Scientific Journal of Impact Factor (SJIF): 5.71
e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 5, Issue 03, March -2018

Automating Characterization Deployment in Distributed Data Stream Management Systems

Dhanasekar.S[1], Chaithanya.M.N[2], Gayathri.R.S[3], Nandhini.N[4]

[1]*Assistant Professor, Info Institute of Engineering, Kovilpalayam, Coimbatore- 641 107*
[2,3,4]*UG graduate, Info Institute of Engineering, Kovilpalayam, Coimbatore- 641 107*

Abstract: *DDSMS composed of two layers: upper layer – Relational Query Systems (RQS) and lower layer – Stream Processing Systems (SPS).After query submission to RQS, query planner needs to get converted into DAG consisting tasks running on SPS.SPS configure different deployment strategies based on query requests and data stream properties.Introducing four-level feature extraction, it uses different query workload as training sets to predict resource usage.Select optimal resource configuration from candidate settings based on current query requests and stream properties.Finally, validate the approach on open source SPS-Storm.*

Keywords: *RQS, SPS, Four level feature extraction, Optimal resource configuration, Candidate settings, SPS-Storm.*

I INTRODUCTION

Orient Stream ,a resource efficient system to implement dynamic characterization deployment for Storm configuration according to the current data flow rate. This is achieved by using a two-phase mechanism First multiple resource usage models are trained incrementally with accumulated historical data features. Then, whenever necessary, these models can be used for deriving the most resource-efficient configuration to be deployed. With the advance of the query process, the amount of training data is dynamically increased. In order to use regression models to predict the resource usage in real time, not need to re-build the models of all training data repeatedly and save the time and resource for building models, choosing incremental learning technique based on MOA and Weka toolkits second introducing Tachyon as an in-memory file server to store the state information, and use the system of Operator States Migration(OSM) to achieve live adjustment from different configurations without building a new topology and also to adjust the operator parallelism dynamically thereby achieving accurate queries without missing operator states. Orient stream focus on constructing training sets for the data stream properties and the parameter configurations of different levels, predicting resource usage by online machine learning techniques, and dynamically tuning the related parameter configurations of DDSMS. Re-balance mechanism of storm can only decrease operator parallelism without increasing. We introduce the mechanism of outlier execution detection, predict the rationality of configurations, and monitor the anomalous executions. Through the methods of normalization and nearest neighbors, we find abnormal data to further improve the prediction accuracy of regression models.

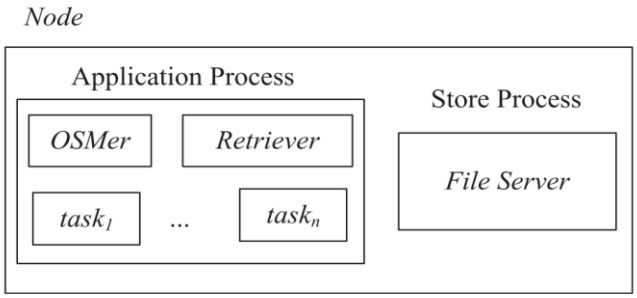

Fig. 1. OSM architecture

II RELATED WORK

To meet users' different query requests, DDSMS needs to build a complete operator library in RQS and dynamically set up the parallelism of processing nodes of SPS based on different query requests. There are roughly three lines of related work as below.

Dynamic Load Scheduling: Recently, several algorithms have been proposed for the strategies of dynamic load scheduling about Storm. For example, Aeolus is an optimizer to set up the parallel degree and the batch size of intra-node dataflow programs. It defined a cost model based on four measurements of processing tuple time, including shipping

