Platforms for Big Data Management and Analysis

Michael J. Carey
Information Systems Group
UCI CS Department
Outline

• Big Data Platform Space
  – The Big Data Era
  – Brief History of Data Platforms
  – Dominant Platforms

• AsterixDB Project @ UCI / UCR
  – Our Approach
  – Feature Overview
  – Call for Customers

• Quick Q&A (time permitting)
It’s the “Big Data” Era

• Unprecedented growth in data being generated and its potential uses/value
  – Tweets, social networks (statuses, check-ins, shared content), blogs, click streams, various logs, ...
  – *Facebook*: > 1.3B monthly active users, > 2.7B likes/day
  – *Twitter*: > 270M monthly active users, > 500M tweets/day

• Everyone is interested (might be why you’re here! 😊)
  – Trade press and popular press: “*Big Data!*”
  – Enterprises, Web companies, online businesses, governments, public health researchers, social scientists, ...
  – Untapped value and countless new opportunities to understand, optimize, assist, and/or compete
One Reason Big Data’s Exploding
Brief History of Big Data

• Database community – all about “Big Data” management since the beginning of time
  – Born from the relational database movement initiated by Tedd Codd in 1970

• Distributed systems community – all about “Big Data” since 2000 or so (thanks to the Web)
  – Google, Yahoo!, Amazon, Facebook, …, all started fresh, but are now following the DB field’s path

• Let’s briefly review architectures from both worlds (to combine/extend their best ideas)
Big Data in the Database World

• Enterprises wanted to store and query historical business data (data warehouses)
  – 1970’s: Relational databases appeared (w/SQL)
  – Late 1970’s: Database machines based on novel hardware and early (brute force) parallelism
  – 1980’s: Parallel database systems based on “shared-nothing” architectures (Gamma, GRACE, Teradata)
  – 2000’s: Netezza, Aster Data, DATA Allegro, Greenplum, Vertica, ParAccel, … (Serious “Big $” acquisitions!)

(Each node runs an instance of a legitimate, indexed DB data storage and runtime system)
Also OLTP Database Systems

• On-line transaction processing is another Big Data dimension
  – Applications that power our daily business
  – Producers of the data being warehoused
• Shared-nothing clustered architectures are also a serious scalable OLTP player
  – 1980’s: Tandem’s NonStop SQL system
Big Data in the Systems World

• Late 1990’s brought a need to index and query the rapidly exploding content of the Web
  — DB technology really didn’t fit the problem(s)
  — Google, Yahoo! *et al* had to do something

• Google responded by laying a new foundation
  — Google File System (GFS)
    • OS-level byte stream files spanning 1000’s of machines
    • 3-way replication for fault-tolerance (and high availability)
  — MapReduce (MR) programming model
    • User writes just two simple functions: *Map* and *Reduce*
    • “*Parallel programming for dummies*” — MR runtime does all the heavy lifting (using partitioned parallelism)
(MapReduce: A Quick Example)

Romeo, Romeo, wherefore art thou Romeo?

What, art thou hurt?

Input Splits (distributed)

Mapper Outputs

SHUFFLE PHASE (based on keys)

Reducer Inputs

Reducer Outputs (distributed)

Partitioned Parallelism!
Soon a Star Was Born...

• Yahoo!, Facebook, and friends cloned Google’s “Big Data” infrastructure from papers
  – GFS → Hadoop Distributed File System (HDFS)
  – MapReduce → Hadoop MapReduce
  – Widely used for Web indexing, click stream analysis, log analysis, information extraction, some machine learning

• Tired of puzzle-solving with just two moves, higher-level languages were developed to “hide” MR
  – E.g., Pig (Yahoo!), Hive (Facebook), Jaql (IBM)
  – Now in heavy use over MR (Pig > 60%, HiveQL > 90%)

• Similar happenings at Microsoft
  – Cosmos, Dryad, and SCOPE (which powers Bing)
Also Key-Value or Document Stores

- Another Big Data dimension, for applications that power social sites, gaming sites, and so on
  - Systems community’s version of OLTP, roughly
- Need for simple record stores
  - Simple, key-based retrievals and updates (put/get)
  - Fast, highly scalable, highly reliable, and available
- Many “NoSQL” systems (see Cattell survey)
  - Proprietary: BigTable (Google), Dynamo (Amazon), …
  - Open Source: HBase (BigTable), Cassandra (Dynamo), …
Notes:

- One storage manager per machine in a parallel cluster
- Upper layers orchestrate their shared-nothing cooperation
- *One way in/out*: through the SQL door at the top
Open Source Big Data Stack

Notes:

- Giant byte sequence files at the bottom
- Map, sort, shuffle, reduce layer in middle
- Possible storage layer in middle as well
- HLLs now at the top

(Notice: More open...!)
No More “One Size Fits All”
Other Up-and-Coming Platforms

• Bulk Synchronous Programming (BSP) platforms, e.g., Pregel, Giraph, GraphLab, ..., for doing Big Graph analysis

  “Think Like a Vertex”
  – Receive messages
  – Update state
  – Send messages

(“Big” is the platform’s concern)

• Spark for in-memory cluster computing – for repetitive data analyses, iterative machine learning tasks, ...

\[ \text{Input} \rightarrow \text{Distributed memory} \left\{ \begin{array}{c}
\text{query 1} \\
\text{query 2} \\
\text{query 3}
\end{array} \right\} \rightarrow \text{one-time processing} \]

\[ \text{Input} \rightarrow \text{Iter. 1} \rightarrow \text{Iter. 2} \rightarrow \ldots \]

\[ \text{Input} \rightarrow \text{one-time processing} \]

\[ \text{In-} \rightarrow \text{memory cluster computing} \rightarrow \text{one-time processing} \]
AsterixDB: “One Size Fits a Bunch”

Semistructured Data Management

Parallel Database Systems

Data-Intensive Computing

BDMS Desiderata:

• Flexible data model
• Efficient runtime
• Full query language
• Costs proportional to tasks at hand
• Built-in support for continuous data ingestion
• Support today’s “Big Data data types”
User Model  (Scale-Independent!)

create dataverse MyNewSocialSite;

create type MugshotUserType as {
  id: int32,
  alias: string,
  name: string,
  user-since: datetime,
  address: {
    street: string, city: string,
    state: string, zip: string,
    country: string
  },
  friend-ids: {{ int32 }},
  employment: [EmploymentType]
}

create dataset MugshotUsers(MugshotUserType)
  primary key id;

Ex: User names and messages sent by users who joined the Mugshot social network within a certain time window:

for $user in dataset MugshotUsers
  where $user.user-since >= datetime('2010-07-22T00:00:00')
  and $user.user-since <= datetime('2012-07-29T23:59:59')
return {
  "uname" : $user.name,
  "messages" :
    for $message in dataset MugshotMessages
      where $message.author-id = $user.id
    return $message.message
};

Highlights include:
- JSON++ based data model
- Rich base types (spatial, temporal, …)
- Records, lists, bags
- Open vs. closed types
- Powerful query language
ASTERIX System Overview

Data loads and feeds from external sources (JSON, XML,...)

AQL queries and results

Data publishing to external sources and applications

High Speed Network

(ADM = ASTERIX Data Model; AQL = ASTERIX Query Language)
ASTERIX Software Stack

AQL
HiveQL
XQuery

AsterixDB
Hivesterix
Apache VXQuery

Algebricks Algebra Layer

Pregel Job

Hadoop M/R Job

M/R Layer

Hyracks Job

Hyracks Data-Parallel Platform

#AsterixDB
Project Status

- 4-year initial NSF project done (now ~300 KLOC)
  - Code developed jointly by UCI and UCR
  - Two new efforts – including hardening/sharing AsterixDB

- AsterixDB BDMS is here! (@ June 6th, 2013)
  - Semistructured “NoSQL” style data model
  - Declarative parallel queries, inserts, deletes, ...
  - Data storage with primary and secondary indexing
  - Internal and external datasets both supported
  - Rich set of data types (incl. text, time, and geo location)
  - Fuzzy and spatial query processing
  - Data feeds and external indexing as well

- We would love to work with you, if you have use cases...!
  - Ex: Social data analytics, education (especially MOOCs), public health, emergency response, power use monitoring, ...
AsterixDB project page: http://asterixdb.ics.uci.edu
Pregelix project page: http://pregelix.ics.uci.edu

Open source code base:
- ASTERIX: http://code.google.com/p/asterixdb/
- (Pregelix: http://hyracks.org/projects/pregelix/)