WANalytics: Geo-Distributed Analytics for a Data Intensive World

Ashish Vulimiri
UIUC
vulimir1@illinois.edu

Carlo Curino
Microsoft CISL
ccurino@microsoft.com

P. Brighten Godfrey
UIUC
pbg@illinois.edu

Thomas Jungblut
Microsoft
thjungbl@microsoft.com

Konstantinos Karanasos
Microsoft CISL
kokarana@microsoft.com

Jitu Padhye
Microsoft Research
padhye@microsoft.com

George Varghese
Microsoft Research
varghese@microsoft.com

ABSTRACT

Many large organizations collect massive volumes of data each day in a geographically distributed fashion, at data centers around the globe. Despite their geographically diverse origin the data must be processed and analyzed as a whole to extract insight. We call the problem of supporting large-scale geo-distributed analytics *Wide-Area Big Data (WABD)*. To the best of our knowledge, WABD is currently addressed by copying all the data to a central data center where the analytics are run. This approach consumes expensive cross-data center bandwidth and is incompatible with data sovereignty restrictions that are starting to take shape. We instead propose *WANalytics*, a system that solves the WABD problem by orchestrating distributed query execution and adjusting data replication across data centers in order to minimize bandwidth usage, while respecting sovereignty requirements. WANalytics achieves an up to 360× reduction in data transfer cost when compared to the centralized approach on both real Microsoft production workloads and standard synthetic benchmarks, including TPC-CH and Berkeley Big-Data. In this demonstration, attendees will interact with a live geo-scale multi-data center deployment of WANalytics, allowing them to experience the data transfer reduction our system achieves, and to explore how it dynamically adapts execution strategy in response to changes in the workload and environment.

1. INTRODUCTION

Large-scale organizations process massive volumes of data each day. The total volume of data from activities such as logging user interactions and monitoring compute infrastructures can be on the order of 10s or 100s of TBs each day. The data are collected in a geo-distributed fashion across multiple data centers to ensure low latency for end-users, regulatory compliance, and availability. Analyzing all these geographically dispersed data as a whole is necessary to expose valuable insight. We call the problem of supporting analytics on large volumes of geographically distributed data *Wide-Area Big Data* or *WABD*.

WABD is typically addressed today by retrieving all the data to a single data center where the analytics are run. This *centralized approach* is problematic at large scales because it uses a substantial amount of cross-data center bandwidth, which is growing increasingly expensive and scarce [10, 7]. Moreover, this approach may soon be rendered impossible by sovereignty regulations governing data movement.

In previous work we proposed an alternative geo-distributed execution approach which pushes part of the computation to each data center whenever this is beneficial, avoiding moving data between data centers and thus reducing bandwidth costs [17, 16]. Our system, WANalytics, automatically optimizes query execution and data replication strategies to minimize bandwidth cost, while respecting given regulatory and fault-tolerance constraints. It employs several novel techniques, including a *pseudo-distributed execution* mechanism to collect precise measures of data transfer costs (used during query optimization), and aggressive *caching* of intermediate results, which further reduce data transfers. In our experiments, WANalytics achieved a reduction in data transfer costs of up to 360× compared to the centralized baseline both on a Microsoft production workload and several standard synthetic benchmarks.

We have designed two versions of the system: one that focuses exclusively on SQL computation [17], as well as a more general-purpose version that supports arbitrary DAGs of computational tasks [16]. This demonstration will showcase the more mature SQL branch of WANalytics.

During the demonstration, attendees will interact with a live deployment of WANalytics spanning data centers spread across multiple continents. They will be able to submit queries, obtain results, visualize in real-time the data transfer reduction WANalytics achieves over the centralized baseline, and examine how it adapts its execution strategy due to changes in the workload and environment. We will provide pre-defined example workflows that attendees can follow to expose the interesting features of our system. Demo

SIGMOD'15, May 31–June 4, 2015, Melbourne, Victoria, Australia.

