A New Platform for Real-Time Big Data BI

Michael Carey
Information Systems Group
CS Department
UC Irvine

#AsterixDB
Rough Plan

• Background and motivation
• Context (a brief history of 2 worlds)
• AsterixDB: a next-generation BDMS
• The ASTERIX software stack
• “One Size Fits a Bunch” (and Q&A)
Everyone’s Talking About Big Data

• Driven by unprecedented growth in data being generated and its potential uses and value
  – Tweets, social networks (statuses, check-ins, shared content), blogs, click streams, various logs, ...
  – Facebook: > 845M active users, > 8B messages/day
  – Twitter: > 140M active users, > 340M tweets/day
What’s going on right now?
A Recent Personal Example

- I was in Studio City, CA, helping my (aspiring filmmaker) son move, and we took a break at *Coffee Bean and Tea Leaf*...
• Might want to search for tweets that mention “Eddie Murphy” and “girlfriend”
  – How many such tweets in the last month?
  – How many in the last month, grouped by location?
  – How many in the last minute, grouped by location, where location is within a block of the CBTL shop in Studio City?
  – How many ... by a user who has actually checked in at CBTL using Facebook or Foursquare?
  – Etc.
“Real Time” for Big Data Analytics

• Not intended for navigating driverless cars or tracking incoming missiles...

• Desired for analyzing trends, especially across multiple large (archived) data sources

• Interested in spotting (and studying) current trends in a sea of data of varying quality/value
  – Fast queries
  – Fast aggregates
  – Continuous queries (later)
Big Data in the *Database* World

• Enterprises needed to store and query historical business data (data warehouses)
  – 1980’s: Parallel database systems based on “shared-nothing” architectures (Gamma/GRACE, Teradata)
  – 2000’s: Netezza, Aster Data, DATAllegro, Greenplum, Vertica, ParAccel (“Big $” acquisitions!)

• OLTP is another category (a source of Big Data)
  – 1980’s: Tandem’s NonStop SQL system
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
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 tried but failed (*e.g.*, Inktomi)
  - Google, Yahoo! *et al* needed to do something
- Google responded by laying a new foundation
  - Google File System (GFS)
    - OS-level byte stream files spanning 1000’s of machines
    - Three-way replication for fault-tolerance (availability)
  - MapReduce (MR) programming model
    - User functions: Map and Reduce (and optionally Combine)
    - “*Parallel programming for dummies*” — MR runtime does the heavy lifting via partitioned parallelism
Soon a Star Was Born...

• Yahoo!, Facebook, and friends read the papers
  – HDFS and Hadoop MapReduce now in wide use for indexing, clickstream analysis, log analysis, ...

• Higher-level languages subsequently developed
  – Pig (Yahoo!), Jaql (IBM), Hive (Facebook)
  – In heavy use over MR (Pig > 60%, HiveQL > 90%)

• Similar story at Microsoft (Cosmos, Dryad, SCOPE)

• Key-value (“NoSQL”) stores are another category
  – Used to power scalable social sites, online games, ...
  – BigTable→HBase, Dynamo→Cassandra, MongoDB, ...

#AsterixDB
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
- Now at the top: HLL’s (Huh...?)
Today’s Tangle
AsterixDB: “One Size Fits a Bunch”
• Build a new Big Data Management System (BDMS)
  – Run on large commodity clusters
  – Handle mass quantities of semistructured data
  – Openly *layered*, for selective reuse by others
  – Share with the community via open source
• Conduct scalable information systems research
  – Large-scale query processing and workload management
  – Highly scalable storage and index management
  – Fuzzy matching, spatial data, date/time data (all in parallel)
  – Novel support for “fast data” (both in and out)

Train next generation of “Big Data” graduates
ASTERIX Hadoop Influences

- Open source availability ("price is right")
- Non-monolithic layers or components
- Support for external data access (in files)
- Replication and roll-forward recovery (*later*)
- Automated data placement and balancing (*later*)
AsterixDB System Overview

Data loads and feeds from external sources

Hi-Speed Interconnect

AQL queries/results

Data publishing

Asterix Client Interface

AQL Compiler

Metadata Manager

Hyracks Dataflow Engine

Dataset / Feed Storage

LSM Tree Manager

ASTERIX Cluster

(A DM = ASTERIX Data Model; AQL = ASTERIX Query Language)
create dataverse LittleTwitterDemo;

create type TweetMessageType as open {
  tweetid: string,
  user: {
    screen-name: string,
    lang: string,
    friends_count: int32,
    statuses_count: int32,
    name: string,
    followers_count: int32
  },
  sender-location: point?,
  send-time: datetime,
  referred-topics: {{ string }},
  message-text: string
};

create dataset TweetMessages(TweetMessageType)
primary key tweetid;

