Proceedings of the 11th International Conference on Future Internet Technologies (CFI2016)

June 15-17, 2016

Nanjing, China

ICPS
Published by ACM

ISBN: 978-1-4503-4181-3

Co-host: AsiaFI, Southeast University
Co-sponsor: CERNET, WIDE, FIF, STARRYSKY

Committee

General Co-Chairs
Guang Cheng (Southeast Univ., China)
Serge Fdida (UMPC, France)

Technical Program Committee Chair
Jun Bi (Tsinghua Univ., China)

Technical Program Committee Co-Chairs
Sonia Fahmy (Purdue Univ., USA)
Xiaoming Fu (Univ. of Goettingen, Germany)
Keisure Uehara (Keio Univ., Japan)

Technical Program Committee
Karl Andersson (Lulea Univ. of Technology, Sweden)
Jiannong Cao (Polytechnic Univ. of Hong Kong)
Jiachen Chen (Rutgers Univ., USA)
Xu Chen (Univ. of Goettingen, Germany)
Guang Cheng (Southeast Univ., China)
Reza Farahbakhsh (Institut Mines-Télécom, Télécom SudP, Paris)
Shang Gao (Dekean Univ., Australia)
Guofei Gu (Texas A&M Univ., USA)
Toru Hasegawa (Osaka Univ., Japan)
Xiaohong Huang (BUPT, China)
Rittwik Jana (AT&T Labs - Research, USA)
Yusheng Ji (National Institute of Informatics, Japan)
Lei Jiao (Bell Labs, USA)
Gunnar Karlsson (KTH The Royal Inst. Tech., Sweden)
Jongwon Kim (GIST, Korea)
Minseok Kwon (RIT, USA)
Jun Li (University of Oregon, USA)
Ruidong Li (NICT, Japan)
Wenzhong Li (Nanjing Univ., China)
Bingyang Liu (Tsinghua Univ., China)
Sangheon Pack (Korea Univ., Korea)
Debbie Perouli (Marquette Univ., USA)
Zhuzhong Qian (Nanjing Univ., China)
Stefano Secci (LIP6 UPMC, France)
Yuji Sekiya (Univ. of Tokyo, Japan)
Charles Shen (Columbia Univ., USA)
Stephan Sigg (Aalto Univ., Finland)
Yuzo Taenaka (Univ. of Tokyo, Japan)
Atsushi Tagami (KDDI R&D Laboratories Inc., Japan)
Xin Wang (Fudan Univ., China)
Kui Wu (Univ. of Victoria, Canada)
Ikjun Yeom (SKKU, Korea)
Yung Yi (KAIST, Korea)
Beichuan Zhang (Univ. of Arizona, USA)
Lin Zhang (BUPT, China)
Zenghua Zhao (Tianjin Univ., China)

Publicity Co-Chairs
Xiaohong Huang (BUPT, China)
Sangheon Pack(Korea Univ., Korea)
Tomohiro Ishihara (Univ. of Tokyo, Japan)

Publication Co-Chairs
Bingyang Liu (Tsinghua Univ., China)
Keiichi Shima(IIJ, Japan)

Poster and Special Session Co-Chairs
Xinggong Zhang (Peking Univ., China)
Yongqiang Dong (Southeast Univ., China)

Web/Local Chair
Xiaoyan Hu (Southeast Univ., China)

Steering Committee Co-Chairs
Dae Young Kim (CNU, Korea)
Jun Murai (Keio Univ., Japan)
Jianping Wu (Tsinghua Univ., China)

Steering Committee
Kenjiro Cho (IIJ II, Japan)
YangHee Choi (SNU, Korea)
Kilnam Chon (KAIST/Keio Univ., Korea/Japan)
Jon Crowcroft (Cambridge Univ., UK)
Hiroshi Esaki (The Univ. of Tokyo, Japan)
Serge Fdida (LIP6, France)
DongMan Lee (KAIST, Korea)
YoungHee Lee (KAIST, Korea)
Xing Li (Tsinghua Univ., China)
Yan Ma (BUPT, China)
Craig Partridge (BBN, USA)
Shinji Shimojo (NICT, Japan)
Joe Touch (USC, USA)
Xingwei Wang (NEU, China)
Lixia Zhang (UCLA, USA)
Martina Zitterbart (Univ. Karlsruhe, Germany)

Table of Contents

Constructing Multiple Steiner Trees for Software-Defined Networking Multicast . . . 1
Jehn-Ruey Jiang and Szu-Yuan Chen

Detecting Network Topology and Packet Trajectory with SDN-enabled FPGA Platform . . 7
Mian Dai, Guang Cheng and Yuxiang Wang

An Easy Defense Mechanism Against Botnet-based DDoS Flooding Attack Originated in SDN Environment Using sFlow . . . 14
Yiqin Lu and Meng Wang

Performance Evaluation of TCP Congestion Control Algorithms in Data Center Networks 21
Truc Anh Nguyen, Siddharth Gangadhar and James Sterbenz

Energy-efficient localization and tracking on smartphones: Design principle and solutions . 29
Yufeng Wang, Yuanting Bu, Qun Jin and Athanasios V. Vasilakos