ACM 978-1-4503-2758-9/15/05 ...$15.00.
http://dx.doi.org/10.1145/2723372.2735365.

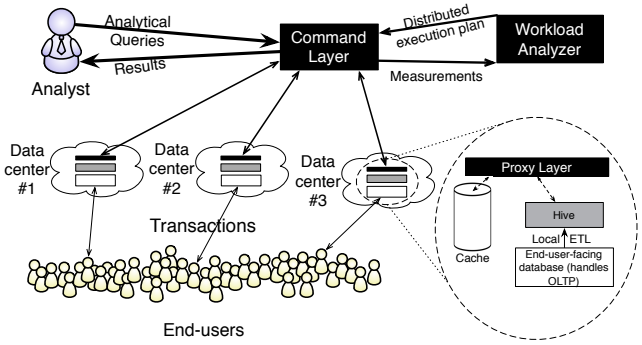

Figure 1: WANalytics architecture

attendees will also be able to write and execute their own arbitrary SQL queries.

We next provide a brief overview of WANalytics' design and capabilities (§2). We then describe the demonstration setup (§3) and provide an overview of related work (§5) before concluding (§5).

2. WANalytics OVERVIEW

Figure 1 shows an overview of WANalytics' architecture.

WANalytics manages data that are partitioned and/or replicated across several data centers. Data are generated externally by end-users processing transactions. WANalytics has no control over which data centers individual tuples are initially logged to — the data partitioning scheme is dictated by external considerations, such as the latency observed by end-users — but may replicate data to other data centers for performance and/or fault-tolerance.

The system presents an abstract, logically-centralized view over the distributed data. Analysts write and submit SQL queries against this logically centralized database. The WANalytics command layer receives queries, partitions them to create a distributed execution plan, and orchestrates distributed execution. At each data center the command layer interacts with a thin proxy layer managing a single-data center analytics stack (Apache Hive in our current implementation). The proxy layer provides support for transferring data between data centers and manages a cache used for the data transfer optimization in §2.3.

A workload analyzer (§2.1) analyzes measurements collected at runtime, potentially using pseudo-distributed execution (§2.2), to identify if changes to the query execution and data replication strategies would reduce data transfer costs. During normal WANalytics execution the workload analyzer would run periodically and occasionally, say once a day, but for the purposes of this demonstration we will reanalyze the workload each time an attendee submits a query.

2.1 Workload Analyzer

WANalytics is targeted towards applications with a slowly evolving core of recurrent queries. This matches our experience with Microsoft production workloads and is consistent with reports on data processing pipelines at other organizations. The workload optimizer jointly optimizes the combination of base data replication strategy, logical execution plan for each query, and the choice of which data center each task in the execution plan for each query is scheduled to run on.

WANalytics makes these decisions using a split, two-level optimization approach, in which (1) some choices, such as join order, are initially locked down by a centralized SQL query planner (a customized version we built of Apache Calcite[1]) using simple column-level statistics, and then (2) the remaining choices, such as distributed execution and data replication, are jointly optimized by an Integer Program formulation, based on measurements collected at runtime during query execution. [17] motivates this choice of architecture, and argues that it is an attractive point in the tradeoff between the complexity of the optimization process and the quality of the solution it generates.

In some cases collecting the measurements needed for phase (2) may require modifying query execution, using a mode we call *pseudo-distributed execution*.

2.2 Pseudo-distributed execution

Consider a single-table database with a workload consisting of one query performing an aggregation on the table. Suppose the system currently operates in a centralized configuration, in which all the data are copied to a single central data center where the aggregation query is then run. Now suppose we wish to evaluate the effect of moving to a distributed deployment in which data are left partitioned across data centers, and the query is executed by obtaining partial aggregation results from each data center which are then combined centrally.

To estimate the cost of distributed execution we would need to know the size of the partially aggregated output that would result at each data center. When the query is run centralized this information is unavailable, since all data are combined during operator execution.

