2015 IEEE International Conference on Cloud Engineering

Towards Optimizing Wide-Area Streaming Analytics

Benjamin Heintz
University of Minnesota
heintz@cs.umn.edu

Abhishek Chandra
University of Minnesota
chandra@cs.umn.edu

Ramesh K. Sitaraman
UMass, Amherst & Akamai Tech
ramesh@cs.umass.edu

***Abstract*—Modern analytics services require the analysis of large quantities of data derived from disparate geo-distributed sources. Further, the analytics requirements can be complex, with many applications requiring a combination of both real-time and historical analysis, resulting in complex tradeoffs between cost, performance, and information quality. While the traditional approach to analytics processing is to send all the data to a dedicated centralized location, an alternative approach would be to push all computing to the edge for in-situ processing. We argue that neither approach is optimal for modern analytics requirements. Instead, we examine complex tradeoffs driven by a large number of factors such as application, data, and resource characteristics. We present an empirical study using PlanetLab experiments with beacon data from Akamai's download analytics service. We explore key tradeoffs and their implications for the design of next-generation scalable wide-area analytics.**

I. Introduction

Data analytics is undergoing a revolution: both the volume and diversity of analytics data are increasing at a rapid rate. Further, across a large number of domains including web analytics, social analytics, and energy analytics, data is often derived from disparate sources that include users, devices, sensors, and servers located around the globe. The application requirements for processing the data can be equally complex. While some application queries require a real-time analysis of current events, others require a historical analysis of data over longer periods of time. But commonly, many application queries require a combination of both real-time and historical analysis, resulting in complex tradeoffs between cost (e.g., WAN traffic, storage, energy), performance (e.g., query latency, throughput), and information quality (e.g., staleness, accuracy, completeness). In addition to data sources and application requirements, the compute and storage resources that are available for analytics processing are themselves often widely distributed across multiple data centers across the globe. Further, the infrastructure available for analytics processing is often a shared resource whose availability can vary over time.

The traditional approach to analytics processing is the *centralized cloud model* where the data streams are sent to a dedicated centralized location with the hardware, network, and software capabilities for analytics processing. The centralized location might offer OLAP (online analytics processing), DBMS, or MapReduce-like functionalities. An alternative approach is the *decentralized cloud model* that moves computation to the data by pushing the analytics processing completely to the edges. In this paper, we argue that neither approach is optimal for modern analytics requirements. Instead, there are complex tradeoffs driven by a large number of factors that must be taken into account to achieve the desired metrics of cost, performance, and information quality. We present an empirical study using PlanetLab experiments with Akamai analytics data to illustrate some of the challenges and tradeoffs, and offer guidelines for approaches towards optimizing wide-area streaming analytics.

A. Requirements of modern analytics services

There are a number of domains where complex analytics services are required. A canonical example is a *web analytics service* provided by a content delivery network (CDN) [1]. A large CDN such as Akamai serves trillions of user requests per day for web and media content on behalf of content providers such as CNN or Facebook. A content provider using a CDN for delivering content to its users requires analytics and business intelligence about its users and content; for instance, which users are watching which content, what are the top trends in content popularity, etc. *Social network analytics* for services such as Facebook or Twitter involves data streams from billions of users that must be analyzed for trends in near real-time. Another example is *energy analytics*, which involves voluminous data—i.e., temperature and energy usage data—derived from sensors deployed in smart homes. The sensor data must be analyzed for energy trends and potential energy cost savings and presented to the homeowner without compromising privacy requirements [2]. Another emerging area is *security analytics* that monitors traffic at edge servers of a CDN to detect DDoS attacks and security exploits. Such a service must detect bot attacks, cross-site scripting attacks, SQL injection and other common exploits in near real-time using both real-time data and voluminous historical data of users accessing the edge servers of the CDN.