time, queuing time, pure processing time and actual processing time. According to the model, Aeolus can get the best configuration scheme for the parallel degree of operators and batch size.
Machine Learning Techniques: Khoshkbarforoushha et al. proposed a model based on the Mixture Density Network for resource consumption estimation of data stream processing workloads. This model can help users to make a decision about whether to register a new query or not. ALOJA project presented an open, vendor neutral repository, featuring over Hadoop executions. It can predict query time and abnormal detection based on ALOJA-ML, which is a framework for using machine learning techniques to interpret Hadoop benchmark performance data and performance tuning. ALOJA-ML can only be applied to Hadoop, not predict application scenes of data stream. BO4CO can only analyze the collected historical training data of SPS, but cannot do the incremental analysis of new data.
Resource Estimation for RQS. As we know, RQS often has the SQL-like query interface. Some research has been devoted to monitoring resource consumption of SQL queries. Li et al. set up two kinds of mechanisms of feature extractions: coarse-grained global features and finegrained operator-specific features for different queries over Microsoft SQL Server. Akdere et al. built three models for predicting query performance based on different query plans: query plan-level model, operator-level model and hybrid model for nested query. However, The two models only consider the static selection process, cannot monitor dynamically, and do not consider the system characteristics under RQS.
OrientStream is orthogonal to the above works since we focus on constructing training sets for the data stream properties and the parameter configurations of different levels, predicting resource usage by online machine learning techniques, and dynamically tuning the related parameter configurations of DDSMS.

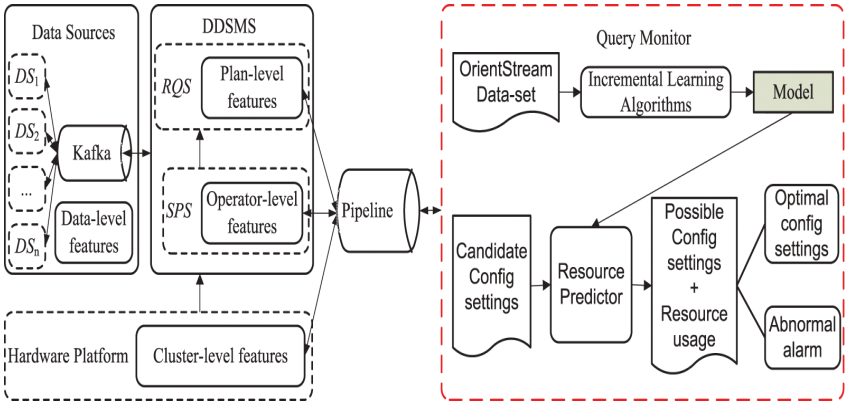

Fig. 2. OrientStream architecture

III PRELIMINARIES

This section presents the related concepts of Storm and OrientStream, and gives the problem definition and description.

3.1 Storm Synopsis

In this paper, Storm is used as an experimental platform of SPS. It is a distributed processing framework that can process data streams in real time. Storm consists of a nimbus node and a number of supervisor nodes.
The nimbus node is responsible for allocating resources, scheduling requests and monitoring each supervisor node; each supervisor node is responsible for accepting requests assigned by nimbus and managing the processes (called workers)under the node. Each worker contains multiple threads (called executors), and each executor can run one or more component instances (called tasks). The entire Storm cluster uses Zookeeper to provide the coordinated management of the distributed application.

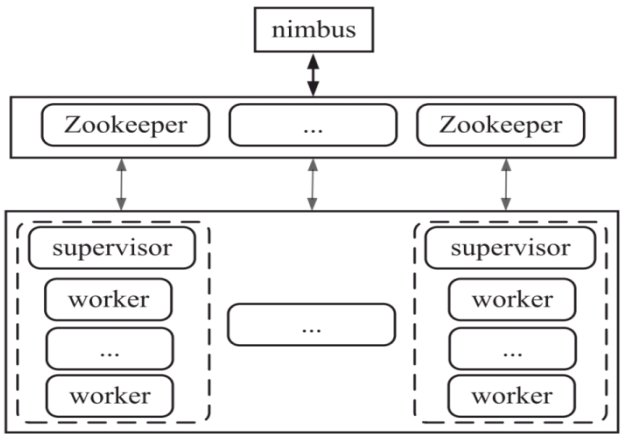

Fig. 3. Strom architecture

3.2 Concept Description

OrientStream needs to monitor the operation of DDSMS in real time, and collects training samples based on different data stream rate and the parameter configurations. Formally, we define the related concepts as follows:

Definition 1: A window model can divide the infinite data stream into a number of finite sub-streams. Each query processing is only aimed at the sub-stream in the current window. Generally, we can set the window size based on the time interval or the number of tuples.

