# ASTERIX: Put Your Social Data Here!

Michael J. Carey
Computer Science Department, UC Irvine
*mjcarey@ics.uci.edu*

## 1 ASTERIX Objectives

The ASTERIX project at UC Irvine [4] began in early 2009 with the goal of combining and extending ideas drawn from semistructured data management [3], parallel database systems [10], and first-generation data-intensive computing platforms [9] to create a new, highly-scalable, semistructured information management system. ASTERIX is targeting use cases related to archiving, querying, and analyzing semistructured data drawn from Web sources, including sources such as Twitter, social networking sites, news sites, blogs, and so on.

Today's "Big Data" stack has formed around open-source software including Hadoop [1], its HDFS file system, and also various key-value stores [8]. Early clients of this stack primarily used the MapReduce programming model; today, the majority of its use is via declarative languages like Pig and HiveQL. Rather than adopting or modifying the current Hadoop stack to add features, the ASTERIX project is asking "What if we'd actually meant to design an open software stack with records at the bottom and a higher-level language API at the top?" We have thus been rethinking the layers [6] while taking care not to lose important attributes such as open-source availability, non-monolithic layers/components, access to file-based external data as well as system-managed data, fault-tolerant job execution and storage, and automatic data placement and rebalancing as data and machines come and go.

## 2 The ASTERIX System

Figure 1 provides an overview of the ASTERIX system. Data management in ASTERIX is based on a semistructured data model that can support use cases ranging from strict, table-like data collections with predictable types to flexible and more complex data where very little is known a priori and the instances in data collections are variant and self-describing. The ASTERIX data model (ADM) design was based on taking data concepts from JSON and adding additional primitive types as well as borrowing collection types from object databases [7]. The ASTERIX query language (AQL) was designed to manipulate semistructured ADM data. It was influenced by XQuery FLWOR expressions [12] while leaving XQuery's significant "XML baggage" behind.

While designing and building ASTERIX, three reusable software layers emerged; these layers are summarized in Figure 2. The bottom-most layer is a data-intensive runtime system called Hyracks [5]. The topmost layer of the ASTERIX stack is the ASTERIX system itself, a full parallel information management system that supports both native storage and indexing of data as well as access to external data residing in a distributed file system. Between these two layers lies Algebricks, a model-agnostic, algebraic "virtual machine" for parallel query processing and optimization. Algebricks is the target for AQL query compilation, but it can also be a target for other declarative data languages (e.g., we currently have Facebook's HiveQL language running on top of Algebricks). In addition, we are experimenting with graph data analysis and a Pregel-like programming API built on Hyracks to support it.

Given the target use cases, incoming data will often be dirty or noisy, making fuzzy search a key feature of ASTERIX. Analyzing such data to make recommendations or to identify sub-populations and trends in social networks requires matching data based on set-similarity measures. AQL supports such fuzzy joins over large data sets and executes them in parallel [11]. As the ASTERIX system is aimed at supporting data drawn from (or pushed to ASTERIX from) around the Web, including the increasingly popular and important mobile Web, ASTERIX includes built-in support for location-based (i.e., geospatial) data as well as support for automatic data ingestion via a new ASTERIX feature called datafeeds. Datafeeds channel data coming from continuous Web sources (such as Twitter and RSS-based news services) into affiliated ASTERIX datasets for either immediate or later searching and analysis.