RBASA Real-Time User Behavior Analysis System for Internet TV in Cloud Computing . 36
Chengang Zhu, Guang Cheng, Xiaojun Guo and Yuxiang Wang

Optimized Root Selection Algorithm for Many-to-Many Communications in ICN . . . 43
Ruidong Li, Hitoshi Asaeda and Klaus Moessner

Spray and Hop Distance Routing Protocol in Multiple-island DTN Scenarios . . . 49
Agussalim Agussalim and Masato Tsuru

Directed-Propagation Link-State Routing Protocol for Information-Centric Networks . . . 56
Lemei Huang, Yu Guan, Xinggong Zhang and Zongming Guo

Establish the Intrinsic Binding in Naming Space for Future Internet Using Combined Public Key . . . 62
Hongwei Meng, Zhong Chen, Jianbin Hu and Zhi Guan

Verification of a Multi-Domain Authentication and Authorization Infrastructure Yamata-no-Orochi . . . 69
Kei Mikami, Daisuke Ando, Kunitake Kaneko and Fumio Teraoka

Detecting Compromised Email Accounts from the Perspective of Graph Topology . . . 76
Xuan Hu, Banghuai Li, Yang Zhang, Changling Zhou and Hao Ma

Protecting DNS services from IP spoofing - SDN collaborative authentication approach . . . 83
Nor Masri Sahri and Koji Okamura

Flo-v: Low Overhead Network Monitoring Framework in Virtualized Software Defined Networks . . . 90
Gyeongsik Yang, Kyungwoon Lee, Wontae Jeong and Chuck Yoo

Implementing a Dynamically Reconfigurable Wireless Mesh Network Testbed for Multi-Faceted QoS Support . . . 95
Siyoung Jang, Byoungheon Shin and Dongman Lee

Efficient Matching Algorithm for Content-based Publish/Subscribe Systems with Compound Predicates . . . 99
Xilai Lv and Yongqiang Dong

A Framework for the Analysis of BGP Data over Long Timescales 104
Geng Zhang, Dahua Zhang, Liang Zhou and Min Zhu

Enabling Fast Recovery from Unanticipated Data Access Failure in Named Data Networking 109
Xiaoyan Hu and Jian Gong

Quantifying the Flattening of Internet Topology 113
Yangyang Wang and Keyao Zhang

Deployment and Evaluation of Software-Defined Inter-Connections for Multi-domain Federated SDN-Cloud 118
Aris Cahyadi Risdianto, Jun-Sik Shin and Jongwon Kim

RBAS：A Real-Time User Behavior Analysis System for Internet TV in Cloud Computing

Chengang Zhu
School of Computer Science and Engineering, Key Laboratory of Computer Network and Information Integration, Ministry of Education
Southeast University
Nanjing China 211189
cgzhu@njnet.edu.cn

Guang Cheng
School of Computer Science and Engineering, Key Laboratory of Computer Network and Information Integration, Ministry of Education
Southeast University
Nanjing China 211189
gcheng@njnet.edu.cn

Xiaojun Guo
School of Computer Science and Engineering, Key Laboratory of Computer Network and Information Integration, Ministry of Education
Southeast University
Nanjing China 211189
xjguo@njnet.edu.cn

Yuxiang Wang
School of Computer Science and Engineering, Key Laboratory of Computer Network and Information Integration, Ministry of Education
Southeast University
Nanjing China 211189
yxwang@njnet.edu.cn

ABSTRACT

The characteristic of Internet TV user behavior is quite essential for designers to optimize resource schedule and improve user experience. With the rapid development of Internet, both Internet TV users and STB (set top boxes) models are booming. This brings a large amount of behavior data which requires matching computing and storage resource to process. Therefore, scalable Internet TV user behavior analysis becomes more difficult. As a solution, cloud computing framework such as Hive is emerged. But limited by performance, it's not an appropriate choice for interactive analysis or real-time data exploration. In this paper, we present a real-time Internet TV user behavior analysis system with advantages of high concurrency, low latency and good transportability. Firstly, we design an event capture scheme, consisted of agents embedded in STBs and capture server clusters, to capture every manipulation performed by users. Secondly, we develop a SQL-on-Hadoop engine with distributed transactional management to decrease the response time. The engine has excellent query performance and ability to interactively query various data sources in different Hadoop formats. Lastly, we evaluate RBAS in a commercial Internet TV platform of 16 million registered users. The results show that, with a 32-node cluster, the system can effectively process 10.2 TB of behavior data every day, which is about 40x faster than original Hive-based system.

Categories and Subject Descriptors

C.2 [Computer-Communication Network]: Miscellaneous; C.4 [Performance of Systems]: Measurement Techniques;

Keywords

User behavior analysis, cloud computing, Internet TV, SQL-on-Hadoop

1. INTRODUCTION

With the wide spread availability of broadband and development of multimedia technology, ISP (Internet Service Providers) cooperates with TV stations to offer Internet TV service. Compared with traditional broadcast TV service, Internet TV is more flexible and abundance. Besides, it provides users with interactive and personalized watching experience. As a result, by the end of the fourth quarter in 2014, the number of global IPTV subscribers reached 117.39 million, of which 39 million are in China. Nevertheless, in order to guarantee user experience, Internet TV consumes large quantities of network bandwidth leading to network congestion and high latency. By analyzing behavior data, ISP is able to know the user access pattern, optimize the resource schedule, and achieve a better user experience with minimum overhead.