Definition 2: The processing latency of DDSMS is defined as the sum of processing delays of each tuple processed by operators. Each operator has the different parallelism, which is often implemented in a multi-threaded manner. Then, the latency of data source i is represented as the mean latency of every thread, and the formal definition is as follows:

$$\text{Latency}(\text{source}_i)= \sum^{m}_{j=1}(\text{Latency}(\text{source}_i)_j)/m_i \quad (1)$$

where, m is the parallelism of the data source node. However, the processing latency of DDSMS is the maximum latency of each data source, it can be defined as below:

$$\text{Latency}=\max_{i=1}{}^{n}\text{Latency}(\text{source}_i) \quad (2)$$

where, n is the number of data sources in a query request.

3.3 Problem Definition

Query requests always need to set different parameter configurations by user's experience. Whether the parameter settings are reasonable or not will directly affect the throughput and the latency of DDSMS, as well as the resource usage.

With the change of data stream rate, we should dynamically

adjust parameter configurations to meet user's needs of the latency and the throughput thresholds. But, SPS are often not allowed to adjust parameters arbitrarily. For example, "re-balance" mechanism of Storm can only decrease operators' parallelism without increasing them.

We need to predict the resource usage, the latency and the throughput for any possible configurations. Based on the predicted results, the optimal configuration is defined as the one using minimal CPU and memory resource while the latency and the throughput requirements are still guaranteed under the thresholds set by users. Thus, the problem can be modeled as a resource usage optimization problem .

Let $N=(n_1,n_2,n_3,\ldots)$ be the set of all nodes in a cluster. The CPU usage U_{CPU} and the memory usage U_{Memory} of each node n_i can be defined as follows:

$$U(n_i)=\alpha*U_{CPU}(n_i)+\beta*U_{Memory}(n_i) \qquad (\alpha+\beta=1), \quad (3)$$

where, α and β are the weight of the CPU and memory usage respectively, which set as 50 percent by default. Then, let $C=(c_1,c_2,c_3,\ldots$ be the candidate configurations supplied by users. For the entire cluster, we need to predict the optimal configuration c_{opt} to achieve the following optimization goals

$$\text{Minimize}\sum_{n_i \epsilon N}U(n_i) \quad (4)$$

$$\text{s.t. } R(\text{latency})<T(\text{latency}),$$

$$\text{and } \quad R(\text{throughput})>T(\text{throughput}),$$

where, R(latency) and R(throughput) are the processing latency and the throughput of query requests. T(latency) and T(throughput) are the thresholds set by users.

The constructed models fall into two categories, i.e., regression and classification models. The regression models are constructed for predicting the resource usage based on their numeric labels. In order to realize regression models with higher accuracy, we select four independent models that support incremental learning and have the highest regression accuracy. For predicting the processing latency and throughput, we build binary classification models to get whether the latency and throughput can meet the system specification. In this paper, we use the J48 algorithm in Weka to classify, which is the implementation of the C4.5 algorithm. J48 algorithm is based on the decision tree, and its attribute with the maximum gain ratio is selected as the splitting attribute. The gain ratio of attribute A is defined as

$Gainrate(A) = \frac{Gain\,(A)}{SplitInfo_{\,A}(D)}$, where Gain(A) is the information gain of attribute A, $\text{SplitInfo}_A(D)$ is the potential information generated by splitting the training data set D into v partitions, corresponding to the v outcomes of a test on attribute A.

IV SYSTEM ANALYSIS

4.1 Existing System

In existing system, User data is generated automatically through a variety of signal processing, data analysis/enrichment techniques before being stored in the web applications. Pushing such data into web applications introduces a challenge since such automatically generated content is often ambiguous and may result in objects with probabilistic attributes. Such probabilistic data must be "determinized" before being stored in legacy web applications. The problem of mapping probabilistic data into the corresponding deterministic representation as the *determinization* problem.

