The BigDAWG Polystore

by

Michael Stonebraker

DBg Database Group
MIT Computer Science and Artificial Intelligence Lab

Purpose of BigDAWG (Intel ISTC on Big Data)

- Flatten BDAS (Badass)

Outline

- Why do we need a polystore
- What exactly is a Polystore
- The BigDAWG stack
- The big kahuna

Why Polystores?

- "One size fits all" is over
- Legacy RBMSs are now good for nothing
  - And should be sent to the "home for tired software"
  - In every vertical market I can think of, there is something that is 1 – 2 orders of magnitude faster
- And other data types are now important
  - Not just structured data

One Size Does Not Fit All

- Recorded Futures runs
  - MongoDB for semi-structured data
  - Postgres for structured data
  - Lucene (elastic search) for text

One Size Does Not Fit All

- Tamr runs
  - Postgres for OLTP
  - Impala/Spark for analytics
  - Lucene (elastic search) for materialized views

One Size Does Not Fit All

- Mimic II data (40K ICU patients from Beth Israel Hospital)
  - Real time waveform data (from bedside monitors)
  - Historical time series data (waveform archive)
  - Patient metadata (standard structured data)
  - Nurse's and doctor's notes (text)
  - Prescriptions (semi-structured data)

Our BigDAWG Prototype

- Real time waveform data
  - S-Store
- Historical time series data
  - SciDB
- Patient metadata
  - Postgres
- Nurse’s and doctor’s notes
  - Accumulo
- Prescriptions
  - Accumulo

BigDAWG Requirements

- Real time support
  - “Code blue”
  - Alert doctor if real time differs from history
- Cross-system queries
  - Find me everybody who is taking drug XXX
  - Find me the average age of patients listed as “very sick” by a nurse or doctor
- Analytics
  - Correlate heart-rate to age for patients over 45
- Novel user-level functionality
  - “tell me something interesting”

Without Help, This is a Programming Nightmare

- Programmer must learn K DBMS interfaces
- And copy data between systems
  - To do a cross system join
- And recode his application if any data moves
  - As requirements evolve

BigDAWG Needs a Federated DBMS (reminiscent of the 1980's)

- K independent data bases
  - Usually on different storage software!!!
- With different schemas and query languages
- With a federation query language
  - Perhaps relational
  - “Shims” to mediate between federation language and local languages

Think of This as an "Island of Information"

- One query language
- N local DBMSs, with shims
- Preserve location transparency
  - Can move stuff for optimization
- Programmer codes against this "island interface"
  - Learns one interface!
  - His programs are insulated from data movement!

Extension: Multiple Islands of Information

- No esperanto query language
  - SQL, array-talk, text talk…
- Impossible to achieve location transparency for all DBMS functionality
  - Occasionally need to talk directly to underlying systems

BigDAWG Research Program

- "Thin Middle" multi-island interface
  - Effort to make islands as semantically inclusive as possible
- Intra-Island middleware
  - Monitoring framework (MIT)
  - Machine-learing optimizer (Northwestern/MIT)
  - Data mover (Chicago, UW)
  - Query rewriter (Northwestern, UW)
  - Myria island (UW)
  - Spark island (MIT)

BigDAWG Research Program

- Novel Storage Managers (for islands to integrate)
  - S-Store (MIT)
  - TileDB (MIT)
  - Tupleware (Brown)
- Novel User-interface systems (which call the thin middle)
  - ScalaR/ForeCache (MIT)
  - SeeDB (MIT/UIUC)
  - VizDom (Brown)
  - Searchlight (Brown)

BigDAWG Stack

- All of this stuff
- Plus Postgres, Accumulo, SciDB
- Open Source
- Available on Github in a few months (as soon as we make it reliable and usable)
  - In staged releases (Bulldog, Rotweiller, Pit Bull)

So Why Not Just Use Hadoop?

- Map-Reduce is good for nothing
    - Is being effectively jettisoned from the Hadoop stack
- Systems built on top of HDFS (e.g. Impala) can be be integrated into BigDAWG islands

So Why Not Just Use Spark?

- Spark is not currently location transparent
  - Matei is looking at this
  - I.e. making Spark an island
- Spark has limited scope (analytics)
  - No OLTP
  - No text

So Why Not Just Use Spark?

- Spark has serious architecture issues
  - Always a good idea to send the query to the data (push all possible functionality into storage managers)
  - Not bring the data to the query (do queries in middleware)
  - Spark does the latter – loses to SciDB by X20 on joins
- Spark supports only column representation and row representation
  - Killed on analytics by an array store

The Big Kahuna

- BigDawg currently assumes:
  - The data “integration-ready” – i.e. is not a legacy structure which needs curation
- I.e. appropriate for new applications not Alastair’s RBS problem

Data Curation

- Ingest
- Clean (-99 means null)
- Transform (Euros to dollars)
- Schema integrate (bottom up or top down)
- Entity consolidation (M.Stonebraker and Mike Stonebraker are the same entity)

Typical Curation (e.g. Tamr)

- Does curation with a human assist when necessary
- Using a push-based architecture
  - Data sources shoved through a workflow

The Big Kahuna

- Integrate BigDawg (pull-based architecture) with curation
- Plus discovery and privacy
- Future focus of our ISTC