AsterixDB Midflight:
A Case Study in Building Systems in Academia

Michael J. Carey

Computer Science Department

University of California, Irvine

mjcarey@ics.uci.edu

(Also affiliated with Couchbase, Inc.)

UCIrvine

(Joint work with ***UC Riverside*** and
with contributions from ***UC San Diego***)

Plan for the Talk

- A Bit of Personal History
- AsterixDB System Overview
- History, Challenges, and Experiences
- Why Build Systems in Academia?
- Initial Commercial Impact
- A Balancing Act
- Closing Thoughts

Plan for the Talk

- A Bit of Personal History
- AsterixDB System Overview
- History, Challenges, and Experiences
- Why Build Systems in Academia?
- Initial Commercial Impact
- A Balancing Act
- Closing Thoughts

CMU in the Late 1970's

- The right place at the right time (by chance)
  - BS (EE/Math), MS (Computer Engineering)
- Science Hall, 3rd floor, had an inspirational view
  - C.mmp (16-processor SMP w/crossbar switch)
  - Cm* (50-processor NUMA machine, DEC LSI-11's)



- Cool distributed/parallel OS work going on
  - Anita Jones (StarOS) – professor, MS co-advisor, mentor
  - John Ousterhout (Medusa) – finishing Ph.D. student
- Knew pretty quickly that I wanted to do a systems Ph.D.
  - Uninspiring Cornell visit, and "ordered" to go to UC Berkeley…

Berkeley in the Early 1980's

- Great place for academic systems work
  - Dave Patterson – about to take a RISC
  - John Ousterhout – new assistant professor
  - Mike Stonebraker – database systems (yawn… ☺)
- Discovered INGRES (and distributed INGRES)
  - Real system, used for teaching and research
  - Prehistoric open source code base
  - Major impact and place in relational history
- Also Berkeley Unix and other major systems projects
  - Operating systems, networking, compilers, architecture, …
  - Senior PhD students included Bill Joy, Eric Schmidt, …