To better understand the requirements of such analytics services, consider a web analytics service provided by a CDN. In such a service, the data about what content users are viewing, what actions the users are performing (e.g., playing or rewinding a video, interacting with a web page) are sent from the users' browsers and media players to the "nearest" edge server of the CDN in the form of "beacons". Thus, beacon data is collected in over a thousand data center locations across the globe. Further, the edge servers themselves write detailed logs describing each user request and server response. These multiple geographically diverse data sources must then be analyzed to answer queries from content providers about their users and content, such as what content is being watched in which geographic location, what the trends are in content popularity, and if their users are experiencing good performance. For some queries such as content trends, the content provider might be willing to tolerate a small degree of inaccuracy (i.e., lower information quality) while insisting on (near) real-time, up-to-the-minute results. For other queries, such as attributions for an

978-1-4799-8218-9/15 $31.00 © 2015 IEEE
DOI 10.1109/IC2E.2015.53

ad campaign that impact revenue sharing, the content provider might insist on 100% accuracy, though the results need not be available in real-time.

An analytics query could involve both real-time streaming analysis as well as historical analysis. For instance, to answer the question of whether users are currently experiencing slower web downloads, one must compute the current download performance in near-real-time and compare the results with historical information for the current mix of users, web pages, and time of day. The resource requirements of the queries could also vary significantly. While some queries require relatively small amounts of raw data to be transfered from the thousands of edge locations to a centralized location for processing, other queries require voluminous amounts of detailed data that would be wasteful to transfer across the wide-area network (e.g., debugging queries involving detailed server logs).

B. Challenges in optimizing analytics services

There are a number of challenges and tradeoffs in architecting an analytics service that optimizes the three key considerations: *performance*, *cost*, and *information quality*. Ideally, such a service should provide information of sufficient quality, and perform in accordance with application requirements, while minimizing the cost of operations. Some key questions and tradeoffs are below.

Where to store and process the data? The data originates in diverse edge locations. Should the analytics computations be performed solely in those edge locations, with only the final results aggregated at the central location? Alternatively, should the data in its entirety be transmitted to the central location with all the processing happening at that location? There are likely no simple answers; both extremes are suboptimal. The best solution depends on a number of factors such as the availability of compute and storage resources, WAN bandwidth capacity and cost, and data privacy requirements. For some applications (such as energy analytics), there may be significantly more computing capacity available at the center, while for others (such as a CDN), sufficient computation and storage would be available at the edges, though its availability may vary over time. Significant network resources are required to backhaul data from edge locations to a central location; network capacity may vary across different edge locations, and the bandwidth prices that are charged on the 95^{th} percentile of traffic may vary throughout the day [3]. Finally, in case of energy analytics, transmitting detailed sensor data from each home to a public cloud might raise serious privacy concerns, requiring at least some computation to be performed at the edge [2].

When to process the data? In the one extreme, the computations can be performed as soon as the data is made available. In the other, data is processed only when a query that accesses the data is performed. There is a continuum of possibilities between the two extremes where some data is preprocessed while the rest is processed "on demand" at query time. The optimal decision depends on application requirements and the query characteristics. For instance, queries that evaluate the top trends in content access will need to be preprocessed as soon as the data is available at the edge locations, whereas a "debugging" query that investigates ongoing performance issues will rely on detailed data archived at the edge locations that are processed only on demand.

What information quality is sufficient? The quality of the results provided by an analytics service is often dictated by the specific way in which those results will be used by the consumer of the service. One measure of information quality is the staleness of the query results; i.e., the delay between input data becoming available *to* the analytics service, and the output results becoming available *from* the analytics service. Different queries may find different values of staleness to be acceptable. For instance, a query for the latest traffic, user, and content trends may require results to be as fresh as possible, while queries used for debugging purposes often work on older "stale" data. There are natural tradeoffs between the staleness and bandwidth costs, since requiring the data to be fresh often means transmitting unaggregated data as soon as possible over WAN links.

Another measure of quality is the result accuracy. It is often not possible to compute the exact query results, especially when there is a high freshness requirement. For instance, computing the order statistics of a data stream "on the fly" in real-time requires approximate solutions rather than exact ones [4]. Further, computing the results of some queries exactly might be infeasible due to resource or cost constraints, requiring methods for gracefully degrading accuracy [5], [6].