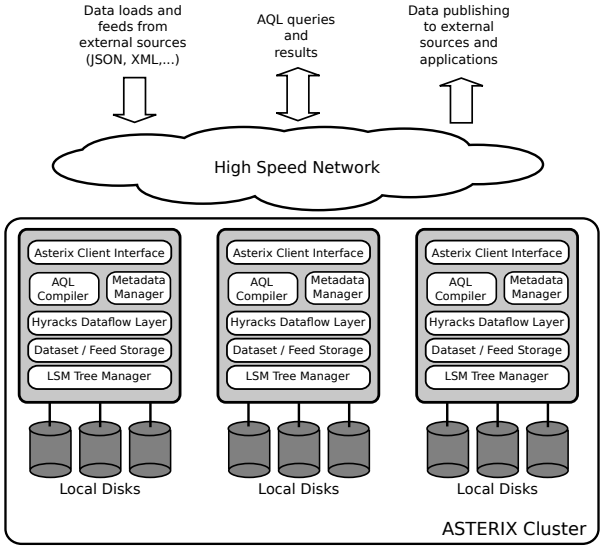


Figure 1: ASTERIX system overview

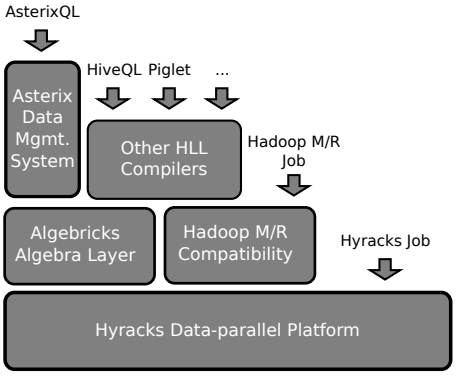


Figure 2: ASTERIX software stack

# 3 ASTERIX Status

We are 2.5 years into our initial 3-year, NSF-sponsored ASTERIX effort [2]. The Hyracks layer is now fairly mature and has been shown to significantly outperform Hadoop and other open-source alternatives on data-intensive computing problems on clusters with up to 200 nodes (with 800 disks and 1600 cores). The Algebricks layer emerged as a result of work to support both AQL and HiveQL on Hyracks; we have measured 3-8x performance gains in preliminary experiments comparing HiveQL on Algebricks with Hive itself on Hadoop. A group at Yahoo! Labs is using Hyracks (and planning to use Algebricks) as a platform for work on data-parallel machine learning over very large data sets. The ADM/AQL layer, *a.k.a.* ASTERIX proper, is now able to run parallel queries including lookups, large scans, parallel joins, and parallel aggregrates efficiently for data stored in a partitioned LSM B+ tree storage component and indexed via LSM B+ tree and LSM-based R-tree indexes. The system's external data access and datafeed features are also running in the lab. We are planning to offer a first open-source release of ASTERIX and its component layers sometime during the latter part of 2012, and we are actively looking for early partners who might like to try the ASTERIX software out on their favorite "Big Data" problems.

# References


[1] Apache Hadoop website. `http://hadoop.apache.org`.

[2] Asterix project website. `http://asterix.ics.uci.edu/`.

[3] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, 1999.

[4] A. Behm, V. R. Borkar, M. J. Carey, R. Grover, C. Li, N. Onose, R. Vernica, A. Deutsch, Y. Papakonstantinou, and V. J. Tsotras. Asterix: towards a scalable, semistructured data platform for evolving-world models. *Distributed and Parallel Databases*, 29(3):185–216, 2011.

[5] V. R. Borkar, M. J. Carey, R. Grover, N. Onose, and R. Vernica. Hyracks: A flexible and extensible foundation for data-intensive computing. In *ICDE*, pages 1151–1162, 2011.

[6] V. R. Borkar, M. J. Carey, and C. Li. Inside "Big Data management": Ogres, onions, or parfaits?". In *Proc. EDBT 2012 Conf.*, March 2012.

[7] R. Cattell, editor. *The Object Database Standard: ODMG 2.0*. Morgan Kaufmann, 1997.

[8] R. Cattell. Scalable SQL and NoSQL data stores. *SIGMOD Rec.*, 39:12–27, May 2011.

[9] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In *OSDI '04*, pages 137–150, December 2004.

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

[11] R. Vernica. *Efficient Processing of Set-Similarity Joins on Large Clusters*. Ph.D. Thesis, Computer Science Department, University of California-Irvine, 2011.

[12] XQuery 1.0: An XML query language. http://www.w3.org/TR/xquery/.