Introduction to Data Management

Lecture #23

SQL  NoSQL (😊)

Instructor: Mike Carey
mjcarey@ics.uci.edu

Announcements

• Homework info:
  – HW #7: Due today (at 5 PM).
  – HW #8 is the end (“NoSQL”):
    • Due a week from today.
    • Discounted lateness: 5 pts/day for HW #8

• The plan:
  – Today: NoSQL & Big Data (a la AsterixDB)
    • Not in book: See paper linked to wiki syllabus!
    • Also see docs on the Apache AsterixDB site.
Last time: SQL/NoSQL History Talk

- The pre-relational era
- The relational DB era
- Beyond rows and columns?
  1. The object-oriented DB era
  2. The object-relational DB era
  3. The XML DB era
  4. The NoSQL DB era
- Reflections (and then: AsterixDB)

Now the NoSQL DB Era?

- Not from the DB world
  - Distributed systems folks
  - Also various startups
- From caches → K/V use cases
  - Needed massive scale-out
  - OLTP (vs. parallel DB) apps
  - Simple, low-latency API
  - Need a key K, but want no schema for V
  - Record-level atomicity, replica consistency varies
- In the context of this talk, NoSQL does not mean
  - Hadoop (or SQL on Hadoop)
  - Graph databases or graph analytics platforms
NoSQL Data (JSON-based)

Collection(Order)
{
"id": "123",
"Customer":
{
"custName": "Fred",
"custCity": "LA"
}
"total": 25.97,
"Items": [
{"product-sku": 401,
"qty": 2,
"price": 9.99
},
{"product-sku": 544,
"qty": 1,
"price": 3.99
]
}

Collection(Product)
{
"sku": 401,
"name": "Garfield T-Shirt",
"listPrice": 9.99,
"size": "XL"

"sku": 544,
"name": "USB Charger",
"listPrice": 5.99,
"power": "115V"

Note that
- The world’s not flat, but it’s less <messy/>
- We’re now in the 2010’s, timing-wise

Current NoSQL Trends

• Popular examples: MongoDB, Couchbase
• Coveting the benefits of many DB goodies
  – Secondary indexing and non-key access
  – Declarative queries
  – Aggregates and now (initially small) joins
• Seem to be heading towards...
  – BDMS (think scalable, OLTP-aimed, parallel DBMS)
  – Declarative queries and query optimization, but applied to schema-less data
  – Return of (some, optional!) schema information
Our Example: *Apache AsterixDB*

**Our Example: Apache AsterixDB**

- Data loads & feeds from external sources (XML, JSON, ...)
- AQL queries & scripting requests and programs
- Data publishing to external sources and apps

**ASTERIX Goal:**
To ingest, digest, persist, index, manage, query, analyze, and publish massive quantities of semistructured information...


---

**Big Data / Web Warehousing**

- Big Data / Web Warehousing
- So what went on – and why?
- What’s going on right now?
Also: Today’s Big Data Tangle

AsterixDB: “One Size Fits a Bunch”

**BDMS Desiderata:**
- Flexible data model
- Efficient runtime
- Full query capability
- Cost proportional to task at hand (!)
- Designed for continuous data ingestion
- Support today’s “Big Data data types”
  -
  -
  -

* BDMS Desiderata: Semistructured Data Management
* Parallel Database Systems
* Data-Intensive Computing
Project Goals

• 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 (ASF)

• Conduct scalable information systems research, e.g.,
  – 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 Data Model (ADM)

```plaintext
create dataverse TinySocial;
use dataverse TinySocial;
create type MugshotUserType as {
  id: int32,
  alias: string,
  name: string,
  userSince: datetime,
  address: {
    street: string,
    city: string,
    state: string,
    zip: string,
    country: string
  },
  friendIds: [{ int32 }],
  employment: [EmploymentType]
};
create type EmploymentType as open {
  organizationName: string,
  startDate: date,
  endDate: date?
};
create dataset MugshotUsers(MugshotUserType)
  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)**

```javascript
create dataverse TinySocial;
use dataverse TinySocial;
create type MugshotUserType as {
id: int32
};
create type EmploymentType as open {
organizationName: string,
startDate: date,
endDate: date?
};
create dataset MugshotUsers(MugshotUserType)
    primary key id;
```

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

```javascript
create dataverse TinySocial;
use dataverse TinySocial;
create type MugshotUserType as {
id: int32,
alias: string,
name: string,
user-since: date,
address: {
    street: string,
city: string,
state: string,
zip: string,
country: string
},
friend-ids: [int32],
employment: [EmploymentType]
};
create dataverse TinySocial;
use dataverse TinySocial;
create type MugshotUserType as {
id: int32
};
create type MugshotMessageType as closed {
    messageId: int32,
    authorId: int32,
timestamp: datetime,
inResponseTo: int32?,
senderLocation: point?,
tags: [{ string }],
message: string
};
create type EmploymentType as open {
organizationName: string,
startDate: date,
endDate: date?
};
create dataset MugshotUsers(MugshotUserType)
    primary key id;
create dataset MugshotMessages(MugshotMessageType)
    primary key messageId;
```

**Highlights include:**
- JSON++ based data model
- Rich type support (spatial, temporal, ...)
- Records, lists, bags
- Open vs. closed types
Ex: MugshotUsers Data

```json
{ "id":1, "alias":"Margarita", "name":"MargaritaStoddard", "address":{
  "street":"234 Thomas Ave", "city":"San Hugo", "zip":"98765",
  "state":"CA", "country":"USA" },
  "userSince":datetime("2012-08-20T10:10:00"),
  "friendIds":{( 2, 3, 6, 10 )}, "employment":[]}