The current study of Internet TV user behavior is still in its early stage. Both measuring methods and research results have a certain degree of limitation. Stem from commercial consideration, the source code and communication protocols of ISP Internet TV system remain in private. Moreover, in order to avoid interference from measurement, the communication between video server and client is encrypted, which makes the user behavior analytics extremely difficult. Therefore, it's quite necessary to involve ISP in measurement work to improve the coverage ratio and precision ratio. In this paper, cooperating with an Internet TV service provider, we present a real-time massive Internet TV user behavior data analytic system. The main contribution of our work is as follows:

(1) This work presents an Internet TV user behavior events capture scheme, which provides a good transportability as well as high throughput performance. It is consisted of event agents embedded in STBs and capture servers carefully organized as clusters. The agents encapsulate every user operation into UDP packets, and can be deployed on all kinds of STBs that are conformed to the standard of the next generation broadcasting network television operating

system (NGB TVOS) [1]. According to experiments results, the events capture system is able to handle 25 million packets of events sent by millions of STBs per minute.

(2) In order to reduce the response time of the analysis process, we develop a SQL-on-Hadoop engine, which can be flexibly scaled to support large numbers of concurrent short operational updates along with long running operational reporting queries. We implemented a distributed transaction management infrastructure that can facilitate transactional updates spanning multiple rows, tables, and statements across both structure data in relational tables, as well as semi-structure data in HBase tables.

The paper is organized as follows. In section2, we describe the related work on Internet TV measurement and analysis. The architecture of Real-Time User Behavior Analysis System for Internet TV is explained in section 3, and experiment results are presented in section 4. Finally, section 5 concludes this paper.

2. RELATED WORK

According to the observation agent whether to participate in the Internet TV system, measurement methods can be divided into active and passive measurement. Active measurement is based on the crawlers to obtain system topology snapshot and node information. Crawlers must be complete or partial implemented communication protocol of the measured system. Researchers usually use reverse engineering methods to obtain enough information about system protocol and topology which is essential to build the crawlers. Hei et al. [2] use active measurement methods to study user behavior of PPLive, including online user number in a particular channel, the variation of online user number and dynamic of users. Vu et al. [3,4] implement a protocol crawler of PPLive and found that the number of online users in particular channel presents characteristics like time-sensitive, self-repeated and event-dependent. Jiang et al. [5] develop and deploy a P2P IPTV crawler supporting multiprotocol, present a large measurement study of user behavior and network topology in PPLive, PPStream and UUsee. Active measurement is suited for a particular Internet TV system and lack of generality. In addition, traffic caused by crawlers influences the original appearance of user behavior, which is interference with measurement results.

There are two kinds of passive measurement methods: one is to set observation host to record and analyze information of the Internet TV system. The other is to deploy log servers to collect user behavior information. Hall et al. [6] analyze the video traffic data collected from Joost clients and present features of VOD (video on demand) system, overlay network topology and user behaviors. Huang et al. [7] deploy log servers in PPLive to collect view records including when and where users watch movies, precisely describe the access pattern and online time. Yu et al. [8] analyze the watching behavior, the arrival and departure pattern, content access mode of China Telecom VOD system and come to the conclusion that the user access pattern is predicable. Based on log information of Imagenio IPTV system, Cha et al. [9] analyze the characteristics of view sessions and channel popularity. In contrast to active measurement, passive measurement method is system-independent, which can be used in varied situations, and makes a few bad effects on the measured system. However, perspectives of passive measurement are limited and the result is incomplete.

Under the restrictions of computing framework, it's difficult to process massive user behavior data in real-time for traditional measurement method mentioned above. The emergence of cloud computing framework, represented by the Hadoop and Hive [10], effectively solves the problem of storage and computing for big data. Ding et al. [11] adopt Hadoop and HBase to store large-scale network measurement data. Lee et al. [12] design a Hadoop-based traffic monitoring system that performs IP, TCP, HTTP, and NetFlow analysis of multi-terabytes of Internet traffic in a scalable manner. Liu et al. [13] implement a traffic monitoring and analysis system for large-scale network based on Hadoop and deploy it in the core network of a large cellular network. However, due to batch job framework, all those works are not suitable for interactive analytics which is the key to promote behavior data exploration and rapid user prototyping.

Recently there has been increasing interests in building real-time query processing tools that answer queries directly, without invoking MapReduce. Cloudera Impala [14] processes SQL queries over data stored in HDFS, the file system of Hadoop, and/or in HBase. Spark SQL [15] is the online query processing module built on top of Spark [16], a novel in-memory big data processing platform. Both Impala and Spark SQL are lack of transaction capability, which makes analysis program is quite error prone. Therefore, we need to develop a SQL-on-Hadoop engine with distributed transaction management, which can be flexibly scale to support large numbers of concurrent short operational updates along with long running operational reporting queries.

3. SYSTEM OVERVIEW

RBAS includes an STB events capturer, a distributed data storage and an interactive analysis engine. The overview of RBAS architecture is described as figure 1.

