Wide-Area Streaming Analytics: Distributing the Data Cube

Benjamin Heintz[†], Abhishek Chandra[†], and Ramesh K. Sitaraman[††]

[†]University of Minnesota
[††]University of Massachusetts & Akamai Technologies

1 Problem

To date, much research in data-intensive computing has focused on batch computation. Increasingly, however, it is necessary to derive knowledge from big data *streams*. As a motivating example, consider a content delivery network (CDN) such as Akamai [4], comprising thousands of servers in hundreds of globally distributed locations. Each of these servers produces a stream of log data, recording for example every user it serves, along with each video stream they access, when they play and pause streams, and more. Each server also records network- and system-level data such as TCP connection statistics. In aggregate, the servers produce billions of lines of log data from over a thousand locations daily.

In order to extract knowledge from this stream—whether to monitor system performance, detect network intrusions, or analyze audience behavior—users need system support for near-real-time queries. In particular, it is useful to maintain a view of aggregated measures such as average session duration over all video streams, or total number of bytes downloaded by customer and content type. A useful abstraction for such aggregate queries is the *data cube* [1]. Our interest in geographically distributed stream data naturally leads to the question of *where* to perform the computation required to maintain a data cube over such streams.

Further, a data cube describing a d-dimensional stream comprises 2^d *cuboids*, each grouping by a subset of the d dimensions. In reality, maintaining all cuboids is infeasible. It is, however, possible to precompute and incrementally maintain a set of cuboids in order to lower query response time or reduce resource utilization.

SoCC'13, Oct 01-03 2013, Santa Clara, CA, USA.
ACM 978-1-4503-2428-1/13/10.
http://dx.doi.org/10.1145/2523616.2525963

Cuboids that are not eagerly precomputed must be lazily computed at query time by aggregating a precomputed cuboid or by scanning the raw data records. A second key question in our work is therefore *which* cuboids to precompute and incrementally maintain.

2 Challenges and Approach

The problem of selecting a set of precomputed cuboids to minimize query time under space or maintenance cost constraints has been shown to be NP-complete [2, 3] even with many simplifying assumptions. The focus of prior research has thus been on approximate solutions. The placement problem, however, has not received such attention; prior work has generally focused on centralized computation. The geo-distributed setting brings additional challenges, from network heterogeneity to time-varying resource conditions, as well as the higher volume of data that such environments can produce. Slow wide-area network links will lead to bottlenecks unless such factors are carefully taken into account. These questions of course interact; for example, the best placement of computation for two cuboids may depend on whether a common ancestor cuboid will be precomputed. Similarly, whether to precompute a cuboid c may depend on the placement of other precomputed cuboids from which c's descendants can be derived.

Recently, Rabkin et al. [5] have also considered geo-distributed streaming applications, but with a focus on issues of data quality and providing explicit facilities for restricting movement of sensitive data. Our focus is on questions of *where* to place computation and *what* to eagerly precompute versus lazily compute at query-time.

To address these questions, our approach combines modeling and system implementation. By modeling the cost of queries as a function of resource characteristics and precomputation decisions, we can explore heuristics that are better suited to geo-distributed settings. Implementing and deploying a system on a globally distributed testbed such as PlanetLab will allow us to assess our contributions in a realistic environment.

References

[1] J. Gray et al. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Min. Knowl. Discov.*, 1(1):29–53, Jan. 1997.

[2] V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. In *Proc. of SIGMOD '96*, pages 205–216, 1996.

[3] K. Morfonios, S. Konakas, Y. Ioannidis, and N. Kotsis. ROLAP implementations of the data cube. *ACM Comput. Surv.*, 39(4), Nov. 2007.

[4] E. Nygren, R. Sitaraman, and J. Sun. The akamai network: A platform for high-performance internet applications. *ACM SIGOPS Oper. Syst. Rev.*, 44(3):2–19, 2010.

[5] A. Rabkin, M. Arye, S. Sen, V. Pai, and M. J. Freedman. Making every bit count in wide-area analytics. In *Proc. of HotOS*, 2013.