To measure data sizes (and thus determine the most beneficial distributed execution plan), WANalytics instead simulates a virtual data center topology in which each partition is housed in a separate logical data center. This is accomplished by rewriting queries and pushing down `WHERE` clauses to restrict their operation to specific partitions/data centers. Details are discussed in [17]. This approach is both completely general, capable of rewriting arbitrary SQL queries to trace their execution on any given database, and very precise, since it directly measures each data transfer instead of relying on heuristic estimation.

Pseudo-distributed execution can end up slowing down query execution (by up to 20% in our experiments), but it never increases the volume of data transferred between data centers, which is the sole metric we are concerned with.

2.3 Data transfer optimization

Consider a query on a sales-log table, asking for the average number of orders from each country over the past week. Suppose the query is run once every day. Unoptimized WANalytics would recompute the query from scratch each day, but this transfers 7× more data than necessary since only the past day's worth of data would have changed each day.

To reduce data transfers, WANalytics aggressively caches all intermediate results produced during query execution. Before the results for any sub-query are transferred between

[1]http://optiq.incubator.apache.org/

two data centers, the system first checks to see if the subquery results are cached. If they are, WANalytics only computes and sends the diff between the old and the new results.

Interestingly, caching helps not only when the same query is run repeatedly, but also when two different queries share the same sub-operation. For instance, the TPC-CH benchmark, one of the several synthetic benchmarks we evaluated [17], contains 6 different queries, all computing slightly different aggregates on top of the same data-intensive join, and caching reduces the cumulative data transfer for all these queries by 5.99×.

3. DEMONSTRATION

During the demonstration attendees will interact with a live geo-distributed deployment of WANalytics, with data stored across three data centers in North America, Europe and South-East Asia[2]. Attendees will be able to run SQL queries, visualize the reduction in data transfer costs from using geo-distributed execution, and explore how WANalytics' optimizer can adapt query plans in response to changing conditions.

The demonstration will proceed in two phases. In the first phase we will demonstrate example workflows (sets of queries) that showcase interesting features of our system. In the second phase attendees will be invited to interact with the system directly by writing and evaluating arbitrary SQL queries against the database.

3.1 Setup

The deployment will host a copy of the CH benchmark database [3]. The database will be populated with some initial data; attendees will also be able to have the system simulate OLTP transactions that update the database (increasing its total size) over the course of the demonstration.

Figure 2 shows the user interface that attendees will interact with. The interface provides three options:

- *Reset* the system to its initial state.
- *Update* the database by simulating a specified volume of OLTP transactions.
- *Run SQL queries* analyzing the data.

Whenever a query is submitted, WANalytics first generates an initial query plan using simple column-level histogram statistics. After the query is run once, the system will have collected enough detailed information (§2.2) to be able to produce an updated, optimized query plan, which will be used on subsequent runs of the query[3]. The log at the bottom of the UI provides a visualization of both plans. Note that WANalytics jointly optimizes the plans for all queries in the workload, and thus the optimal plan identified for a given query may change as the workload evolves and expands.

The dashboard on the top-right provides visualizations of both the current, real-time data transfer volumes along each network link, as well as a comparison of cumulative cost over time against the centralized baseline.

[2]We will also provide a simulated 3-data center setup, hosted locally on the demonstration machine, as a fallback in case of e.g. network connectivity problems at the conference site.

[3]Recall (§2) that WANalytics targets applications with a recurrent workload, consisting of queries that arrive repeatedly. Using a potentially sub-optimal plan for the very first run of any query is reasonable in this context.

3.2 Demo phase 1: Example workflows

In the first phase of the demonstration we will run through example workflows to showcase the interesting features of our system. Some candidate examples:

1. Intra-query caching

Consider a query asking for the total sales volume on each day:

```
Q: SELECT date, SUM(sales_price) AS total_sales
   FROM orders
   GROUP BY date
```

The first time Q is run, the system will need to retrieve daily sales volumes over all historical data from each data partition. But subsequent reruns of Q, after OLTP updates logging new orders, will be much less data-intensive — the caching mechanism WANalytics uses (§2.3) will ensure that only the new information since the last run is retrieved.