Highlights:
• JSON++ based data model
• Rich type support (spatial, temporal, …)
• Records, lists, bags
• Open vs. closed types
• External data sets and datafeeds
**Ex: TweetMessages Dataset**

```json
{{
    "tweetid": "1023",
    "user": {
        "screen-name": "dflynn24",
        "lang": "en",
        "friends_count": 46,
        "statuses_count": 987,
        "name": "danielle flynn",
        "followers_count": 47
    },
    "sender-location": "40.904177,-72.958996",
    "send-time": "2010-02-21T11:56:02-05:00",
    "referred-topics": [{ "verizon" }],
    "message-text": "i need a #verizon phone like nowwww! :(
    }
},
{
    "tweetid": "1024",
    "user": {
        "screen-name": "miriamorous",
        "lang": "en",
        "friends_count": 69,
        "statuses_count": 1068,
        "name": "Miriam Songco",
        "followers_count": 78
    },
    "send-time": "2010-02-21T11:43:08-00",
    "referred-topics": [{ "Commercial", "verizon", "att" }],
    "message-text": "#verizon & #att #commercials, so competitive"
}
{
    "tweetid": "1025",
    "user": {
        "screen-name": "dj33",
        "lang": "en",
        "friends_count": 96,
        "statuses_count": 1696,
        "name": "Don Jango",
        "followers_count": 22
    },
    "send-time": "2010-02-21T12:38:44-05:00",
    "referred-topics": [{ "charlotte" }],
    "message-text": "Chillin at dca waiting for 900am flight to charlotte and from there to providenciales"
}
{
    "tweetid": "1026",
    "user": {
        "screen-name": "reallyleila",
        "lang": "en",
        "friends_count": 106,
        "statuses_count": 107,
        "name": "Leila Samii",
        "followers_count": 52
    },
    "send-time": "2010-02-21T21:31:57-06:00",
    "referred-topics": [{ "verizon", "at&t", "iphone" }],
    "message-text": "i think a switch from #verizon to #at&t may be in my near future... my smartphone is like a land line compared to the #iphone!"
}
}}

#AsterixDB
ASTERIX Query Language (AQL)

- **Ex:** List the user information and Tweet message text for Tweets by user “dflynn24”:

```plaintext
for $tweet in dataset TweetMessages
where $tweet.user.screen-name = 'dflynn24'
return {
  "tweeter": $tweet.user,
  "tweet": $tweet.message-text
}
```

**Highlights:**
- Lots of other features (see Beta!)
- Set-similarity matching (~= operator)
- Spatial predicates and aggregation
- And plans for more...

#AsterixDB
AQL (cont.)

- *Ex:* List the topics being Tweeted about, along with their associated Tweet counts, in Verizon-related Tweets:

```plaintext
for $tweet in dataset TweetMessages
where some $topic in $tweet.referred-topics
    satisfies contains($topic, "verizon")
for $topic in $tweet.referred-topics
group by $topic with $tweet
return {
    "topic": $topic,
    "count": count($tweet)
}
```

```plaintext
{{
    { "topic": "verizon", "count": 3 },
    { "topic": "commercials", "count": 1 },
    { "topic": "att", "count": 1 },
    { "topic": "at&t", "count": 1 },
    { "topic": "iphone", "count": 1 }
}}
```
Fuzzy Joins in AQL

- Ex: Find Tweets with similar content:

```plaintext
for $tweet1 in dataset TweetMessages
for $tweet2 in dataset TweetMessages
where $tweet1.tweetid != $tweet2.tweetid
  and $tweet1.message-text =~ $tweet2.message-text
return {
  "tweet1-text": $tweet1.message-text,
  "tweet2-text": $tweet2.message-text
}
```
Continuous Data Feeds (Future)

• *Ex:* Create “Fast Data” feeds for Tweets:

```sql
create feed TwitterFirehoseSimulator
using TwitterAdapter (|“tps”|=2000))
apply function addHashTagsToTweet;

create dataset TweetMessages (TweetMesageType)
primary key tweetid;

create index locationIndex on
TweetMessages(sender-location)
type rtree;