Disadvantage:

- The problem of determinization has not been explored extensively in the past.
- The main drawback of this approach is that it is *unaware* of the query workload and thus does not necessarily optimize the given quality metrics, which leads to lower quality.

4.2 Proposed System

We introduce the problem of *determinizing* probabilistic data. Given a workload of triggers/queries, the main challenge is to find the deterministic representation of the data which would optimize certain quality metrics of the answer to these triggers/queries. We propose a framework that solves the problem of determinization by minimizing the expected cost of the answer to queries. We develop a branch and- bound algorithm that finds an approximate near-optimal solution.
We address the problem of determinizing a collection of objects to optimize set-based quality metrics, such as F-measure. We develop an efficient algorithm that reaches near-optimal quality. We extend the solutions to handle a data model where mutual exclusion exists among tags. We show that correlations among tags can be leveraged in our solutions to get better results. We also demonstrate that our solutions are designed to handle various types of queries.
Advantage:

- The proposed algorithms are very efficient
- High-quality results that are very close to those of the optimal solution

V ALGORITHMS

At this time, OrientStream data-set is generated in real time.Through the experimental comparison, we choose four incremental learning models in MOA and Weka toolkits with the highest prediction accuracy.
Bayes model:We use the Naïve Bayes(NB) from the MOA toolkit.Bayes algorithm updates internal counters with each new instance and uses the counter to assign a class in a probabilistic fashion to a new item in the stream.
Hoeffding trees model:We use the HoeffdingTree(HT),also from the MOA toolkit.A hoeffding tree or VFDT is an incremental decision tree that is capable of learning from massive data stream.
Online bagging model:We use this model from the MOA toolkit and the hoeffding tree as the base learner.Oza and Russell propose the online bagging algorithm that gives each instance a weight for re-sampling.
Nearest neighbors model:We use the IBK , instance-based learning algorithm from the Weka toolkit,by introducing updateable classifier method to achieve incremental learning characteristics.It is identical to the K nearest neighbors algorithm expect that it normalizes its attributes ranges,processes instances incrementally.

5.1 EDK Regression Model

Ensemble learning is one of the most effective methods to improve the accuracy of the regression algorithm, which is considered as one of the most effective learning ideas. Its contribution is to improve regression accuracy by aggregating multiple base models. Ensemble learning methods mainly include Bagging and Boosting . Although both of mathematical foundations are not exactly the same for improving the accuracy of regression algorithms, they attempt to produce efficient committees by combining the outputs of multiple base models to improve the performance of regression algorithms. However, Bagging and Boosting only ensemble the same regression model (e.g., hoeffding tree).
Four models have no contact and the prediction is independent of each other. Therefore, we combine these four models and propose a regression model EDK Regression (Ensemble Different Kind of Regression).
Without loss of generality, EDK Regression combines k base regression models $R_1,R_2,\ldots,R_k$ and creates an improved composite regression model R_ by scoring the regression models. For the new arrival tuple tn, EDK Regression can obtain the prediction result of the vote by k base regression models.

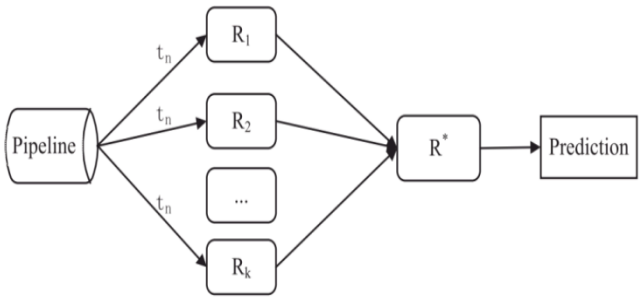

Fig. 4. EDK Regression prediction process

Algorithm 1. EDK Regression Algorithm

Require:
Each prediction values of base learning models for predicting sample n ;
Ensure:
The prediction value of EDK Regression for predicting sample n;
1: Computing AVG_N;
2: for (i = 1;i < n; i++) do
3: Computing RAEn;

4: end for
5: for (j = 1;j < K; j++) do
6: Computing Fn;
7: end for

In the process of prediction, first, EDK Regression lets each base regression model to make prediction; then, according to the relative absolute error (RAE) value of the each base regression model, EDK Regression calculates the voting weights respectively; finally, it determine the regression value of the sample with a weighted vote.