2. Inter-query caching

Consider the following two queries, run one after the other:

```
Q1: SELECT supplier_country,
           SUM(sales_price) AS total_sales
    FROM orders JOIN suppliers
    USING supplier_no
    GROUP BY supplier_country

Q2: SELECT supplier_category,
           SUM(sales_price) AS total_sales
    FROM orders JOIN suppliers
    USING supplier_no
    GROUP BY supplier_category
```

Both queries compute slightly different projections on top of the same (data-intensive) join. Since WANalytics caches all intermediate results produced during query computation (§2.3), it reduces data transfer volume almost by half by avoiding processing the join from scratch twice.

3. Single-query optimization

Consider the query

```
Q: SELECT customer_name, SUM(order_value)
   FROM customers JOIN orders
   USING customer_id
   WHERE orders.order_date BETWEEN
         '2001-01-01' AND '2001-12-31'
   GROUP BY customer_name
```

asking for historical sales volumes. Since the `customers` table keeps growing over time, as new customers keep being added, only a fairly small fraction of the `customers` table will have any matching entries in the orders table for the specified date range. A semi-join strategy, starting out by shipping out the list of distinct `customer_id`s who placed an order in the date range, turns out to be optimal in this scenario. However, the simple histogram heuristics that WANalytics' initial query planner uses [17] can cause it to overestimate the join cardinality and suggest a hash join instead. However, once statistics are collected after an initial run using this strategy (§2.2), the optimizer would notice and switch to the optimal query plan.