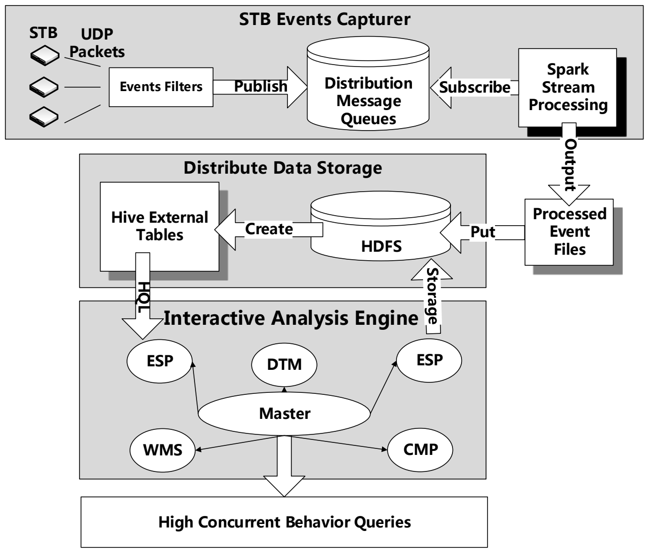

Figure 1. Overview of RBAS architecture

(1) STB events capturer. When users manipulate STB, agents embedded in STB capture the interrupts of TVOS, translate interrupts into predefined message structures, and encapsulate message data into UDP packets transmitting to event filtering modules deployed in regional management servers.

(2) Distributed data storage. According to the locality of STB, type of behavior events and timestamp, behavior data is published into different topics as distributed message queues, processed in streaming, and stored in native Hive or HBase tables using their native storage engines and data format. Storage provides a relational schema abstraction on top of HBase. Traditional relational database objects (schemas, tables, views, secondary indexes, stored procedures) are supported using familiar DDL (data

definition language)/DML (data manipulation language) semantics including object naming, column definition and data types support.

(3) Interactive analysis engine. It is comprised of a number of services or processes used for interactive analysis. Services include connection management, SQL statement compilation and optimized execution plan creation, SQL execution (both parallel and non-parallel) against database objects (tables in HBase), transaction management, and workload management. It provides transparent parallel SQL execution as warranted thereby eliminating the need for complex map-reduce programming development.

3.1 STB Events Capturer

STB events capturer is consisted of event agents embedded in set top boxes(STBs) and capture servers carefully organized as clusters. In order to be compatible with different STB models manufactured by different vendors, the agents are developed based on NGB TVOS which is official standard conformed by manufacturers in China. User behaviors are monitored by process forked by the agents, encapsulated into one of 30 predefined message structures and stored in the output buffer. To minimize the overhead of transmission, dozens of messages are sent in a UDP packet and the interval of transmission is set at 5 minutes.

The process of capturing and filtering STB events is described in Figure 2. Region servers collect the packets containing behavior data. During the peak time, the number of packets sent per second usually reaches about 20,000,000 per minute. Servers are carefully designed into clusters to process so many packets and publish behavior data into the correspond topic queue. Load balancer is introduced to distribute packets to the server with minimum response time.

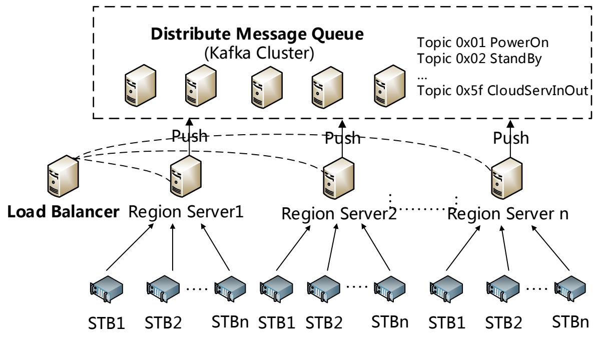

Figure 2. Capturing and filtering STB events

The captured behavior data involves user access pattern, program popularity and view session duration, from which view habits and preferences can be revealed. It's flexible to expand behavior events by adding new event codes. According to perspectives to measure, service providers can update agent program online without extra hardware investment. Little time delay is caused in whole capture process.

3.2 Distributed Data Storage

Internet TV measurement system usually stores data in relational database systems and data model are designed based on the relational model. The size of data depends on the scale of users and measurement duration. As the scale of users and measurement duration increases, the volume of behavior data grows exponentially [17].

Considering both capacity and scalability of centralized storage is limited, we leverage HBase for performance, scalability and availability. These features are key to supporting interactive analysis workload with high concurrency. Fine grained load balancing, scalability, and parallel performance is provided via standard HBase services such as auto-sharding data across multiple regions and region servers. Data availability and recovery when a server or disk fails or is decommissioned is provided by standard Hadoop and HBase services such a replication and snapshots. Figure 3 describes the architecture of RBAS storage.

Although RBAS stores database objects in HBase/HDFS storage structures, it differs and brings value-add over HBase in a multitude of ways. RBAS provides a relational schema abstraction on top of HBase which allows researchers to leverage known and well tested relational design methodologies and SQL programming skills. From a physical layout perspective, RBAS uses standard HBase storage mechanisms (column family store using key-value pairs) to store and access objects. RBAS stores all columns in a single column family to improve access efficiency and speed for operational data. Additionally, RBAS incorporates a column name encoding mechanism to save space on disk and to reduce messaging overhead for improving SQL performance.

