Extensible Context-aware Stream Processing on the Cloud: Rationale and Challenges

Walid G. Aref [1]
Department of Computer Science, Purdue University, West Lafayette, Indiana
aref@cs.purdue.edu

1. Rationale and Challenges for Massive Data Stream Processing on the Cloud

The ubiquity of mobile devices, location services, and sensor pervasiveness, e.g., as in smart city initiatives, call for scalable computing platforms and massively parallel architectures to process the vast amounts of the generated streamed data. Cloud computing provides some of the features needed for these massive data streaming applications. For example, the dynamic allocation of resources on an as-needed basis addresses the variability in sensor and location data distributions over time. However, today's cloud computing platforms lack very important features that are necessary in order to support the massive amounts of data streams envisioned by the massive and ubiquitous dissemination of sensors and mobile devices of all sorts in smart-city-scale applications. To support massive data stream processing, cloud platforms need to be extended in the following directions:

1.1 Hybrid memory-based and disk-based processing: Cloud platforms are typically designed for data-intensive disk-based applications. For massive data streaming applications, disks need to be replaced by distributed main-memory buffers at all the various layers. For example, distributed file systems or distributed disk-based databases are to be replaced by a memory-based online data stream acquisition layer that continuously receives streamed inputs via network sockets and stores them into distributed memory buffers. Disk storage has to be eliminated from the various other layers of massively parallel systems. For example, in a Map/Reduce scenario [4], the communication layer between the mappers and the reducers also has to be memory-based. It is important to point out that even for massive data streaming applications; combined memory- and disk-based platforms have to be integrated to answer user queries. For example, moving objects in the form of spatiotemporal data streams that are processed in memory will also need access to a road network that is stored in the disk-based platform. A typical continuous query on the moving objects' data streams would also have to access the disk-based road-network data. Therefore, a hybrid stream- and disk-based parallel architecture is needed for these applications.

1.2 Dynamic rate-based load-balancing and multi-stream partitioning: Typically, to achieve load balancing, distributed files or tables are partitioned based on their sizes and on the computing power of each of the participating machines, i.e., a more powerful machine claims a bigger portion of the data to process. In massively parallel streaming, the equivalent is to perform rate-based partitioning, i.e., the data streams with high rates are to be split into multiple processing units to avoid buffer overflows and load shedding. Additionally, slow streams needs to be bundled together to avoid under-utilization of resources. Therefore, three load balancing operations are speculated for data streams, namely, split, bundle, and migrate.

1.3 Fault-tolerance and stream replication: In batch cloud systems, data is replicated multiple times to guarantee availability of disk-data despite of disk, machine, and rack failures [e.g., as in [3]). In a massive data streaming cloud, most likely, systems cannot afford to redundantly replicate streamed data multiple times as data is continuously arriving. Novel approaches have to be developed to achieve fault tolerance for massive streaming environments.

1.4 Continuous and progressive window-based processing: In contrast to a batch mode of operation, cloud platforms have to process streamed data continuously using some notion of system pulse or beat, where between two pulses; newly arriving streamed data that gets accumulated into the memory buffers during the previous pulse are consumed and processed while more data arrives. Because data is infinite, some notion of scoping or windowing is needed. Distributed and reliable memory-based maintenance of window states and intermediate results that are accessible to all cloud processing units across pulses is necessary to guarantee correctness and progressive output that builds upon that intermediate state.

1.5 Online data stream summarization and analytics: In addition to the online processing of the massive data streams, data summarization and online analysis of the streamed data have to take place at various spatial and temporal granularities. This online data analytics component will have access to the memory-based streamed data and their summaries as well as to the disk-based related data, e.g., the road-network data. Phenomena detection, tracking, prediction, and emergency response are example functionalities at this level.

2. Rationale and Challenges for Extensible Context Awareness

Extensibility architectures have been proposed in the early 1980's to address the needs of emerging applications, e.g., as in extensible database servers. Although useful, these extensibility features are not adequate as they do not capture a key sensitive parameter in social networks, and database and web search applications, which is the "context" of the user or

[1] Walid G. Aref's research is supported in part by the National Science Foundation under Grants III-1117766, IIS-0964639, and IIS-0811954.