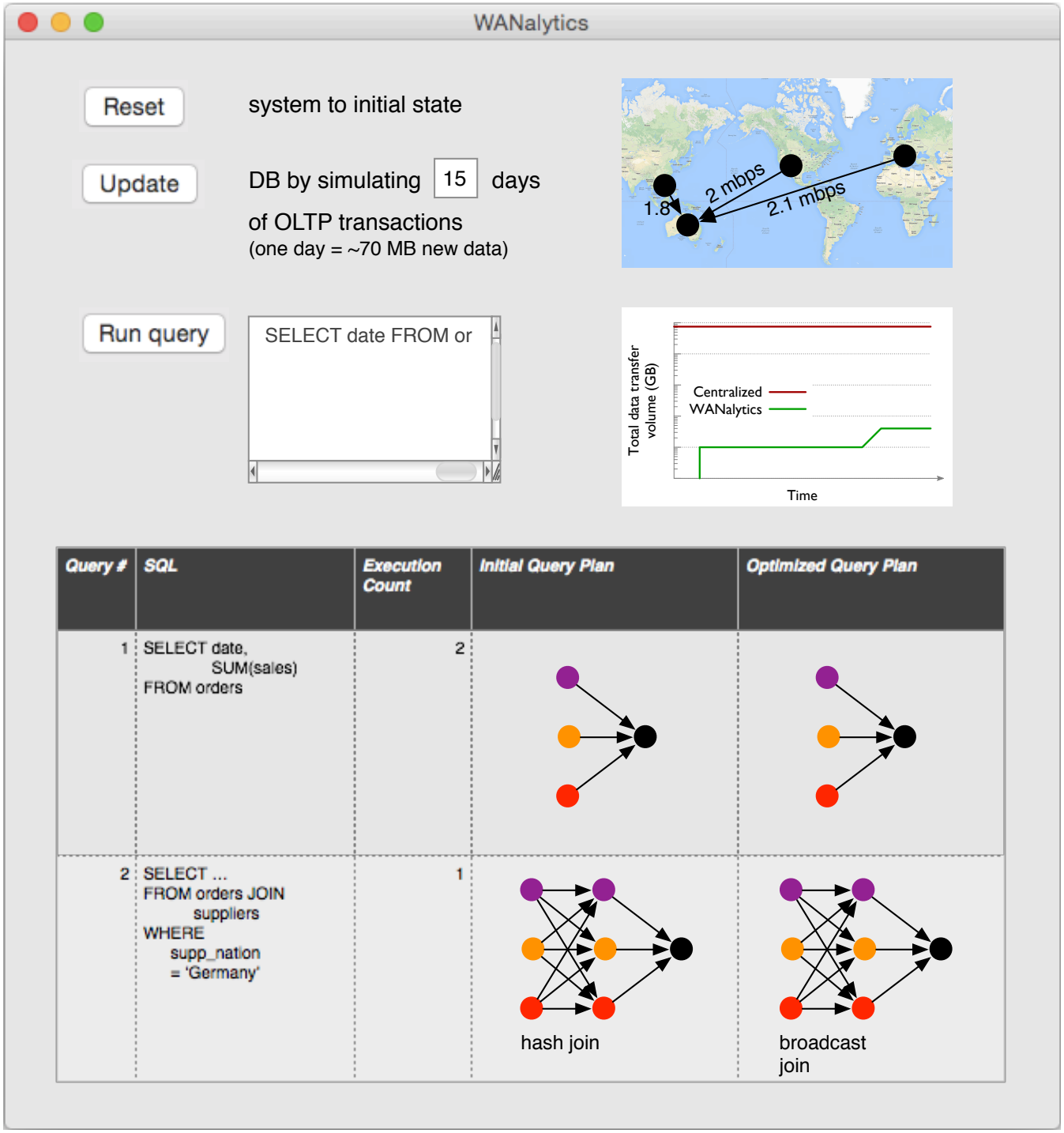

Figure 2: Demonstration GUI

4. Multi-query optimization

Consider the following two queries

```
Q1: SELECT item_id, COUNT(supplier_id)
    FROM stock JOIN suppliers
    USING (item_id)
    WHERE stock.country = 'USA'
    GROUP BY item_id

Q2: SELECT supplier_country, item_id, SUM(sale_price)
    FROM lineitems JOIN suppliers
    USING (supplier_id, item_id)
    GROUP BY supplier_country, item_id
```

The (filtered) `stock` table is much smaller than the `suppliers` table, which is in turn much smaller than the `lineitems` table. If `Q1` is the only query in the workload, WANalytics would process the join in `Q1` by filtering the `stock` table and broadcasting the results to each of the larger `suppliers` table's partitions. But once `Q2` is added to the workload, WANalytics notices that broadcasting the `suppliers` table instead for `Q1` would allow it to take advantange of cross-query caching, as in example 2, and amends the query plan for `Q1` accordingly.

3.3 Demo phase 2: Arbitrary user-provided workflows

In the second phase of the demonstration attendees will be invited to interact directly with the system, and explore how WANalytics processes and optimizes arbitrary SQL workloads they construct.

4. RELATED WORK

We now briefly discuss some directly relevant contributions from past work; we survey related work in greater detail in [16] and [17].

The WABD problem is closely related to literature in distributed and parallel databases [9, 13, 6], but our focus on analytics instead of transactions, the scale of the problem we target, our attention to bandwidth, and our exploration of more general computational models (with workflows of arbitrary tasks [16]) all render our problem significantly different.

Hive [15], Pig [12], Impala [1], Shark [18] and similar scale-out systems support analytics on continuously updated data, but are designed for single-cluster operation. PNUTS/Sherpa [4] does support geographically dispersed data but lays out data to reduce latency and not to minimize analytics cost. Spanner [5] supports OLTP workloads on geo-distributed data. Mesa [8] and Riak [2] replicate data across data centers for fault-tolerance but do not address distributed analytics execution.

Sensor networks [11] share a similar problem of minimizing the use of expensive network bandwidth, but not our massive scale or the breadth of our computational model. Stream-processing databases process long-standing queries on dispersed data, but the focus of work in this area has been on relatively simple computational models. A representative recent example, JetStream [14], targets a limited computational model centered around aggregation queries and does not, for instance, support joins.