5.2 Abnormal Detection Mechanism

Due to the abnormal execution of the cluster (e.g., the cluster load is not balanced, network transmission is not stable), or the observation does not fit into the model, it results in some abnormal training data. Based on the difference between the predicted value and the observed value (e.g., resource usage), we divide the training data into three types:

- Normal. Detected data can be predicted correctly by learning models.
- Warning. Detected data is mis-predicted, and more than a half of nearest neighbors to the data are mispredicted too.
- Outlier. Detected data is mis-predicted, but more than a half of nearest neighbors to the data are predicted correctly.

Algorithm 2. Abnormal Detection Algorithm

Require:
Prediction Error Threshold Te;
The number of tuples to nearest tag tuple k_{NN};
The regression file F contains predicted value and true value;
Ensure:
The type of detected tuple t;
1: if (|t:PredictedValue _ t:TrueValue| < Te) then
2: t is Normal;
3: else
4: for (i = 1; i < Number of TestDataSet; i++) do
5: Computing the k_{NN} tuples nearest to t;
6: end for
7: for (j = 1;j < kNN; j++) do
8: if ($|t_k$:PredictedValue _ tk:TrueValuej > Te) then
9: $N_{warning}$ + 1;
10: else
11: $N_{outlier}$ + 1;
12: end if
13: end for
14: if ($N_{warning} > N_{outlier}$) then
15: t is Warning;
16: else
17: t is Outlier;
18: end if
19: end if

Algorithm 2 gives the detailed description of the detection procedure. For each detected data, first, we select the normal data based on the error threshold Te (Algorithm 2 lines 1-2). Then, for each abnormal test data, we compute the k_{NN} tuples that are nearest to the test data. Lines 4-6 in Algorithm 2 depicts this process. Next, we count the number of warning or outlier of these k_{NN} tuples by the definition of warning or outlier category (Algorithm 2 lines 7-13). Finally, we judge whether the test data is warning or outlier (Algorithm 2 lines 14-18).

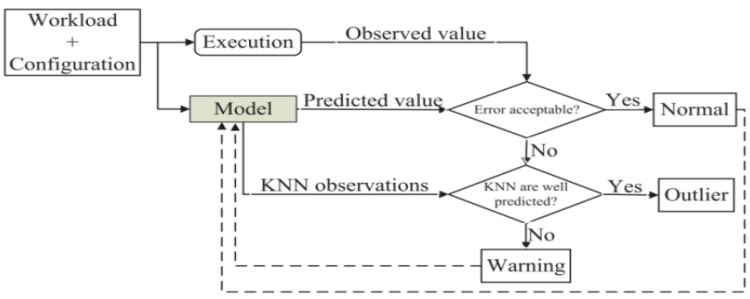

Fig. 5. Anomaly detection mechanism flow

Fig. 5 shows the anomaly detection flow. We can improve the prediction accuracy by discarding abnormal training data and not getting it into the incremental learning model. Besides using this method to test new observations, we can re-train our models by subtracting observations marked as outlier.

VI CONCLUSION AND FUTURE WORK

In this paper, we propose Orient Streamto predict the resource usage of DDSMS by using incremental learning techniques, develop the learning tool that can be implemented data acquisition automatically based on the different workloads. Our approach uses MOA and Weka toolkits to build real-time prediction models by defining four levels features extraction at different granularities. In addition, we introduce the EDK Regression algorithm and the outlier execution detection mechanism for improving the prediction accuracy, and verify the validity of our methods. Moreover, we introduce the system of the Operator States Migration to achieve live adjustment of the operator parallelism from different nodes. It can reduce the adjustment time and processing latency significantly.
We evaluate OrientStream by running three workloads with different runtime characteristics. Our experimental results demonstrate that OrientStream can significantly reduce 8-15 percent CPU usage and 38-48 percent memory usage, and save more resources with incrementally updated models. Hence, it is promising to deploy OrientStream in production to significantly increase the user's return-on investment.