connect feed TwitterFirehoseSimulator to dataset Tweets;
disconnect feed TwitterFirehoseSimulator from dataset Tweets;
```

*Highlights:*
- Philosophy: “keep everything”
- Data ingestion, not data streams
- Previous queries unchanged
AsterixDB

**Data Feed Pipelines**

External World

Continuous Data Source

AsterixDB

**INGESTION**

AsterixDB Node

AsterixDB Operator

**COMPUTE**

# (PRIMARY KEY)

(Policy knobs to dial in robustness and elasticity)

STORAGE

LSM Index

(Logical Storage Map)

(External World)

(Continuous Data Source)

(AsterixDB Node)

(AsterixDB Operator)
LSM-Based Storage & Index Partitions

New data

In-Memory Component

On-Disk Components

C_0

C_1

C_2

Instance of Index I

Deleted-Key B^+-Tree

Bloom Filter

LSM-ified Indexes:
- B+ trees
- R trees (secondary)
- Inverted (secondary)
The ASTERIX Software Stack

AsterixQL

HiveQL

Piglet

... Hadoop M/R Job

... Pregel Job

... IMRU Job

Algebricks

Hadoop M/R Compatibility

Pregelix

IMRU

Hyracks Job

Hyracks Data-parallel Platform

#AsterixDB
Hyracks

- Partitioned-parallel platform for data-intensive computing
- Job = dataflow DAG of operators and connectors
  - Operators consume and produce *partitions* of data
  - Connectors *route* (repartition) data between operators

Hyracks vs. the “competition”
- Based on time-tested parallel database principles
- vs. Hadoop: More flexible model and less “pessimistic”
- vs. Dryad: Supports data as a first-class citizen
Hyracks Performance
(On a small cluster with 40 cores & 40 disks in 2011)
Hyracks Performance Benefits

• Does eager (push-based) job activation
• Sorts/hashes on serialized (binary) data
• Has pipelining as well as options to persist results
• Has \( n \)-ary operators based on parallel DB principles
  – Availability and use of indexes
  – Hash-based joins and grouping (not just sort-based)
• Etc. (see paper)

Proc. 27th ICDE Conf., Hannover, Germany, April 2011.
AsterixDB for RTBDBI?

• “One Size Fits a Bunch” can offer better functionality, manageability, and performance than gluing together multiple point solutions (e.g., Hadoop + Hive + MongoDB):
  – LSM-based indexes for dynamic data with queries
  – Spatial indexing and spatial query capabilities
  – Fuzzy indexing and query processing for similarity
  – External datasets (and data feeds) for external data

• Hyracks is a more powerful, flexible, and efficient run-time dataflow engine than Hadoop – and offers an open stack
  – Operators/primitives based on parallel DB best practices
  – Experiments show up to $10x$ performance speedups at scale (on disk-resident problems and data sizes)
The New Kid on the Block!

http://asterixdb.ics.uci.edu
Current Status

- 4+ years of initial NSF project (~250 KLOC)
- Code scale-tested on a 6-rack Yahoo! Labs cluster with roughly 1400 cores and 700 disks
- AsterixDB BDMS is now here! (@ June 6th, 2013)
  - Semistructured “NoSQL” style data model
  - Declarative parallel queries, inserts, deletes, ...
  - LSM-based storage/indexes (primary & secondary)
  - Internal and external datasets both supported
  - Fuzzy and spatial query processing
  - NoSQL-like transactions (for inserts/deletes)
  - And data feeds are running in the lab...
Partial Cast List

• Faculty and research scientists
  – UCI: Michael Carey, Chen Li; Vinayak Borkar, Nicola Onose (Google)
  – UCR: Vassilis Tsotras
  – Oracle Labs: Till Westmann

• PhD students
  – UCI: Rares Vernica (HP Labs), Alex Behm (Cloudera), Raman Grover, Yingyi Bu, Sattam Alsubaiee, Yassar Altowim, Hotham Altwajry, Pouria Pirzadeh, Zachary Heilbron, Young-Seok Kim
  – UCR/UCSD: Jarod Wen, Preston Carman, Nathan Bales (Google)

• MS students
  – UCI: Guangqiang Li (MarkLogic), Vandana Ayyalasomayajula (Yahoo!), Siripen Pongpaichet, Ching-Wei Huang, Manish Honnatti (Zappos), Xiaoyu Ma, Madhusudan Cheelangi (Google), Khurram Faraaaz (IBM DB2), Tejas Patel

• BS students (alumni)
  – UCI: Roman Vorobyov, Dustin Lakin

Foreign affiliates
  – Thomas Bodner (T.U. Berlin), Markus Dressler (HPI), Rico Bergmann (Humboldt U.)
Collaborations

• Facebook
• Yahoo! Research
• Rice University
• UC Santa Cruz
• EMC/Greenplum
• UC San Diego
• IIT Mumbai
• Apache Software Foundation
• Oracle Labs
• HTC
• Microsoft Research
• UCI Informatics
For More Info

AsterixDB project page: http://asterixdb.ics.uci.edu

Open source code base:

- ASTERIX: http://code.google.com/p/asterixdb/
- Pregelix: http://hyracks.org/projects/pregelix/
Questions?

http://www.zazzle.com/asterixdb+gifts