AsterixDB

A Scalable Open Source DBMS

This presentation is based on slides made by Michael J. Carey, Chen Li, and Vassilis Tsotras
So what went on – and why?

What’s going on right now?
Also: Today’s Big Data Tangle

Spark SQL

MongoDB

Cassandra

MySQL

HBase

Hadoop

Hive
AsterixDB: “One Size Fits a Bunch”

Semistructured Data Management

BDMS Desiderata:
- Able to manage data
- Flexible data model
- Full query capability
- Continuous data ingestion
- Efficient and robust parallel runtime
- Cost proportional to task at hand
- Support “Big Data” data types

Parallel Database Systems
1st Generation “Big Data” Systems
ASTERIX Data Model (ADM)

CREATE DATAVEVERSE TinySocial;
USE TinySocial;

CREATE TYPE GleambookUserType AS {
    id: int,
    alias: string,
    name: string,
    userSince: datetime,
    friendIds: {{ int }},
    employment: [EmploymentType]
};

CREATE TYPE EmploymentType AS {
    organizationName: string,
    startDate: date,
    endDate: date?
};

CREATE DATASET GleambookUsers (GleambookUserType)
PRIMARY KEY id;

Highlights include:
- JSON++ based data model
- Rich type support (spatial, temporal, …)
- Records, lists, bags
- Open vs. closed types
ASTERIX Data Model (ADM)

CREATE DATaverse TinySocial;
USE TinySocial;

CREATE TYPE GleambookUserType AS {
   id: int
}

CREATE TYPE EmploymentType AS {
   organizationName: string,
   startDate: date,
   endDate: date?
}

CREATE DATASET GleambookUsers (GleambookUserType)
PRIMARY KEY id;

Highlights include:
› JSON++ based data model
› Rich type support (spatial, temporal, …)
› Records, lists, bags
› Open vs. closed types
CREATE DATaverse TinySocial;
USE TinySocial;
CREATE TYPE GleambookUserType AS {
    id: int
};
CREATE TYPE GleambookMessageType AS {
    messageId: int,
    authorId: int,
    inResponseTo: int?,
    senderLocation: point?,
    message: string
};
CREATE TYPE EmploymentType AS {
    organizationName: string,
    startDate: date,
    endDate: date?
};
CREATE DATASET GleambookUsers (GleambookUserType)
PRIMARY KEY id;
CREATE DATASET GleambookMessages (GleambookMessageType)
PRIMARY KEY messageId;
Ex: GleambookUsers Data

{"id":1, "alias":"Margarita", "name":"MargaritaStoddard", "nickname":"Mags", "userSince":datetime("2012-08-20T10:10:00"), "friendIds":{2,3,6,10}, "employment": [{"organizationName":"Codetechno", "startDate":date("2006-08-06")}, {"organizationName":"geomedia", "startDate":date("2010-06-17"), "endDate":date("2010-01-26")}], "gender":"F" },

{"id":2, "alias":"Isbel", "name":"IsbelDull", "nickname":"Izzy", "userSince":datetime("2011-01-22T10:10:00"), "friendIds":{1,4}, "employment": [{"organizationName":"Hexviafind", "startDate":date("2010-04-27")} ]},

{"id":3, "alias":"Emory", "name":"EmoryUnk", "userSince":datetime("2012-07-10T10:10:00"), "friendIds":{1,5,8,9}, "employment": [{"organizationName":"geomedia", "startDate":date("2010-06-17"), "endDate":date("2010-01-26")}] },

.....
CREATE INDEX gbUserSinceIdx ON GleambookUsers(userSince);
CREATE INDEX gbAuthorIdx ON GleambookMessages(authorId) TYPE BTREE;
CREATE INDEX gbSenderLocIndex ON GleambookMessages(senderLocation) TYPE RTREE;
CREATE INDEX gbMessageIdx ON GleambookMessages(message) TYPE KEYWORD;

CREATE TYPE AccessLogType AS CLOSED
{ ip: string, time: string, user: string, verb: string, `path`: string, stat: int32, size: int32 };
CREATE EXTERNAL DATASET AccessLog(AccessLogType) USING localfs
  ("path"="localhost:///Users/mikejcarey/extdemo/accesses.txt"),
  ("format"="delimited-text"), ("delimiter"="|"));

CREATE FEED myMsgFeed USING socket_adapter
  ("sockets"="127.0.0.1:10001"), ("address-type"="IP"),
  ("type-name"="GleambookMessageType"), ("format"="adm"));
CONNECT FEED myMsgFeed TO DATASET GleambookMessages;
START FEED myMsgFeed;

External data highlights:
• Equal opportunity access
• Feeds to “keep everything!”
• Ingestion, not streams
Q1: List the user names and messages sent by Gleambook social network users with less than 3 friends:

```
SELECT user.name AS uname,
       (SELECT VALUE msg.message
        FROM GleambookMessages msg
        WHERE msg.authorId = user.id) AS messages
FROM GleambookUsers user
WHERE COLL_COUNT(user.friendIds) < 3;
```

```
{ "uname": "NilaMilliron", "messages": [ ] }
{ "uname": "WoodrowNehling", "messages": [ " love acast its 3G is good:)" ] }
{ "uname": "IsbelDull", "messages": [ " like product-y the plan is amazing", " like product-z its platform is mind-blowing" ] }
...
Q2: Identify active users (last 30 days) and group and count them by their numbers of friends:

```sql
WITH endTime AS current_datetime(),
     startTime AS endTime - duration("P30D")
SELECT nf AS numFriends, COUNT(user) AS activeUsers
FROM GleambookUsers user
LET nf = COLL_COUNT(user.friendIds)
WHERE SOME logrec IN AccessLog SATISFIES
    user.alias = logrec.user
    AND datetime(logrec.time) >= startTime
    AND datetime(logrec.time) <= endTime
GROUP BY nf;
```

SQL++ highlights:
- Born at UCSD (Yannis P.)
- Many features (see docs)
- Spatial & text predicates
- Set-similarity matching
Updates and Transactions

Q3: Add a new user to Gleambook.com:

```sql
UPSERT INTO GleambookUsers (
    "id":667,"alias":"dfrump",
    "name":"DonaldFrump",
    "nickname":"Frumpkin",
    "userSince":datetime("2017-01-01T00:00:00"),
    "friendIds":{{ }},
    "employment":[{"organizationName":"USA",
    "startDate":date("2017-01-20")}],
    "gender":"M"
);```

- Key-value store-like transactions (w/record-level atomicity)
- Insert, delete, and upsert ops; index-consistent
- 2PL concurrency
- WAL no-steal, no-force with LSM shadowing
AsterixDB System Overview
Hyracks Dataflow Runtime

- 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 MR: More flexible model and less “pessimistic”
  - *vs.* newer SQL-on-Hadoop runtimes: Emphasis on out-of-core execution and adherence to memory budgets
  - Fast job activation, data pipelining, binary format, state-of-the-art DB style operators (hash-based, indexed, ...)
- Early test at Yahoo! Labs on 180 nodes (1440 cores, 720 disks)
Hyracks (cont.)

Query

use dataverse TinySocial

avg {
  for $m$ in dataset MugshotMessages
  where $m$.timestamp >=
    datetime("2014-01-01T00:00:00")
  and $m$.timestamp <
    datetime("2014-04-01T00:00:00")
  return string-length($m$.message)
}
Algebricks Query Compiler Framework

**Algebricks**
- Logical Operators
- Logical Expressions
- Metadata Interface
- Model-Neutral Logical Rewrite Rules
- Physical Operators
- Model-Neutral Physical Rewrite Rules
- Hyracks Job Generator

**Target Query Language**
- Query Parser (AST)
- AST Translator
- Metadata Catalog
- Expression Type Computer
- Logical Rewrite Rules
- Physical Rewrite Rules
- Language Specifics
Native Storage Management

- Datasets Manager
- Memory
  - Working Memory
  - Buffer Cache
  - In-Memory Components
- Transaction Sub-System
  - Transaction Manager
  - Lock Manager
  - Log Manager
  - Recovery Manager
- Disk 1
- Disk n
- IO Scheduler

Hadoop HDFS
LSM-Based Filters

Intuition: Do NOT touch unneeded records

Idea: Utilize LSM partitioning to prune disk components

Q: Get all tweets > T14
Transaction Support

• Key-value store-like transaction semantics
  – Entity-level transactions (by key) within “transactors”
  – Atomic insert, delete, and upsert (including indexing)
  – Concurrency control (based on entity-level locking)
  – Crash recovery (based on no-steal logging + shadowing)
  – Backup and restore support (just in case... 😊)

• Expected use of AsterixDB is to model, capture, and track the “state of the world” (not to be...)

  SELECT ... FROM Weather W...
  // return current conditions by city

(Long serializable reads)
Example AsterixDB Use Cases

Potential use case areas include

- Behavioral science
- Cell phone event analytics
- Social data analytics
- Public health
- Cluster management log analytics
- Power usage monitoring
- IoT data storage and querying

....
Commercial Use: Big Data Analytics

Couchbase Data Platform

- Service-Centric Clustered Data System
- Multi-process Architecture
- Dynamic Distribution of Facilities
- Cluster Map Distribution
- Automatic Failover
- Enterprise Monitoring/Management
- Security
- Offline Mobile Data Integration
- Streaming REST API
- SQL-like Query Engine for JSON
- Clustered* Global Indexes
- Lowest Latency Key-Value API
- Active-Active Inter-DC Replication
- Local Aggregate Indexes
- Full-Text Search*
- Operational Analytics (currently DP)
For More Information

- Asterix project UCI/UCR research home
- Apache AsterixDB home
- SQL++ Primer
  - [http://asterixdb.apache.org/docs/0.9.4.1/sqlpp/primer-sqlpp.html](http://asterixdb.apache.org/docs/0.9.4.1/sqlpp/primer-sqlpp.html)