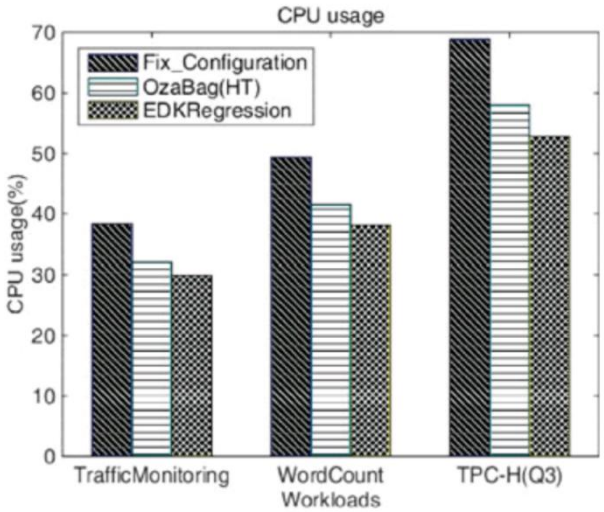

Fig. 6. CPU usage

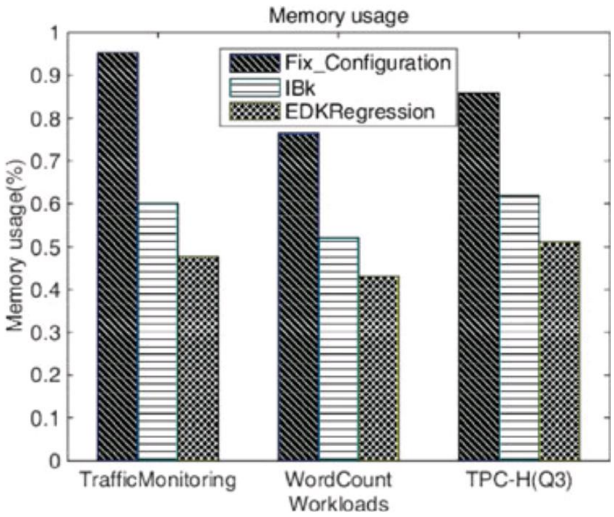

Fig. 7. Memory usage

As future work, there are two directions to further improve our work. (1) We need to collect more training data to improve the prediction accuracy by using the incremental learning, and plan to further optimize the regression model to improve the prediction accuracy and use less samples. (2) In this paper, the optimal configuration is predicted from the manually supplied candidate configurations. So, we need to design the real-time recommendation to help users get the optimal configuration.

VII REFERENCES

[1] https://spark.apache.org/streaming/
[2] https://moa.cs.waikato.ac.nz/
[3] https://github.com/intel-hadoop/HiBench/
[4] L.Neumeyer, B.Robbins, A.Nair and A.Kesari, "S4:Distributed stream computing platform", inProc. IEEE Int. Conf. Data Mining Workshops, 2010, pp. 170-177.
[5] T.Z.J.Fu, J.Ding, R.T.B.Ma, M.Winslett, Y.Yang and Z.Zhang, "DRS:Dynamic Resource Scheduling for real-time analytics over fast streams", Computer Science, vol 690, no 1, pp. 411-420, 2015.
[6] P.Domingos and G.Hulten, "Mining high-speed data streams", in Proc. ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining, 2000, pp. 71-80.
[7] P.Jamshidi and G.Casale, "An uncertainity-aware approach to optimal configuration of stream processing systems", in Proc. IEEE Int. Symp. Model. Anal. Simul. Comput. Telecommun. Syst. Computer Soc., 2016, pp. 39-48.
[8] C.Wang, X.Meng, Q.Guo, Z.Weng and C.Yang, "OrientStream: A framework for dynamic resource allocation in distributed data stream management systems", in Proc. 25th ACM Int. Conf. Inf. Knowl. Manage., 2016, pp. 2281-2286.