II. Empirical Evaluation of Tradeoffs

The questions posed in Section I are challenging to answer in a wide-area setting. In this section, we illustrate the challenges by studying an important class of analytics queries called *grouped aggregation*. Grouped aggregation is used to combine and summarize large quantities of data from one or more data streams. As a result, it is provided as a key operator in most data analytics frameworks, such as the Reduce operation in MapReduce, GroupBy in SQL and LINQ, etc. Here, we consider windowed grouped aggregation in the streaming analytics context, where queries are performed on data produced with finite specified time window, say the past minute or the past hour.

Grouped aggregation involves constructing data cubes [7] that are then used to compute the aggregate query results. A data cube groups data by *dimensions* and aggregates the set of *measures* within each group. In our evaluation, we consider the question of how frequently the edges should transmit their local aggregate results to the center. We define the interval between these transmissions as the *aggregation interval*. Concretely, to implement an aggregation interval of t seconds, an edge computes aggregates locally for t seconds, then asynchronously transmits results to the center while beginning aggregating for the next t seconds, and so on. We study the impact of the chosen edge aggregation interval on two metrics: *staleness*, which is a key measure of information quality, and *WAN traffic*, which is a key measure of cost. Staleness is defined as the time interval δ such that the aggregate results for the window from time t_0 to time t_1 first become available at the center at time $t_1 + \delta$. The WAN traffic is measured in terms of the number of aggregate records that are sent over the WAN.

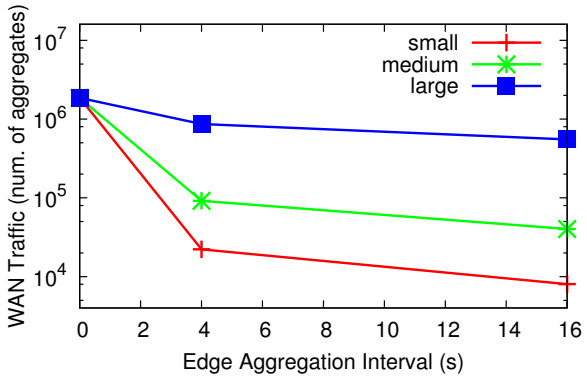

Fig. 1. When the edge aggregation interval increases, more data is aggregated on the edge, reducing the volume of data sent to the center. Note the vertical axis uses a logarithmic scale.

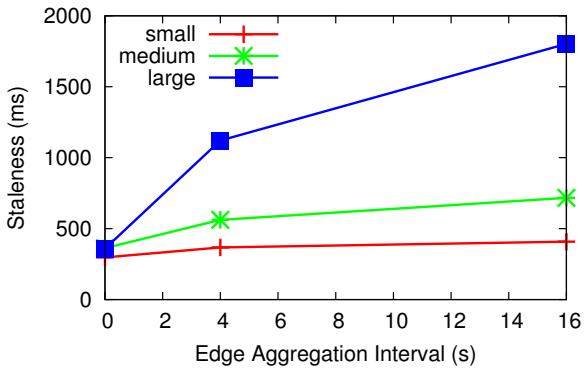

Fig. 2. When the edge aggregation interval increases, a larger burst of records must be sent to the center at the very end of the window, and the resulting transmission delay causes higher staleness.

A. Experimental Methodology

To simulate the tradeoffs in a typical analytics service, we collected the beacon logs from Akamai's download analytics service over the month of December, 2010. The beacons contain information reported by Akamai's download manager that runs on the users' desktop or mobile devices. The download manager is widely used by users around the globe for downloading software, games, and music. Each download results in one or more beacons, each containing information pertaining to the download, being sent to an edge server. The beacons contain anonymized information about users downloading content from edge servers. In particular, the beacons contain time of access, information about the accessed content such as url, size, number of bytes downloaded, information about the user including ip, network, and geography, and information about the edge server including its network and geography. We replay these beacon logs from a set of PlanetLab machines to simulate the data collection and processing aspects of the download analytics service. We use four nodes at Texas A&M for edge computation, and each is responsible for replaying a geographic partition of the input data. The final centralized computation is performed at one node in Princeton. At the time we ran our tests, the bandwidth from each of these edge nodes at Texas to the central node at Princeton averaged 21.4 Mbps.