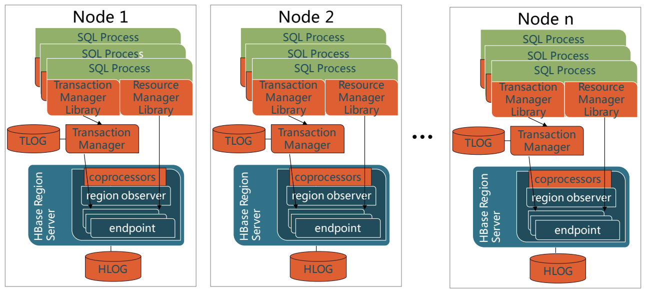

Figure 3. RBAS storage over HBase

Unlike HBase that treats stored data as an un-interpreted array of bytes, RBAS defined columns are assigned specific data types that are enforced by RBAS when inserting or updating its data contents. This greatly improves data quality and integrity. Also it eliminates the need to develop application logic to parse and interpret the data contents. HBase provides ACID (atomic, consistent, isolated and durable) transaction protection only at the row level. RBAS extends ACID protection to application defined transactions that can span multiple SQL statements, multiple tables, and rows. This greatly improves database integrity by protecting the database against partially completed transactions i.e. ensuring that either the whole transaction is completely materialized in the database or none of it. HBase's native API is at a very low level and is rarely used programming API. In contrast, RBAS API is ANSI SQL which is a familiar and well known programming interface and allows companies to leverage existing SQL knowledge and skills. Unlike HBase's key structure that is comprised of a single un-interpreted array of bytes, RBAS supports the common relational practice of allowing the primary key to be a composite key comprised of multiple columns. Finally, unlike HBase, RBAS supports the creation of secondary indexes that can be used to speed transaction performance when accessing row data by a column value that is not the row key.

In order to support transactional workloads across structured and semi-structured data, RBAS has implemented a Distributed Transaction Management infrastructure that will facilitate transactional updates spanning multiple rows, tables, and statements across both structured data in relational tables, as well as semi-structured data in HBase tables. In this architecture, the work related to transaction management is truly distributed.

Multiple processes that are initiating transactions talk to a local Transaction Manager (TM) to coordinate the transaction. The TM is only involved in coordinating the transaction initiation, commit, abort, auditing, and recovery operations. It does not keep track of all the transaction updates or reads against the HBase regions. Those are delegated to the regions. So it is very lightweight. The TM is multi-threaded and can be scale-up, handling as many transactions in parallel via threads as necessary. One can also define multiple logical nodes per physical nodes, each with its own TM process, if it is deemed to be necessary for scale-up.

The job of managing the transaction is separated from the actual updates for each transaction. The Resource Manager Library is used by the analysis engine process to make the transactional versions of the get / put / delete / getScanner calls directly to the HBase regions involved in the transaction. That is, TM does not get involved in tracking these calls or the updates. The updates themselves are tracked in a completely distributed way as well by the HBase region, via a thread of the Endpoint coprocessor running in the region server, for the region hosting the transactional call. Again, the coprocessor does not communicate the updates related to a transaction to the TM. It talks to the TM minimally, only when transaction coordination activities – such as commit or abort are initiated.

3.3 Interactive Analysis Engine

Interactive analysis engine encapsulates all of the services required for managing database objects as well as efficiently executing submitted SQL database requests. Services include connection management, SQL statement compilation and optimized execution plan creation, SQL execution (both parallel and non-parallel) against database objects, transaction management, and workload management. It provides transparent parallel SQL execution as warranty to eliminate the need for complex map-reduce programming development. Figure 4 describes the query execution workflow.

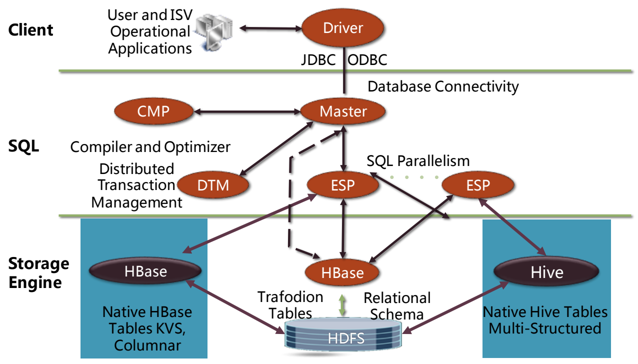

Figure 4. Query execution workflow

When the analytic requests to open a connection, RBAS's database connection services (DCS) process the request and assign the connection to a Master SQL process. RBAS uses Zookeeper to coordinate and manage the distribution of connection services across the cluster for load-balancing purposes as well as to ensure that an analytic session can immediately reconnect in the event when the assigned master process should fail.

The Master process is responsible for coordinating the execution of SQL statements passed from the analytic application. The Master calls upon the compiler and optimizer process (CMP) to parse, compile, and generate the optimized execution plan for the SQL statements.