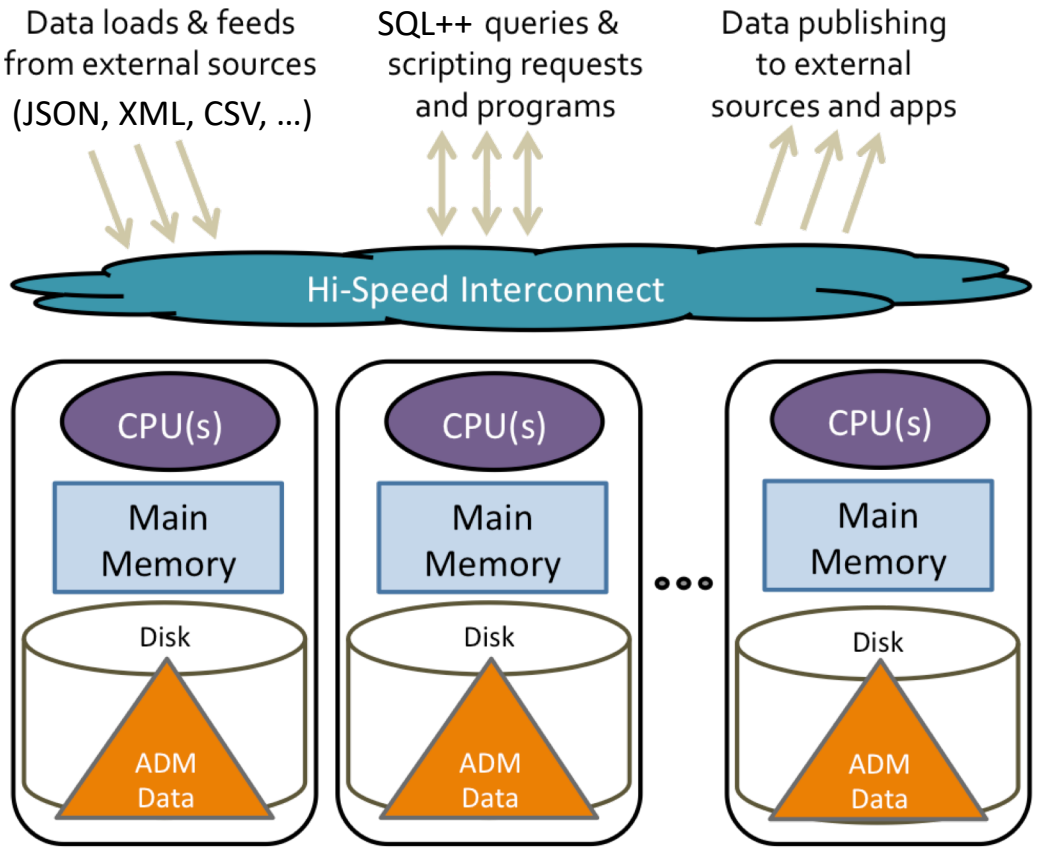
Wisconsin in the Mid/Late 1980's

- Wisconsin DB Systems Group
  - David DeWitt – founder, mentor
  - Gamma parallel shared-nothing DB machine project
  - Other great colleagues soon followed
- Several major systems projects w/David and others
  - EXODUS – extensible, object-oriented (OO) DB system
  - SHORE – distributed OODB system (w/file system view)
- Very conducive to systems work
  - DARPA was funding core DB work
  - NSF provided several large multicomputer grants
  - Department culture was very favorable towards systems
- 13-year industrial "sabbatical" (1995-2008), but then…

Plan for the Talk

- A Bit of Personal History
- AsterixDB System Overview
- History, Challenges, and Experiences
- Why Build Systems in Academia?
- Initial Commercial Impact
- A Balancing Act
- Closing Thoughts

The ASTERIX Project

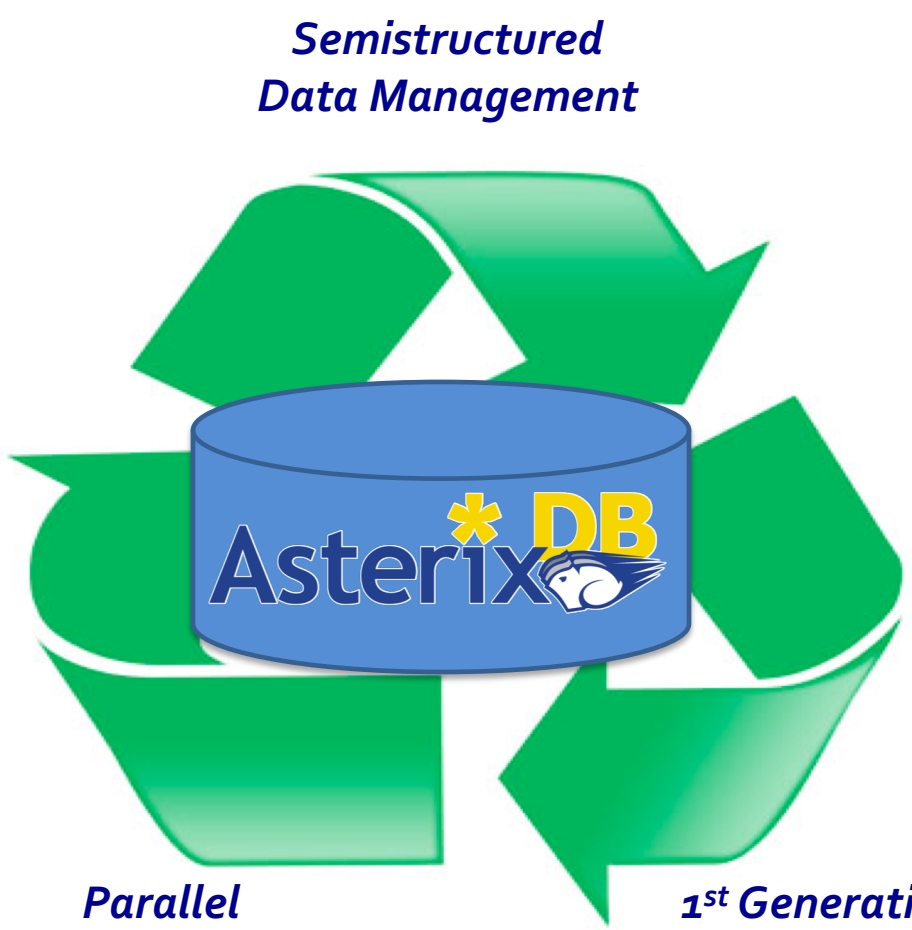


ASTERIX Goal:
To *ingest*, *digest*, *persist*, *index*, *manage*, *query*, *analyze*, and *publish* massive quantities of semistructured information…