query issuer. For example, in location-service architectures, the location of the user or the query issuer is a sensitive parameter. In privacy-aware "Hippocratic" database systems [1], the identity of the user or query issuer is a sensitive parameter. In temporal databases, the time the query is issued is a sensitive parameter. In a location-aware social network, both the location and the identity of the user are sensitive parameters. Instead of tailoring a system for each sensitive parameter or each combination of sensitive parameters, we can make the system aware of the notion of "contexts" or "sensitive parameters" related to the user or query issuer. By using "contexts" as an abstraction, we can eliminate the need for tailoring specialized engines for each new context. Devising a general context-aware server will eliminate the need for the costly tailoring of a specialized server, e.g., a location server, a temporal DBMS, or a combined Hippocratic location-aware database server, since space, time, and identity of both the user or query issuer and the underlying database objects are treated as contexts. One can instantiate a new specialized server by defining appropriate contexts using context definition and manipulation languages. An extensible context-aware system should provide:

2.1 User or query-issuer contexts: These are high-level interfaces to define the user's or query issuer's contextual interfaces. This context reflects the situation of the query issuer, e.g., the query issuer's location, the time the query is issued, the identity of the query issuer, or even the temperature or the weather conditions surrounding the query issuer. A Context-aware server should be able to make use of these contexts when responding to a user's request.

2.2 Objects' reciprocal contexts: These are high-level interfaces to define the reciprocal contexts of the database objects. These object contexts will reflect on or reciprocate the user's or query issuer's registered contexts, e.g., the location of the database objects, the time duration of an object (or when the object can be available for querying, e.g., as in a restaurant's opening hours when a user is asking for a close-by restaurant), and the identity of the object (or the ids of the query issuers or classes of query issuers that are allowed to access the object as in privacy-aware database systems).

2.3 Binding mechanisms: The extensible context-aware server should have high-level interfaces that dynamically bind the contexts of the query issuer with those of the database objects. For example, binding the location and identity of the query issuer to the location and privacy profiles of social network objects can realize a location-aware social network.

2.4 Performance: The declarative definition of contexts in context-aware systems can have a strong impact provided that performance of these systems is competitive to those of tailored systems. The efficient realization of context-aware database management systems is one of the vital challenges. Related to the issue of performance is that of indexing. In contrast to building tailored indexing methods, e.g., for the spatial locations of objects, or for temporal intervals, can one construct generalized indexing techniques for contexts?

2.5 Dynamic context profiles: In many application scenarios, changes take place in the contexts, e.g., some active contexts may become inactive, inactive ones may become active, or new contexts get introduced. Another form of change is that the contextual values themselves within a context may change, e.g., the surrounding temperature may change or the location of a moving object may change, etc. These changes may affect the query being executed. This is similar in spirit to mid-query reoptimization [7]. However, the difference is that when the contexts change, the system may need to augment the query being executed by additional predicates that reflect the changes in the contexts.

3. Where We Stand

M3 [2] is a prototype data streaming system that is being realized at Purdue using Hadoop [3] and that eliminates all of Hadoop's disk layers, including the distributed file system (HDFS), and the disk-based communication layer between the mappers and the reducers. So far, M3 realizes features 1.1-1.4 above except that as of now, M3 handles only streaming data and does not handle queries that mix streaming with disk-based data. M3 extends Hive [9] to support streaming and continuous SQL querying using SyncSQL-like extensions [6]. Context awareness as an extensible vehicle has been demonstrated separately in Chameleon [5] based on extensions to PostgreSQL [8]. We are currently studying performance issues and general context-based indexing techniques as a step towards realizing context awareness in M3.

4. References

[1] R. Agrawal, J. Kiernan, R. Srikant, Y. Xu, "Hippocratic Databases", VLDB'02, Hong Kong, China 2002.

[2]A.M. Aly, A. Sallam, B.M. Gnanasekaran, L.-V. Nguyen-Dinh, W.G. Aref, M. Ouzzani, A. Ghafoor, "M3: Stream Processing on Main-Memory MapReduce". Demo Paper, IEEE ICDE, April 2012.

[3] "Apache hadoop: http://hadoop.apache.org/."

[4] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters", OSDI'04, pp. 137–150, December 2004.

[5] H.G. Elmongui, W.G. Aref, M.F. Mokbel, "Chameleon: Context-Awareness inside DBMSs", ICDE'09, pp. 1335-1338, April 2009.

[6] T.M. Ghanem, A.K. Elmagarmid, P.A. Larson, and W.G. Aref, "Supporting views in data stream management systems," ACM TODS, 2010.

[7] N. Kabra, D. Dewitt, "Efficient Mid-Query Re-Optimization of Sub-Optimal Query Execution Plans", ACM SIGMOD'98, Seattle, WA 1998.

[8] "PostgreSQL. www.postgresql.org/".

[9] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, "Hive - a warehousing solution over a MapReduce framework," PVLDB, 2009.