If the optimized plan calls for parallel execution, the Master divides the work among Executive Server Processes (ESP) to perform the work in parallel on behalf of the Master process. The results are passed back to the Master for consolidation. In some situations, where there a highly complex plan specified (e.g. large n-way joins or aggregations), multiple layers of ESPs may be requested. If a non-parallel plan is generated, then the Master calls upon HBase services directly for optimal performance.

For distributed transaction protection services the RBAS DTM service is called upon to ensure the ACID (atomicity, consistency, isolation and durability) protection of transactions across the Hadoop cluster. DTM (distributed transaction management) calls upon RBAS supplied HBase coprocessor services that provide transaction resource management on behalf of HBase.

Last, but not least, HBase, HBase coprocessors, and HDFS services are called upon by either the Master or ESP processes using standard and native API's to complete the I/O requests i.e. retrieving and maintaining the database objects. Where appropriate RBAS will push down SQL execution into the HBase layer using filters or coprocessors.

RBAS's SQL executor uses a dataflow and scheduler-driven task model to execute the optimized query plan. Each operator of the plan is an independent task and data flows between operators through in-memory queues (up and down) or by inter-process communication. Queues between tasks allow operators to exchange multiple requests or result rows at a time. A scheduler coordinates the execution of tasks and runs whenever it has data in one of its input queues. RBAS's executor model is starkly different from alternative SQL-on-Hadoop DBMS that store intermediate results on disk—for example, spool space. In most cases, the RBAS's executor is able to process queries with data flowing entirely through memory, providing superior performance and reduced dependency on disk space and I/O bandwidth. The executor incorporates 3 types of parallelism:

(1) Partitioned parallelism which is the ability to work on multiple data partitions in parallel. In a partitioned parallel plan, multiple operators all work on the same plan. Results are merged by using multiple queues, or pipelines, enabling the preservation of the sort order of the input partitions. Partitioning is also called "data parallelism" because the data is the unit that gets partitioned into independently executable fractions.

(2) Pipelined parallelism is an inherent feature of the executor resulting from its dataflow architecture. This architecture interconnects all operators by queues with the output of one operator being piped as input to the next operator, and so on. The result is that each operator works independently of any other operator, producing its output as soon as its input is available. Pipelining occurs naturally and is engaged in almost all query plans.

(3) Operator parallelism is also an inherent feature of the executor architecture. In operator parallelism, two or more operators can execute simultaneously, that is, in parallel. Except for certain synchronization conditions, the operators execute independently.

Therefore, RBAS naturally provides parallelism without special processing such as Hadoop map-reduce programming or coding on the part of the application client. An individual query plan produced by the optimizer can contain any combination of partitioned, pipelined, or operator parallelism. The degree of parallelism at any plan stage may vary depending on the optimizer's heuristics.

4. EXPERIMENTS

We deploy RBAS in a commercial Internet TV platform of 16 million registered users. All the systems are deployed on a 32-node cluster connected by two 48-port 10 GigE switches, of which 30 nodes are used for Data Nodes, and 2 nodes are used for management nodes. Each node has two 2.93GHz Intel Xeon 6-coreprocessors, and runs 64-bits CentOS 6.3 with 2.6.23 kernel. And it has 256GB memory, 2 900GB SAS hard disks, 5 600GB SSD hard disks and one Intel X520 Dual-Port 10 GigE NIC. Table 1 show the overall workload and performance in this environment.

Table 1. Overview of workload and performance

Metric	Value
Link speed	32*10G
Total number of registered user	16 million
Average number of online user per day	9.5 million
Average number of STB events packet per day	7.2 billion
Size of HDFS files per day	10.2 Tbytes
Average CPU utility of analysis server	40.6%
Average Mem utility of analysis server	67.1%

We run a behavior analysis style benchmark consisting of a subset of the queries on a 250TB queries data set. Based on the amount of data, queries are categorized into interactive, reporting and deep analytic queries. For our comparisons, we used the most popular SQL-on-Hadoop systems, such as Impala, SparkSQL and Hive, to show results.

4.1 Single-User and Multi-User Performance

Figure 5(a) compares the performance of the four systems on single-user runs, where a single user is repeatedly submitting queries with zero think time. RBAS outperforms all alternatives on single-user workloads across all queries run. Its performance advantage ranges from 2.1x to 13.0x and on average is 6.7x faster.

RBAS superior performance becomes more pronounced in multi-user workloads, which are ubiquitous in real-time behavior analytic applications. Figure 5(b) shows the response time of the four systems when there are 100 concurrent users submitting queries from the interactive category. In this scenario, RBAS outperforms the other systems from 6.7x to 18.7x when going from a single user to concurrent user workloads. The speed varies from 10.6x to 27.4x depending on the comparison.

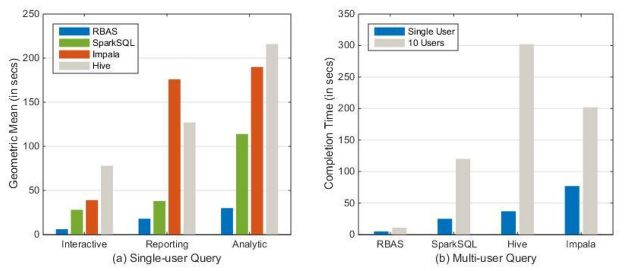

Figure 5. Comparison of query response times on single-user and multi-user

4.2 Simple selection and Complex join queries

Based on the complexity of their execution plans, we categorize behavior analytic queries into two groups: simple selection queries and complex join queries.