B. Impact of query characteristics

The characteristics of the query dictate the *cube size*; that is, the number of distinct combinations of values possible for the set of grouping dimensions. We study three queries that we call `small`, `medium`, and `large`. The `small` query requires a cube of two dimensions: the content provider id and the user's last mile bandwidth classified into four buckets. The `medium` query requires a cube of three dimensions: the content provider id, user's last mile bandwidth, and the user's country code. The `large` query consists of three dimensions: the content provider id, the user's country code, and the url accessed. Note that the last dimension—url—can take on hundreds of thousands of distinct values, resulting in a very large cube size. Figures 1 and 2 show the impact on network traffic and staleness respectively for these three queries as the edge aggregation interval is increased. Increasing the edge aggregation interval leads to more data records being aggregated at the edge. Thus, it reduces the amount of data that must be sent from the edge to the central location, in turn reducing the WAN traffic. However, increasing the edge aggregation interval also leads to a larger batch of records due for network transmission at the very end of the window, resulting in higher network transmission delays and in turn higher staleness. We find that the extent of the tradeoff is markedly different in the three cases: an edge aggregation interval of 16 seconds reduces WAN traffic—relative to a baseline with no edge aggregation—by more than 99% for query `small`, but only by 70% for query `large`. At the same time, the staleness for query `small` increases by 37% while the staleness for query `large` increases by 4.0x. This is because, for query `large`, as we increase the edge aggregation interval, we are sacrificing communication-computation pipelining without gaining much in terms of actual data reduction. On the other hand, for query `small`, deferring communication provides ample opportunity for reducing data volumes at the edge.

These results have several implications. First of all, contrary to conventional wisdom, aggregation at the edge may not always be beneficial. In some cases, it may be *better* to simply send all data to the center. At the same time, the optimal decision of where and when to aggregate can be made if the system is aware of or can "learn" the query characteristics. Second, achieving the desired quality of information (staleness) may be much more costly for certain query types than others. In particular, this can result in interesting optimization opportunities in the presence of multiple queries, e.g., allocating fewer resources to certain queries against others depending on where each of them lies on the cost-quality curve.

C. Impact of data source characteristics

The data sources can be characterized along several axes such as key distribution, data volume, and skew of data across different locations. The latter is particularly likely to occur in geographically distributed settings. As an example, most website visits or keyword searches are influenced by location of the user (e.g., most users in Japan may access Japanese news sites). In Figures 3 and 4, we plot the WAN traffic

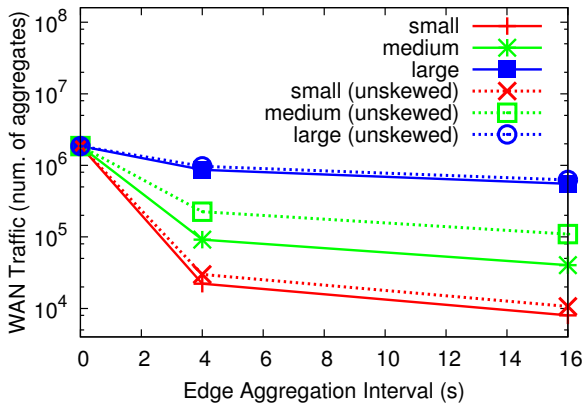

Fig. 3. Geographic skew inherent in the dimension values results in greater edge aggregation, resulting in less data being sent over the wide-area network.

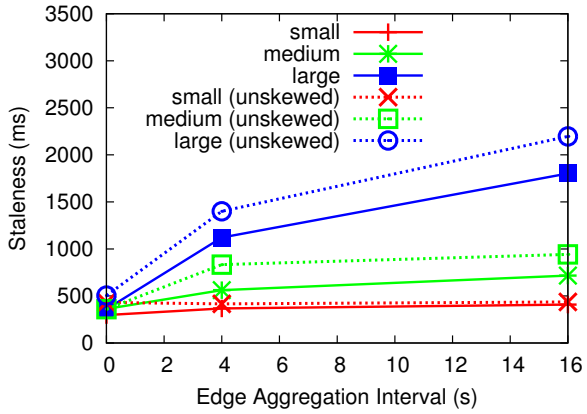

Fig. 4. Geographic skew inherent in the dimension values results in greater aggregation and data reduction. The data reduction leads to smaller compute and communication delays, resulting in less stale results.

and staleness respectively both with and without location-dependent data skew. The plots corresponding to data skew are based on the actual data skew inherent in the original Akamai beacon data. The unskewed results correspond to removing the location-dependent skew from the data by randomizing the source location of each record (while still keeping the relative data volumes the same as before).

