The BigDawg Architecture and Reference Implementation

Jennie Duggan
Northwestern U.
jennie@eecs.northwestern.edu

Aaron Elmore
U. of Chicago
aelmore@cs.uchicago.edu

Tim Kraska
Brown U.
tim_kraska@brown.edu

Sam Madden
M.I.T.
madden@csail.mit.edu

Tim Mattson
Intel Corp.
timothy.g.mattson@intel.com

Michael Stonebraker
M.I.T.
stonebraker@csail.mit.edu

ABSTRACT

This paper presents the reference implementation of a new architecture for future "Big Data" applications. Such applications require "big analytics" as one might expect, but they also require real-time streaming support, real-time analytics, data visualization, and cross-storage queries. We are guided by the principle "one size does not fit all" [7], and we build on top of three storage engines, each designed for specialized use cases. In addition, we demonstrate novel support for querying across multiple storage engines as well as pioneering solutions to data visualization. In the remainder of this short paper, we describe the first of three BigDawg reference implementations, Bulldog. In the next two years we expect to follow with Pitbull and Rottweiler releases.

1. INTRODUCTION

Intel created an Intel Science and Technology Center (ISTC) focused on "Big Data" in 2012. This center, with a hub at MIT and spokes at five other universities, has built a big data architecture and a reference implementation. We are guided by the following tenets:

Tenet 1: One size does not fit all. It is clear that high performance SQL analytics, real time decision support, OLTP, and complex analytics will be optimized by different engines. Hence, Big Data applications with complex heterogeneous data will utilize multiple storage engines in a single application with three novel storage engines in our current reference implementation.

Tenet 2: Real-time decision support is crucial. We expect an increasing amount of data will stream into an application through an Internet of Things (IOT) architecture, and one needs to act on this data in real time. It serves no useful purpose, for example, to send a hospital dischargee home with a monitoring device, if downstream hardware and software is not capable of real-time response. Hence, support for real time streaming data and real-time analytics is crucial. In this paper, we describe a pioneering real-time system for streaming data.

Tenet 3: The interface to big data applications will move from today's form-based interactions to a visualization focus. Historically, visualization system loaded their data into main memory to provide interactive responses to users gestures. Such "small vis" cannot have a place in a big data stack. In this paper, we sketch our ideas and preliminary code to deal with big vis.

Tenet 4: Any big data application will fundamentally deal with data in multiple storage engines, whether it be to run analytics that compare today (streaming system) with yesterday (data warehouse) or to run complex analytics that combine patient metadata (RDBMS) with time series data (array engine). We are developing multi-database support and present our initial results in this paper.

To address these challenges, we are building a new database architecture, that we call BigDawg, which is designed to support multiple storage backends with a unified querying, visualization, and analytics interface on top. This paper describes our initial architecture, which we call Bulldog.

To make sure our efforts are concrete, we are using Bulldog to host a data set from the MIMIC II [6], and have built a demo application using it to explore implications of the above tenets. In Section 2 we briefly describe the MIMIC II data and in the remaining sections our explorations on each of the four tenets above.

2. MIMIC II DATA

The MIMIC II data set is a publicly accessible data set on about 26,000 intensive care unit (ICU) admissions at Boston's Beth Israel Deaconess Hospital [6]. It contains waveform data (up to 125 Hz measurements from bedside devices), patient metadata (name, age, etc.), doctor's and nurse's notes (text), lab results (semi-structured data) and prescriptions filled (semi-structured data). In a possible production implementation, one would store all of this data, augmented by streaming input from sensors. Hence, this system must support a variety of data types, standard SQL analytics (how many patients were given a particular drug), complex analytics (compute the FFT of a patients waveform data and then compare it to "normal"), and text search (find patients who responded well to a particular drug or treatment).

3. MULTIPLE STORAGE ENGINES

The Bulldog implementation loads MIMIC II data in a mixture of three storage engines. Text is stored in Accumulo [1], patient metadata in Postgres and waveform data in SciDB [3]. Postgres and SciDB are well known, while Accumulo is a key-value store, popular in government circles, and favored by our implementation partner, MIT Lincoln Labs. Future releases will incorporate additional data models and storage engines. In particular, we plan to integrate a prototype array engine, TileDB, a high-performance functional engine, Tupleware, and a transactional stream processing engine, S-Store.

4. REAL-TIME OPERATIONS

To support streaming operation, one can adopt either a stream-processing point of view (and use an engine such as Aurora or Storm) or a high performance OLTP strategy that uses a main memory DBMS such as VoltDB, MemSQL, or Hana. We take an OLTP strategy and have built an engine S-Store that extends a main memory OLTP system (H-Store) with streaming primitives. Hence, we expect waveform data from MIMIC II patients to enter BigDawg through S-Store, with real-time processing and modification provided by stored procedures. Ultimately, the data ages out of S-Store and is loaded in big "chunks" into SciDB, where historical processing can be performed.