Simple selection queries: all these queries either perform simple selection and aggregation against one single table, or straightforward join of two or three tables. Here we take quantitative amount of program watching time as an example. Only the viewrecord table is involved. The execution plan for this query is simple: first a sequential scan, and then a two-phase aggregation. Figure 6(a) shows that, for such simple queries, RBAS is 10x faster than Hive. There are factors contributing to this performance gap. Firstly, the task start-up and coordination of RBAS is more efficient than YARN. Secondly, data movement in RBAS is pipelining, while MapReduce jobs materialize the output of each stage, either locally or remotely on HDFS.

Complex join queries: The common characteristics of these queries are that the selection and aggregation operations are run against several tables of various sizes, involving a join on 3 or more tables. These queries are too complex to be optimized manually. In such case, the join order and data movement efficiency have significant impacts on query execution performance. As illustrated in Figure 6(b), RBAS is 40x faster than Hive for complex join queries. Besides the factors discussed previously for simple selection queries, this huge performance gain is also closely related to planning algorithms. Given the table statistics, RBAS employs a cost-based query optimizing algorithm, which makes it capable to figure out an optimal plan. Hive uses a simple rule-based algorithm and makes little use of such hints. Thus, most of the time, Hive can only give a sub-optimal query plan. Furthermore, multiple table join introduces large volume data movement.

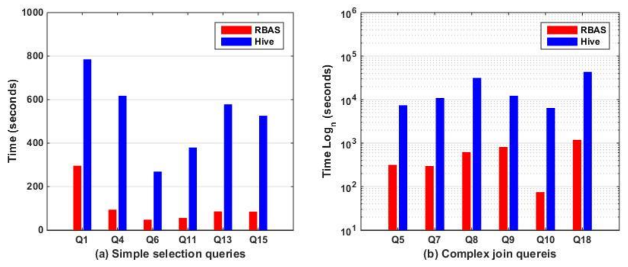

Figure 6. Comparison of response times on simple selection queries and complex join queries

4.3 Scalability

To evaluate the scalability of RBAS, two tests are designed. The first one assigns 300GB data on each node, while the other has a total data size to 10TB and distribute the data to 32 nodes by distribution columns. We run the analysis queries on the cluster size of 8, 16, 24, and 32 nodes. It measures how well RBAS can scale as new nodes added in the cluster. The result of the first test is shown in Figure7(a), where the red lines are the actual test results, while the blue lines are the ideal linear expectations. When the cluster size scales from 8 nodes to 32 nodes, the dataset being processed increases from 2.4TB to 9.6TB, and the execution time increases slightly, approximately about 11%. This result implies that the size of dataset which RBAS can process is near-linearly increased as the size of the cluster increases.

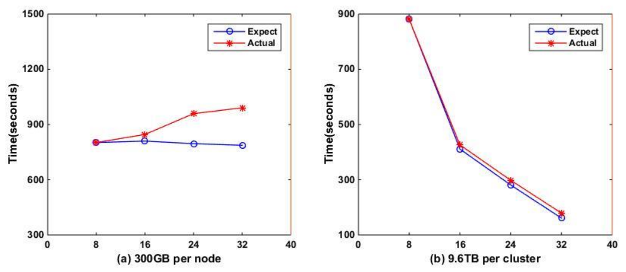

Figure 7. Scalability of RBAS as nodes increasing

Figure 7(b) illustrates the result of the second test. The execution time decreases from 850 seconds to 236 seconds, about 28% of the former one, as the cluster size scales from 8 nodes to 32 nodes. This result suggests that for a fixed dataset, the execution time is near-linearly decreased as the cluster size increases.

5. CONCLUSIONS

In this paper, we proposed a real-time Internet TV user behavior analysis system for high concurrency, low latency and good transportability. We designed an Internet TV user behavior events capture scheme, which is able to handle 25 million packets of event sent by millions of STBs per minute. Moreover, based on clouding computing platform, Hadoop, we develop a SQL-on-Hadoop engine with distributed transactional management, which has excellent query performance and can also interactively query various data sources in different Hadoop formats. This system has been deployed in a commercial Internet TV platform of 16,000,000 registered users and extensively evaluated. The results show that, with a 32-node cluster, the system can effectively process 10.2 TB of behavior data every day, which is about 40x faster than system based on the original Hive. For future work, we plan to improve the performance of analysis programs by optimizing execution plans based on statistics. In addition, we also plan to extend the current online analytical processing oriented system to support machine learning algorithm by leveraging Spark streaming processing technologies.

6. ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful feedback and XXXX for her kind guidance in developing the final draft. This work was supported by the National High-tech R&D Program of P. R. China (863 program) (2015AA015603), the Prospective Research Program on Future Networks of Jiangsu(BY2013095-5-03) and the Six Industries Talent Peaks Plan of Jiangsu (2011-DZ024). The corresponding author of this paper is Guang Cheng.

7. REFERENCES

[1] Delin Chen, Ying W, Liangfu Z., 2013. The main technical characteristics and software architecture of NGB TVOS[J]. Radio and television information, 2013, 10.