The figures show that both the WAN traffic as well as the staleness become *worse* without this skew. This phenomenon can be attributed to the fact that location-dependent skew leads to higher concentration of data with a given set of dimension values in each edge location, providing more opportunities for aggregation at the edge. This leads to lower WAN utilization as the greater aggregation reduces the data that needs to be transmitted from the edge to the center. Further, more data reduction due to greater aggregation also reduces the computation and communication time, leading to smaller delays and fresher (i.e., less stale) query results.

D. Impact of the characteristics of the analytics infrastructure

The compute and storage resources available for analytics processing also influence the choice of the edge query aggregation interval and cost-quality tradeoffs. Infrastructure characteristics include network, compute, and storage capacities at the edge and center locations. Intuitively, higher resource capacity (e.g., higher compute capacity or network bandwidth) implies higher quality (low staleness) at lower cost (low WAN traffic). We illustrate this tradeoff empirically by comparing different network capacities between the edges and the center. We ran two experiments that are identical in all respects except the location of the center node. First, the three queries (`small`, `medium`, and `large`) are run across a WAN with the center at a PlanetLab node in Princeton, NJ and the edges at a PlanetLab nodes in Texas. To emulate more plentiful network resources, we repeat the same experiment with both the center and the edge nodes running at Texas A&M, and communicating over the local area network (LAN).

Figure 5 shows results for both experiments. It can be seen that information quality is improved (i.e., staleness is reduced) when network resources are more plentiful. The reason is that the edge-to-center communication is faster when there is more bandwidth from the edges to the center.

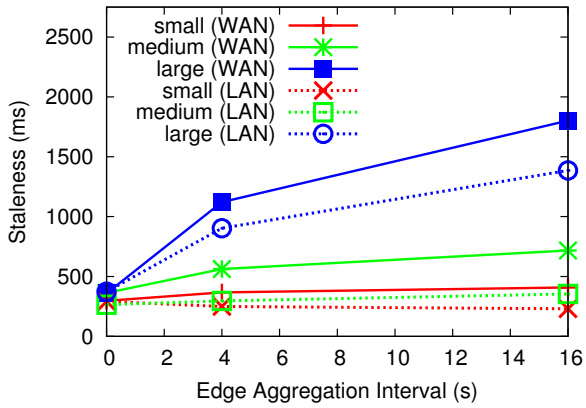

Fig. 5. For a given query, staleness is reduced when network resources are more plentiful.

III. Related Work

A. Streaming systems

Numerous streaming systems have been developed both in academia and in industry [8], [9], [10], [11], [12], [13], [14]. While each focuses on its own unique engineering requirements and research challenges, these systems share many useful and time-tested ideas upon which any new streaming analytics system should build. They do not, however, address all of the challenges that we've described here, such as determining the appropriate placement of computation, or how to effectively trade off between cost and quality.

B. Grouped aggregation

Grouped aggregation has broad applicability in many analytics applications. For example, the MapReduce [15] programming model is essentially grouped aggregation composed

with an initial map transformation. Muppet [16] extends this model into the streaming context. A Data Cube [7] represents the set of aggregations over all possible groups of dimensions in a dataset; the Roll-Up operation corresponds directly to aggregation over groups. This pattern of computation, though seemingly restrictive at first, has surprisingly broad applicability. For example, Twitter has developed systems and libraries aimed at streaming aggregation. In particular, Summingbird [17] provides a higher-level programming API for writing aggregate computation to run on Storm or Hadoop, and the Algebird [18] library generalizes over data types that have the algebraic structure of monoids or semigroups, and can therefore be easily aggregated.