5. CONCLUSION

Persistent growth in data volumes and rising interest in sovereignty regulation are rendering the current centralized approaches to geo-scale analytics increasingly untenable. We propose an alternative geo-distributed execution approach that is much better adapted to these requirements. Our system, WANalytics, incorporates several novel techniques inspired by revisiting classical database problems from a networking perspective. This demonstration showcases WANalytics' capabilities by allowing attendees to interactively construct and submit arbitrary SQL workloads on a live geo-scale multi-data center WANalytics deployment.

Acknowledgements

We gratefully acknowledge the support of an Alfred P. Sloan Research Fellowship.

6. REFERENCES

[1] Cloudera Impala. `http://bit.ly/1eRUDeA`.

[2] Greenplum. `http://basho.com/riak/`.

[3] R. Cole et al. The mixed workload CH-benCHmark. In *DBTest '11*, DBTest '11, pages 8:1–8:6, New York, NY, USA, 2011. ACM.

[4] B. F. Cooper, R. Ramakrishnan, U. Srivastava, A. Silberstein, P. Bohannon, H.-A. Jacobsen, N. Puz, D. Weaver, and R. Yerneni. Pnuts: Yahoo!'s hosted data serving platform. *Proc. VLDB Endow.*, 1(2):1277–1288, Aug. 2008.

[5] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, W. Hsieh, S. Kanthak, E. Kogan, H. Li, A. Lloyd, S. Melnik, D. Mwaura, D. Nagle, S. Quinlan, R. Rao, L. Rolig, Y. Saito, M. Szymaniak, C. Taylor, R. Wang, and D. Woodford. Spanner: Google's globally-distributed database. In *10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, pages 261–264, Hollywood, CA, Oct. 2012. USENIX Association.

[6] D. DeWitt and J. Gray. Parallel database systems: The future of high performance database systems. *Communications of the ACM*, 35(6):85–98, June 1992.

[7] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel. The cost of a Cloud: Research problems in data center networks. *ACM CCR*, 39(1), 2008.

[8] A. Gupta, F. Yang, J. Govig, A. Kirsch, K. Chan, K. Lai, S. Wu, S. Dhoot, A. Kumar, A. Agiwal, S. Bhansali, M. Hong, J. Cameron, M. Siddiqi, D. Jones, J. Shute, A. Gubarev, S. Venkataraman, and D. Agrawal. Mesa: Geo-replicated, near real-time, scalable data warehousing. In *VLDB*, 2014.

[9] D. Kossmann. The state of the art in distributed query processing. *ACM Computing Surveys*, 32(4):422–469, Dec. 2000.

[10] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez. Inter-datacenter bulk transfers with Netstitcher. In *SIGCOMM 2011*.

[11] S. Madden. Database abstractions for managing sensor network data. *Proc. of the IEEE*, 98(11):1879–1886, 2010.

[12] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. Pig Latin: a not-so-foreign language for data processing. In *SIGMOD 2008*.

[13] M. T. Özsu and P. Valduriez. *Principles of distributed database systems*. Springer, 2011.

[14] A. Rabkin, M. Arye, S. Sen, V. S. Pai, and M. J. Freedman. Aggregation and degradation in jetstream: Streaming analytics in the wide area. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, pages 275–288, Seattle, WA, Apr. 2014. USENIX Association.

[15] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, S. Antony, H. Liu, and R. Murthy. Hive - a petabyte scale data warehouse using hadoop. In *ICDE 2010*.

[16] A. Vulimiri, C. Curino, B. Godfrey, K. Karanasos, and G. Varghese. WANalytics: Analytics for a geo-distributed data-intensive world. *CIDR*, 2015.

[17] A. Vulimiri, C. Curino, B. Godfrey, J. Padhye, and G. Varghese. Global analytics in the face of bandwidth and regulatory constraints. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, Oakland, CA, May 2015. USENIX Association.

[18] R. S. Xin, J. Rosen, M. Zaharia, M. J. Franklin, S. Shenker, and I. Stoica. Shark: Sql and rich analytics at scale. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 13–24, New York, NY, USA, 2013. ACM.