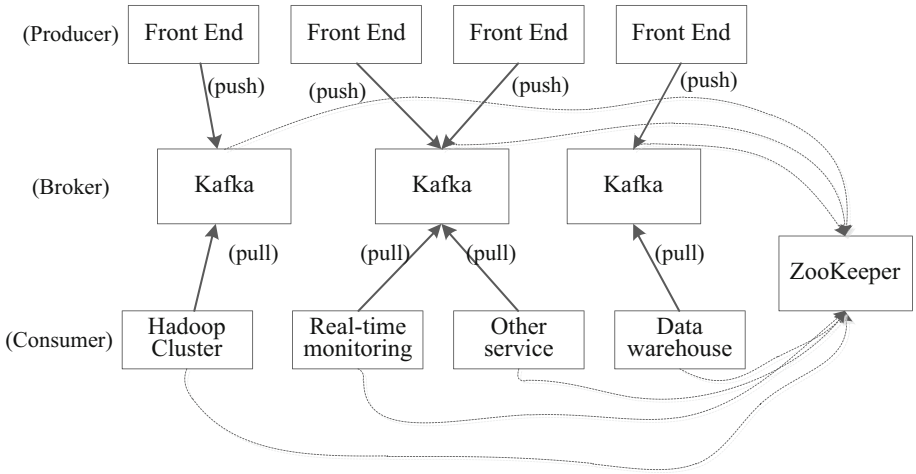

Fig. 11 Real-time data flow chart based on Storm

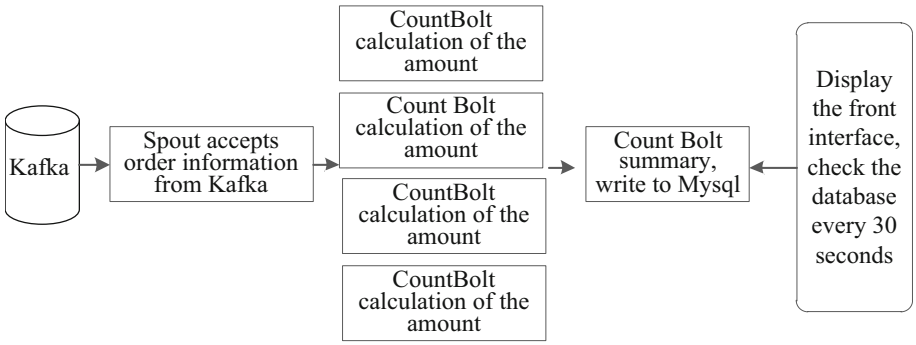

Fig. 12 The calculated data flow chart of the Spout and Bolt components

the final total, and then deposited into the mysql database. The front page read data from the raysql database in real time and presented it to the customer.The picture below shows the real-time data Kanban feature through the Storm engine.

5 Conclusion

The paper mainly describes how the e-commerce big data platform system based on distributed computing builds from the underlying data warehouse to the real-time computing engine, and to real-time recommendations for the value of today's data-driven business. It is convenient to maintain for operation and maintenance staff and warehouse engineers through the metadata management system. Helping marketers to predict sales through the reporting system and user portraits. According to the degree of response to different marketing methods of the history of the user, marketing costs, marketing products and the relationship between marketing effectiveness, locking target population more accurately and targeted marketing to improve marketing efficiency and reduce marketing costs. Based on the recommendation system, the Hoeffding tree-based stochastic forest model is trained on the online behavior of user's portrait, product portrait and knowledge map, and the continuously adjusted model is fed back to the forecasting engine in time. The closed-loop formation of this data makes real-time prediction more accurate. On the system architecture, it is recommended that the system adopt the solution of Cassandm + Lucene deployed on the same machine with the storage and indexing service to improve the stability and robustness of the recall service, reducing the number of remote calls, and reducing the time delay.

References

1. Huang, T.E., Guo, Q., Sun, H.A.: Distributed computing platform supporting power system security knowledge discovery based on online simulation. IEEE Trans. Smart Grid. **8**(99), 1–1 (2016)
2. Wu, P.L., He, F., Zhong, Y.: An envelope analysis algorithm of experiment data based on distributed computing platform. Cyberspace Secur. **1**(2), 12–14 (2017)
3. Guo, W., Liu, F.: Research on parallel algorithm based on Hadoop distributed computing platform. Int. J. Grid Distrib. Comput. **8**(1), 77–78 (2015)
4. Zhang, X., Jiang, J., Zhang, X., et al.: A data transmission algorithm for distributed computing system based on maximum flow. Cluster Comput. **18**(3), 1157–1169 (2015)

5. Rho, S., Park, S., Hwang, S.: Privacy enhanced data security mechanism in a large-scale distributed computing system for HTC and MTC. Int. J. Contents. **12**(2), 6–11 (2016)
6. Kumar, R.S., Chandrasekharan, E.: A parallel distributed computing framework for Newton-Raphson load flow analysis of large interconnected power systems. Int. J. Electr. Power Energy Syst. **73**, 1–6 (2015)
7. Hofmann, M., Rünger, G.: Sustainability through flexibility: building complex simulation programs for distributed computing systems. Simul. Model. Pract. Theory. **58**, 65–78 (2015)

Junmin Hu (1966.4), male, native of Zhangjiajie in Hunan, Tujia Nationality, Vice Principal in Party School of Guangxi District Organs of C. P. C, Professor, Ph.D. in economics, major research areas in energy economics, regional economics and international business theory.