http://asterixdb.apache.org/

AsterixDB™

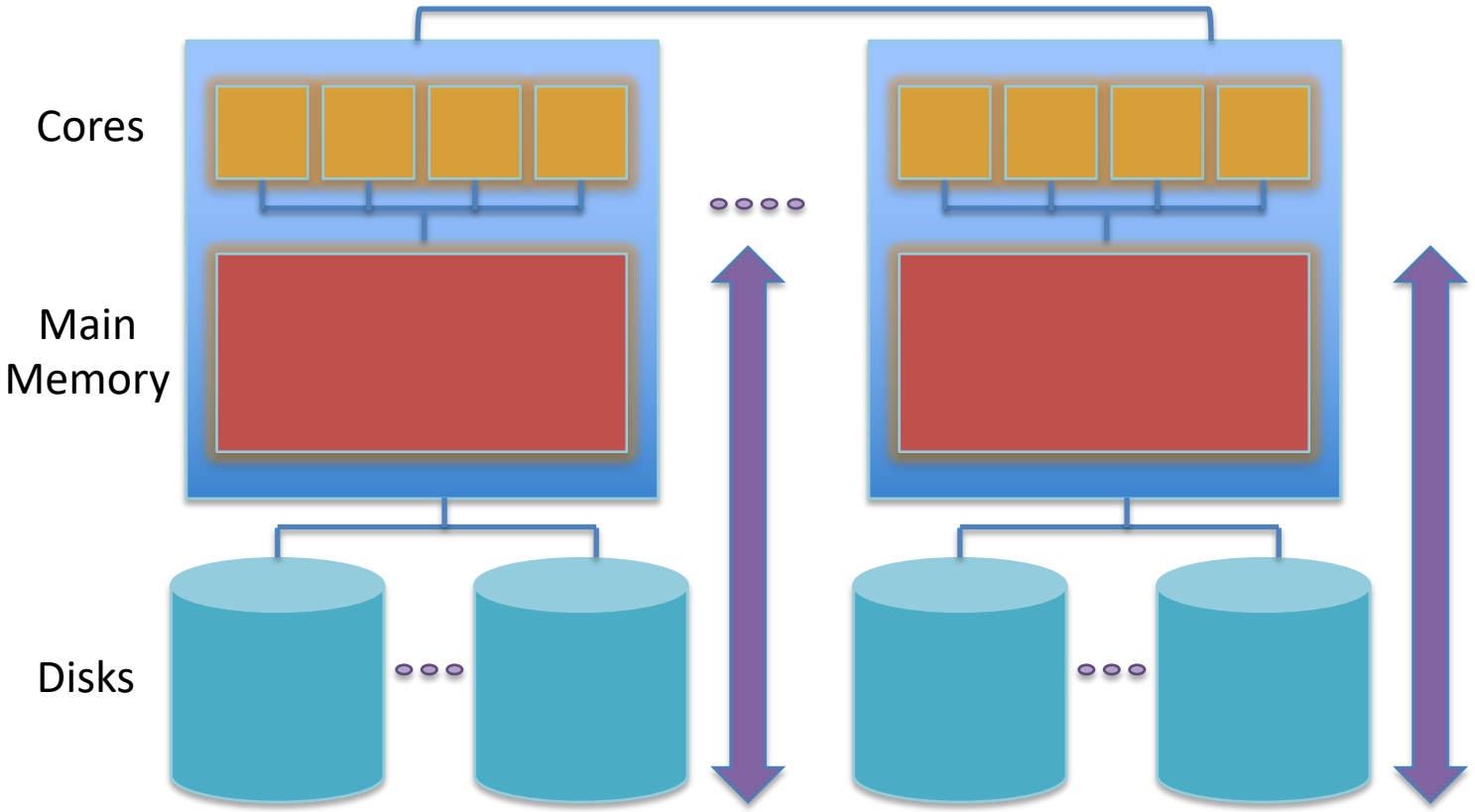
AsterixDB: “One Size Fits a Bunch”



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”

What is “Big Data”?



This is big data!