[2] HEI, X., LIANG, C., LIANG, J., LIU, Y., and ROSS, K.W., 2007. A Measurement Study of a Large-Scale P2P IPTV System. IEEE Transactions on Multimedia 9, 8, 1672-1687. DOI= http://dx.doi.org/10.1109/TMM.2007.907451.

[3] VU, L., GUPTA, I., LIANG, J., and NAHRSTEDT, K., 2007. Measurement of a large-scale overlay for multimedia streaming. In *Proceedings of the Proceedings of the 16th international symposium on High performance distributed computing* (Monterey, California, USA2007), ACM, 1272410, 241-242. DOI= http://dx.doi.org/10.1145/1272366.1272410.

[4] VU, L., GUPTA, I., NAHRSTEDT, K., and LIANG, J., 2010. Understanding overlay characteristics of a large-scale peer-to-peer IPTV system. *ACM Trans. Multimedia Comput. Commun. Appl. 6*, 4, 1-24. DOI= http://dx.doi.org/10.1145/1865106.1865115.

[5] JIANG, Z.-H., WANG, H., and FAN, P.-Y., 2011. Crawler-Based Measurement of Large Scale P2P IPTV Systems [J]. *Journal of Software 6*, 022. DOI= http://dx.doi.org/10.3724/SP.J.1001.2011.03849.

[6] HALL, Y.J., PIEMONTE, P., and WEYANT, M., 2007. Joost: A measurement study. School of Computer Science, Carnegie Mellon University, Technical Report.

[7] HUANG, Y., FU, T.Z.J., CHIU, D.-M., LUI, J.C.S., and HUANG, C., 2008. Challenges, design and analysis of a large-scale p2p-vod system. In *Proceedings of the Proceedings of the ACM SIGCOMM 2008 conference on Data communication* (Seattle, WA, USA2008), ACM, 1403001, 375-388. DOI= http://dx.doi.org/10.1145/1402958.1403001.

[8] YU, H., ZHENG, D., ZHAO, B.Y., and ZHENG, W., 2006. Understanding user behavior in large-scale video-on-demand systems. In *Proceedings of the Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems 2006* (Leuven, Belgium2006), ACM, 1217968, 333-344. DOI= http://dx.doi.org/10.1145/1217935.1217968.

[9] CHA, M., RODRIGUEZ, P., CROWCROFT, J., MOON, S., and AMATRIAIN, X., 2008. Watching television over an IP network. In *Proceedings of the Proceedings of the 8th ACM SIGCOMM conference on Internet measurement* (Vouliagmeni, Greece2008), ACM, 1452529, 71-84. DOI= http://dx.doi.org/10.1145/1452520.1452529.

[10] THUSOO, A., SARMA, J.S., JAIN, N., SHAO, Z., CHAKKA, P., ZHANG, N., ANTONY, S., LIU, H., and MURTHY, R., 2010. Hive - a petabyte scale data warehouse using Hadoop. In Data Engineering (ICDE), 2010 IEEE 26th International Conference on, 996-1005. DOI= http://dx.doi.org/10.1109/ICDE.2010.5447738.

[11] HAIJIE, D., YUEHUI, J., YIDONG, C., and TAN, Y., 2012. Distributed storage of network measurement data on HBase. In *Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on*, 716-720. DOI= http://dx.doi.org/10.1109/CCIS.2012.6664268.

[12] LEE, Y. and LEE, Y., 2012. Toward scalable Internet traffic measurement and analysis with Hadoop. *SIGCOMM Comput. Commun. Rev. 43*, 1, 5-13. DOI= http://dx.doi.org/10.1145/2427036.2427038.

[13] LIU, J., LIU, F., and ANSARI, N., 2014. Monitoring and analyzing big traffic data of a large-scale cellular network with Hadoop. *IEEE Network 28*, 4, 32-39. DOI= http://dx.doi.org/10.1109/MNET.2014.6863129.

[14] KORNACKER, M., BEHM, A., BITTORF, V., BOBROVYTSKY, T., CHING, C., CHOI, A., ERICKSON, J., GRUND, M., HECHT, D., and JACOBS, M., 2015. Impala: A Modern, Open-Source SQL Engine for Hadoop. In *CIDR*.

[15] ARMBRUST, M., XIN, R.S., LIAN, C., HUAI, Y., LIU, D., BRADLEY, J.K., MENG, X., KAFTAN, T., FRANKLIN, M.J., GHODSI, A., and ZAHARIA, M., 2015. Spark SQL: Relational Data Processing in Spark. In Proceedings of the Proceedings of the 2015 ACM SIGMOD International

Conference on Management of Data (Melbourne, Victoria, Australia2015), ACM, 2742797, 1383-1394. DOI= http://dx.doi.org/10.1145/2723372.2742797.

[16] ZAHARIA, M., CHOWDHURY, M., DAS, T., DAVE, A., MA, J., MCCAULEY, M., FRANKLIN, M.J., SHENKER, S., and STOICA, I., 2012. Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation* (San Jose, CA2012), USENIX Association, 2228301, 2-2.

[17] INDEX, C.V.N., 2014. Forecast and Methodology, 2012–2017 (2013). *URL:* http://www. cisco. com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360. pdf