Resa: Realtime Elastic Streaming Analytics in the Cloud

Tian Tan[1,2] Richard T. B. Ma[2] Marianne Winslett[2]
Yin Yang[2] Yong Yu[1] Zhenjie Zhang[2]
[1]Shanghai Jiao Tong University [2]Advanced Digital Sciences Center
{tantian, yyu}@sjtu.edu.cn {tbma, winslett, yin.yang, zhenjie}@adsc.com.sg

ABSTRACT
We propose Resa, a novel framework for robust, elastic and realtime stream processing in the cloud. In addition to traditional functionalities of streaming and cloud systems, Resa provides (i) a novel mechanism that handles dynamic additions and removals nodes in an operator, and (ii) a node re-assignment scheme that minimizes output latency using a queuing model. We have implemented Resa on top of Twitter Storm. Experiments using real data demonstrate the effectiveness and efficiency of Resa.

Categories and Subject Descriptors
D.1.3 [**Programming Techniques**]: Concurrent Programming – *Parallel Programming.*

General Terms
Algorithms, Design.

Keywords
cloud, stream, migration, resource allocation.

1. PROBLEM STATEMENT
We focus on realtime streaming analytics in the cloud. Most existing data stream management systems (DSMSs), e.g., Aurora [1], STREAM [2], etc., fail to provide sufficient functionalities that are crucial for the cloud platform, e.g., elastic execution. Meanwhile, existing cloud-based paradigms mostly focus on static, batch data processing. Our goal is to build a system for realtime streaming analytics that combines the advantages of both a DSMS and a cloud-based paradigm. Such a system should provide (i) *elasticity*, i.e., the ability to dynamically adjust to the amount of available computational resources, (ii) *fault tolerance*, i.e., robustness against node failures, (iii) *realtime processing*, which requires producing outputs within strict time limits.

2. BRIEF TECHNICAL OVERVIEW

2.1 Framework
The basic programming unit in Resa is an *operator*. The user specifies the logic of each operator, and the topology of operator interconnections. An operator is performed in parallel by multiple computational nodes. Unlike existing parallel DSMSs, Resa requires that the number of nodes in an operator be *dynamically adjustable*. To do this, each input tuple is represented by a key-value pair, and it is assigned to a node according to the hash value of its key. For a stateful operator, the user also specifies a map from input keys to local storage entries.

SIGMOD'13, July 22–27, 2013, New York City, New York, USA.

2.2 Intra-Operator Migration
When the number of nodes changes in a stateful operator, Resa migrates operator states between different nodes. Consider again a frequent pattern count operator. Unlike traditional plan-level migration [3], the migration in Resa occurs inside a single operator. When adding a new node *N*, a portion of the inputs that were previous handled by other nodes will be handled by *N*. Hence, intra-operator migration moves the corresponding counts of these inputs from their previous handlers to *N*. Conversely, when removing a node *N*, Resa moves the states of *N* to their new handlers. Currently, the migration is implemented by flushing the states of all nodes to a distributed storage system, and re-reads them to the nodes according to the new inputs assignment. The flushed states are also kept for fault recovery purposes.

2.3 Node Re-Allocation
Resa monitors the each job's response time. When a job's response time exceeds (resp., is far below) the user specified threshold, Resa assigns new (resp., removes) nodes to the jobs. Once the total number of nodes changes for a job, Resa must re-allocate these nodes to the operators, such that the response time is minimized. Note that assigning more nodes to an operator does not necessarily improves the job's overall response time, since a downstream may become the bottleneck. We derive a cost model for a job's response time, based on the theory of *Jackson networks*. This model takes into account both the operators' scalability to the number of nodes, and the topology structure. Given this model, node re-allocation becomes a constrained optimization problem. We prove the convexity of the problem, and solve it through gradient descent.

3. SUMMARY OF MAJOR RESULTS
We implemented Resa based on Twitter Storm (*storm-project.net*), and evaluated its performance for outlier detection and frequent pattern mining, using two datasets: *WorldCup*, which contains 5 million network packages, and *Twitter*, which contains 28.6 million tweets from 2.1 million users in 3 years. The evaluation results confirm (i) that our cost model accurately predicts system response time, (ii) that intra-operator migration successfully adjusts an operator's performance based on the number of assigned nodes, and (iii) that the proposed node allocation scheme obtains high-quality assignments.

4. REFERENCES
[1] Abadi, D., Carney, D., Cetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, S. Zdonik. Aurora: A New Model and Architecture for Data Stream Management. *VLDB J.*, (12)2: 120-139, 2003.
[2] Motwani, R., Widom, J., Arasu, A., Babcock, B., Babu, S., Datar, M., Manku, G., Olston, C., Rosenstein, J., Varma, R. Query Processing, Resource Management, and Approximation in a Data Stream Management System. *CIDR*, 2003.
[3] Yang, Y., Krämer, J., Papadias, D., Seeger, S. HybMig: a hybrid approach to dynamic plan migration for continuous queries. *IEEE TKDE*, 19(3): 398-411, 2007.