C. When to compute

LazyBase [19] allows users to trade off between query latency and result freshness by extracting results from various stages of a processing pipeline. This essentially answers the question of when to *return* results, but it does not directly address the question of when to *compute* them. Several systems such CIEL [20], and Spark [21] use lazy evaluation, deferring computation until results are requested. At the other extreme, Twitter's Rainbird [22] takes a fully eager approach, updating the counts for all impacted groups when a data item arrives. In a streaming context where wide-area bandwidth is expensive or constrained, and staleness needs to be bounded, neither of these extreme approaches is always the right answer.

The question of whether to compute results eagerly when new data items arrive, or lazily when queries are issued, is well known in database settings as the view selection problem, and approximate solutions have been the focus of a large body of research. In the setting of data cubes in particular, research has focused on both offline techniques [23], which often make too many simplifying assumptions to be directly implemented in real systems, as well as dynamic approaches [24], [25] that make selection decisions based on runtime conditions.

D. Where to compute

Little work on streaming computation has focused on wide-area deployments, or the associated questions such as where to place computation. Pietzuch et al. [26] optimize operator placement in geo-distributed settings to balance between system-level bandwidth usage and latency. Hwang et al. [27] rely on replication across the wide area in order to achieve fault tolerance and reduce straggler effects. JetStream [5] focuses on wide-area deployments, but attempts to place as much computation as possible at the edge, which our experiments have shown is not always an appropriate policy.

E. What quality level

The OVIS(θ) algorithm [28] trades off between performance and freshness in selecting which dynamic web content to materialize and cache. This extends the previous literature on the view selection problem by beginning to incorporate quality of results as an important objective. More recently, systems such as BlinkDB [6] and JetStream [5] have provided mechanisms to trade off between accuracy and response time, and between accuracy and bandwidth utilization, respectively. We aim to build on these contributions by exploring additional tradeoffs such as that between staleness and traffic as we have described in this paper. Keeton et al. [29] provide a concise discussion of several important research challenges in the area of information quality.

IV. Conclusion

In this paper, we examined the problem of optimizing modern analytics services that process large quantities of geo-distributed data. We presented empirical results with grouped, windowed aggregation on PlanetLab using Akamai log data, and highlighted the complexity of tradeoffs that we show are driven by several factors such as query, data, and resource characteristics. Further, we see that neither a purely centralized nor a purely decentralized approach to computation is optimal in general. The insights gained from our results suggest several interesting research directions. One area of interest is the automatic placement of data and computation based on an understanding of query, data, and resource characteristics. Another area of interest is to identify the right set of compute operators to place at the edge vs. at the center. While operators such as filtering are obvious candidates for edge placement, we have shown that aggregation at the edge is not always beneficial. Further, these operators need to be combined with approximation techniques such as sampling and sketching to achieve desired cost-quality tradeoffs. Finally, the issue of efficiently supporting multiple concurrent queries is important, given the potential conflicts as well as sharing opportunities.

References

[1] E. Nygren, R. Sitaraman, and J. Sun, "The Akamai Network: A platform for high-performance Internet applications," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 3, pp. 2–19, 2010.

[2] R. P. Singh, S. Keshav, and T. Brecht, "A cloud-based consumer-centric architecture for energy data analytics," in *Proc. of ACM e-Energy*, 2013, pp. 63–74.

[3] M. Adler, R. K. Sitaraman, and H. Venkataramani, "Algorithms for optimizing the bandwidth cost of content delivery," *Computer Networks*, vol. 55, no. 18, pp. 4007–4020, 2011.

[4] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems," in *Proc. of PODS*, 2002, pp. 1–16.

[5] 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 *Proc. of NSDI*, 2014, pp. 275–288.

[6] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica, "BlinkDB: queries with bounded errors and bounded response times on very large data," in *Proc. of EuroSys*, 2013, pp. 29–42.

[7] J. Gray *et al.*, "Data Cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals," *Data Min. Knowl. Discov.*, vol. 1, no. 1, pp. 29–53, Jan. 1997.