{ "id":2, "alias":"Isbel", "name":"IsbelDull", "address":{
  "street":"345 James Ave", "city":"San Hugo", "zip":"98765",
  "state":"CA", "country":"USA" },
  "userSince":datetime("2011-01-22T10:10:00"),
  "friendIds":{( 1, 4 )}, "employment":[]} }

{ "id":3, "alias":"Emory", "name":"EmoryUnk", "address":{
  "street":"456 Jose Ave", "city":"San Hugo", "zip":"98765",
  "state":"CA", "country":"USA" },
  "userSince":datetime("2012-07-10T10:10:00"),
  "friendIds":{( 1, 5, 8, 9 )}, "employment":[]} }

... 
```

Other DDL Features

```sql
create index msUserSinceIdx on MugshotUsers(userSince);
create index msTimestampIdx on MugshotMessages(timestamp);
create index msAuthorIdx on MugshotMessages(authorId) type btree;
create index msSenderLocIndex on MugshotMessages(senderLocation) type rtree;
create index msMessageIdx on MugshotMessages(message) type keyword;

// ---------------------- and also ------------------------------------------------------------------
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"="{hostname}://{path}"), ("format"="delimited-text"), ("delimiter"="\"")));
create feed mySocketFeed using socket_adaptor
  ( ("sockets"="{address}:{port}"), ("addressType"="IP"),
   ("type-name"="MugshotMessageType"), ("format"="adm")));
connect feed mySocketFeed to dataset MugshotMessages;
```

External data highlights:
- Equal opportunity access
- "Keep everything!"
- Data ingestion, not streams
ASTERIX Queries (SQL++ or AQL)

• **Q1:** List the user name and messages sent by those users who joined the Mugshot social network in a certain time window:

```sql
select user.name as uname,
    (select value msg.message
    from MugshotMessages msg
    where msg.authorId = user.id) as messages
from MugshotUsers user
where user.userSince >= date('2010-07-22T00:00:00Z')
    and user.userSince <= date('2012-07-29T23:59:59Z');
```

```json
{
    "uname": "IsbelDull", "messages": ["like samsung the plan is amazing",
    "like t-mobile its platform is mind-blowing"]
}
{
    "uname": "EmoryUnk", "messages": ["love sprint its shortcut-menu is awesome:)", ...
}
```

SQL++ (cont.)

• **Q2:** Identify active users and group/count them by country:

```sql
with endTime as current_datetime(),
    startTime as endTime - duration("P30D")
select user.address.country as country, count(users) as activeUsers
from MugshotUsers user
where some logrec in AccessLog satisfies
    user.alias = logrec.user
    and datetime(logrec.time) >= startTime
    and datetime(logrec.time) <= endTime
group by user.address.country;
```

**SQL++ highlights:**
- Lots of other features (see website!)
- Spatial predicates and aggregation
- Set-similarity (“fuzzy”) matching
Updates and Transactions

- **Q3**: Add a new user to Mugshot.com:

```sql
insert into MugshotUsers
(
    "id": 11, "alias": "John",
    "name": "JohnDoe",
    "userSince": datetime("2012-08-20T10:10:00.000Z"),
    "address": { "street": "789 Jane St",
    "city": "San Harry", "state": "CA",
    "zip": "98767", "country": "USA" },
    "friendIds": [{5, 9, 11}],
    "employment": [{ "organizationName": "Kongreen",
    "startDate": date("2009-08-11") } ] } );
```

- 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 Cluster Overview

Data Loads and Feeds  AQL queries and results  Data publishing

Cluster Controller

AsterixDB

MD Node Controller  Node Controller  Node Controller
ASTERIX Software Stack

A Peek at Performance

Table 2: Dataset sizes (in GB)

<table>
<thead>
<tr>
<th>Batch Size</th>
<th>Asterix Schema</th>
<th>Asterix KeyOnly</th>
<th>Syst-X</th>
<th>Mongo</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.091</td>
<td>0.093</td>
<td>0.040</td>
<td>0.035</td>
</tr>
<tr>
<td>20</td>
<td>0.010</td>
<td>0.011</td>
<td>0.026</td>
<td>0.024</td>
</tr>
</tbody>
</table>

Table 4: Average insert time per record (in sec)
Example AsterixDB Use Cases

- Potential use case areas include
  - Social data analytics
  - Cell phone event analytics
  - Behavioral science
  - Education
  - Public health
  - Power usage monitoring
  - Cluster management log analytics
  - ....
Current Status

• 4 year initial NSF project (250+ KLOC), started 2009
• Now officially *Apache AsterixDB*
  – Semistructured “NoSQL” style data model
  – Declarative parallel queries, inserts, deletes, ...
  – Data storage/indexing (primary & secondary, LSM-based)
  – Internal and external datasets both supported
  – Rich set of data types (including text, time, location)
  – Fuzzy and spatial query processing
  – NoSQL-like transactions (for inserts/deletes)
  – Data feeds and indexes for external datasets
  – ....

For More Information

• Asterix project UCI/UCR research home
• Apache AsterixDB home
  – [https://asterixdb.apache.org/docs/0.9.0/index.html](https://asterixdb.apache.org/docs/0.9.0/index.html)
• SQL++ Primer
  – [https://asterixdb.apache.org/docs/0.9.0/sqlpp/primer-sqlpp.html](https://asterixdb.apache.org/docs/0.9.0/sqlpp/primer-sqlpp.html)
• Navigate from CS122a wiki (HW) to get and install it!

QUESTIONS?