5. BIG DATA VISUALIZATIONS

Although our demo application includes real time monitoring and standard data operations, most of the user interface deals with visualization. We envision four distinct interfaces:

Browsing: This is a pan/zoom interface whereby a user can browse through the entire MIMIC II data set, drilling down on demand to access more detailed information. This system must be able to efficiently display top level data (an icon for each group of the 20,000 patient-days) and then drill down as needed. To provide interactive response, it is imperative to prefetch data in anticipation of user movements. We have built both a markov model (prefetch objects in the direction the user is currently going) and a semantic model (prefetch objects with a similar signature to the current one). A user study has confirmed that a hybrid approach of the two tactics provides best overall performance.

"Tell me something interesting": Here, the user has a mountain of data and is looking for "something interesting". We have built a system (SeeDB [8]) that runs analytics, looking for distinctive patterns in the output as specified by the user. This screen runs SeeDB and presents such results.

Complex analytics: This screen enables a non-programmer to run a variety of complex analytics, such as linear regression, FFT, and PCA on specified sets of patient waveform data.

Text analysis: Using this interface, a user may run complex keyword searches such as "find me the patients that have at least three doctor's report saying 'very sick' and are taking a particular drug".

Our approach is to build visualization interfaces using the University of Washington composition engine Vega [2]. In addition, we will explore how client-side caching and prefetching can interact with their server side counterparts.

6. CROSS-SYSTEM QUERYING

Data is stored in multiple storage engines and is exposed through a streaming interface and a visualization system, as noted in Figure 1. Our goal is to provide "location transparency" so application programmers do not need to know which storage engine is actually used to run their queries. Ideally, programmers would use a single "universal query language". In practice, this may prove challenging owing to each storage engine being designed around a different target problem space. Hence, we are investigating ways to support location transparency across the intersecting subsets of the capabilities of our engines. Both Myria [4] and D4M [5] are efforts in this direction.

We diagram this architecture in Figure 1. In this system, we have three storage engines ("S", "A", "R"), for the streaming, array, and relational stores. BigDawg communicates to any number of storage engines using *shims* that translate utterances to and from a common intermediate BigDawg language.

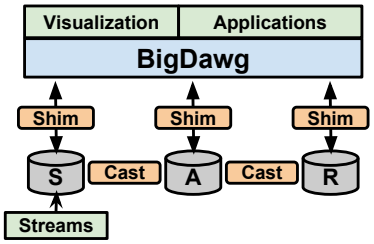

Figure 1: BigDawg / Bulldog Architecture

When queries across multiple systems are required, we will move datasets or intermediate results from one system to another as needed. In particular, when an object, such as an array, is queried using semantics that differ from its native storage, a *cast* is needed to make it clear when the semantics will change during query execution. BigDawg queries select the system that will be responsible for executing different clauses in a query using a *scope*. To illustrate the operation of a cast, consider the following relational filtering operation on an array A:

```
RELATIONAL(select * from CAST(A, relation) where v > 5);
```

The above *cast* notation provides a way to express cross-system physical query plans. We are investigating higher-level declarative systems that don't require end-users to manually write plans with cast operations, but that instead use cost-based estimation techniques to determine where and when to perform casts. We are also investigating techniques to make cross-system interfaces more efficient than file-based import/export.

7. CONCLUSIONS

Bulldog is the 2015 reference implementation of BigDawg, containing S-Store, Tupleware, TileDB as well as Postgres and SciDB. It contains both D4M and Myria as integration systems, and multiple vis systems built using Vega. Some of the components are available now, and the entire system will be available in Q3/2015, along with a reference example using MIMIC II data.

8. REFERENCES

[1] Accumulo. https://accumulo.apache.org/.

[2] Vega. http://git.io/xKyrmQ.

[3] P. Cudré-Mauroux et al. A demonstration of SciDB: A science-oriented dbms. *PVLDB*, 2(2):1534–1537, 2009.

[4] D. Halperin et al. Demonstration of the myria big data management service. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 881–884, 2014.

[5] J. Kepner et al. Dynamic distributed dimensional data model (d4m) database and computation system. In *ICASSP*, pages 5349–5352. IEEE, 2012.

[6] M. Saeed and others. Multiparameter Intelligent Monitoring in Intensive Care II (MIMIC-II): A public-access intensive care unit database. *Critical Care Medicine*, 39:952–960, May 2011.

[7] M. Stonebraker and U. Cetintemel. "One Size Fits All": An Idea Whose time has come and gone. In *ICDE*, pages 2–11, 2005.

[8] M. Vartak, S. Madden, A. Parameswaran, and N. Polyzotis. Seedb: Automatically generating query visualizations. In *PVLDB*, 2014.