[8] "Amazon Kinesis," http://aws.amazon.com/kinesis/, 2014.

[9] "S4: Distributed Stream Computing Platform," http://incubator.apache.org/s4/, 2014.

[10] "Storm, distributed and fault-tolerant realtime computation," http://storm.incubator.apache.org/, 2014.

[11] T. Akidau *et al.*, "MillWheel: Fault-tolerant stream processing at internet scale," *Proc. of the VLDB Endow.*, vol. 6, no. 11, pp. 1033–1044, Aug. 2013.

[12] S. Chandrasekaran *et al.*, "TelegraphCQ: Continuous dataflow processing for an uncertain world," in *Proc. of CIDR*, 2003.

[13] Z. Qian *et al.*, "TimeStream: reliable stream computation in the cloud," in *Proc. of EuroSys*, 2013, pp. 1–14.

[14] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, and I. Stoica, "Discretized streams: Fault-tolerant streaming computation at scale," in *Proc. of SOSP*, 2013, pp. 423–438.

[15] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *Proc. of OSDI*, 2004, pp. 137–150.

[16] W. Lam, L. Liu, S. Prasad, A. Rajaraman, Z. Vacheri, and A. Doan, "Muppet: MapReduce-style processing of fast data," *Proc. of the VLDB Endow.*, vol. 5, no. 12, pp. 1814–1825, Aug. 2012.

[17] O. Boykin, S. Ritchie, I. O'Connel, and J. Lin, "Summingbird: A framework for integrating batch and online MapReduce computations," in *Proc. of VLDB*, vol. 7, no. 13, 2014, pp. 1441–1451.

[18] "twitter/algebird - GitHub," https://github.com/twitter/algebird, 2014.

[19] J. Cipar, G. Ganger, K. Keeton, C. B. Morrey, III, C. A. Soules, and A. Veitch, "LazyBase: trading freshness for performance in a scalable database," in *Proc. of EuroSys*, 2012, pp. 169–182.

[20] D. G. Murray, M. Schwarzkopf, C. Smowton, S. Smith, A. Madhavapeddy, and S. Hand, "CIEL: a universal execution engine for distributed data-flow computing," in *Proc. of NSDI*, 2011, pp. 113–126.

[21] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing," in *Proc. of NSDI*, 2012.

[22] "Rainbird: Realtime Analytics at Twitter (Strata 2011)," http://www.slideshare.net/kevinweil/rainbird-realtime-analytics-at-twitter-strata-2011, 2011.

[23] K. Morfonios, S. Konakas, Y. Ioannidis, and N. Kotsis, "ROLAP implementations of the data cube," *ACM Comput. Surv.*, vol. 39, no. 4, Nov. 2007.

[24] J. Han, Y. Chen, G. Dong, J. Pei, B. Wah, J. Wang, and Y. Cai, "Stream Cube: An architecture for multi-dimensional analysis of data streams," *Distributed and Parallel Databases*, vol. 18, no. 2, pp. 173–197, 2005.

[25] Y. Kotidis and N. Roussopoulos, "DynaMat: a dynamic view management system for data warehouses," in *Proc. of SIGMOD*, 1999, pp. 371–382.

[26] P. Pietzuch, J. Ledlie, J. Shneidman, M. Roussopoulos, M. Welsh, and M. Seltzer, "Network-aware operator placement for stream-processing systems," in *Proc. of ICDE*, 2006.

[27] J.-H. Hwang, U. Cetintemel, and S. Zdonik, "Fast and highly-available stream processing over wide area networks," in *Proc. of ICDE*, 2008, pp. 804–813.

[28] A. Labrinidis and N. Roussopoulos, "Exploring the tradeoff between performance and data freshness in database-driven web servers," *The VLDB Journal*, vol. 13, no. 3, pp. 240–255, Sep. 2004.

[29] K. Keeton, P. Mehra, and J. Wilkes, "Do you know your IQ?: a research agenda for information quality in systems," *SIGMETRICS Perform. Eval. Rev.*, vol. 37, no. 3, pp. 26–31, Jan. 2010.