ASTERIX Data Model (ADM)

```
CREATE DATAVERSE 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***

ASTERIX Data Model (ADM)

```
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;
```

Highlights include:

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

Other DDL Features

```
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 ONGleambookMessages(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"="localhost:///Users/mikejcarey 1/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)

ASTERIX Queries (SQL++ or AQL)

- *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" ] }
. . .
```

SQL++ *(cont.)*

- *Q2:* Identify active users (last 30 days) and group and count them by their numbers of friends:

```
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 SATI
     user.alias = logrec.user
     AND datetime(logrec.time) >= startTime
     AND datetime(logrec.time) <= endTime
GROUP BY nf;
```

```
{ "numFriends": 2, "activeUsers": 1 }
{ "numFriends": 4, "activeUsers": 2 }
 . . .
```

***SQL++ highlights*:**

- UCSD (Papakonstantiou)
- See AsterixDB docs, or
- See D. Chamberlin book!

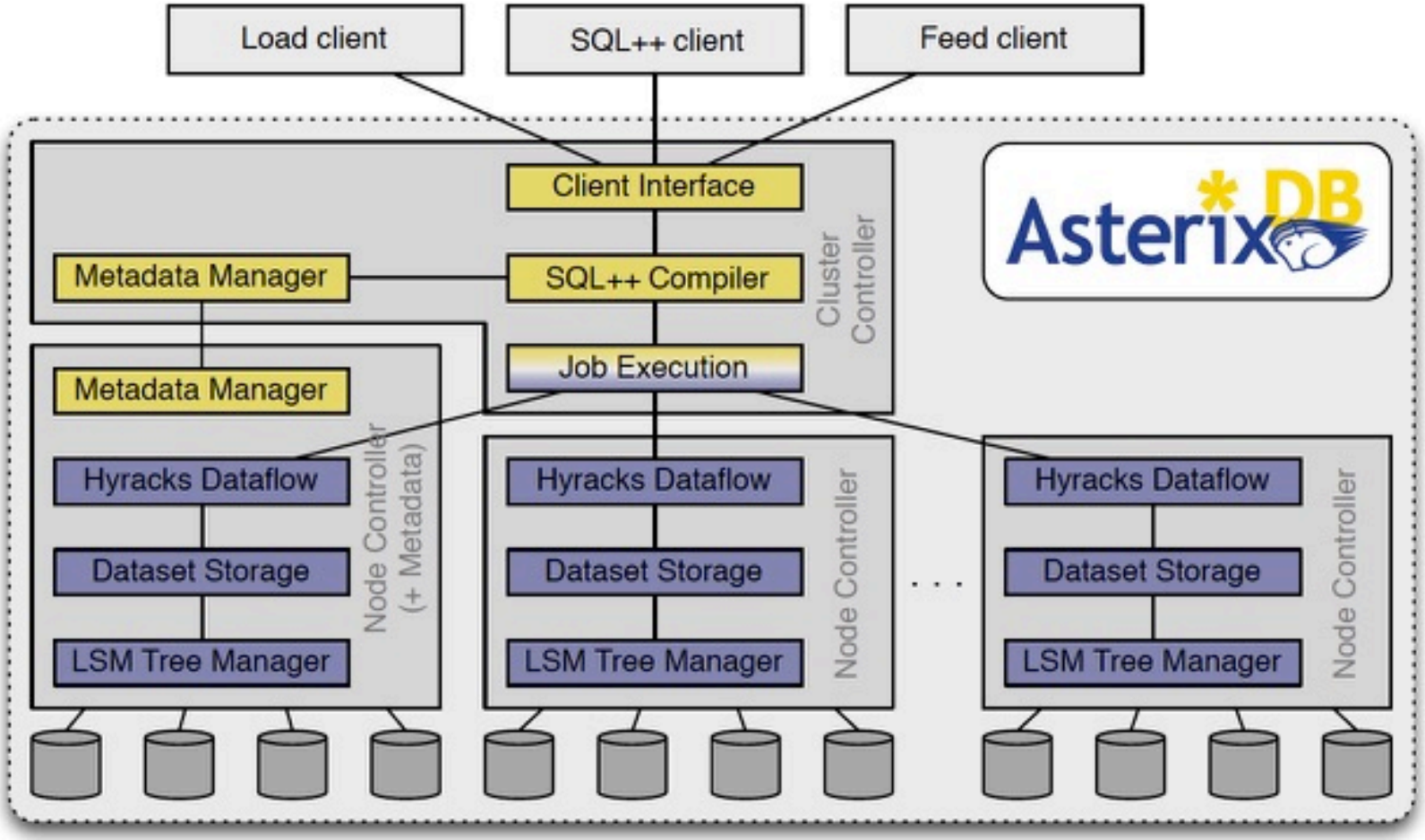
Updates and Transactions

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

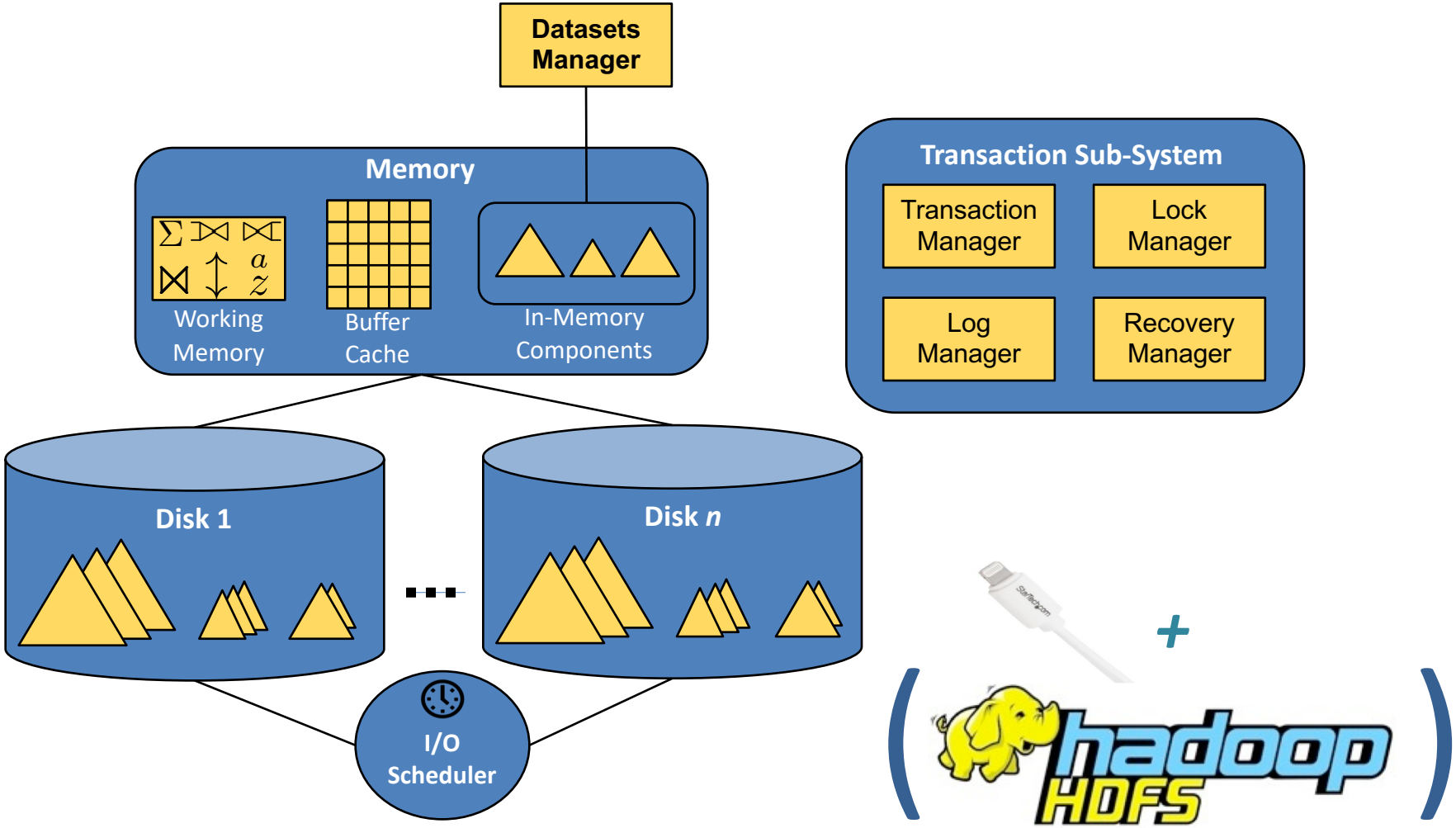
```
UPSERT INTO GleambookUsers (
{"id":667,"alias":"dfrump",
 "name":"DonaldFrump",
  "nickname":"Frumpkin",
  "userSince":datetime("2017-01-01T00:00:00"),
  "friendIds":{{ }},
  "employment":[{"organizationName":"USA",
                 "startDate":date("2017-01-20")}],
  "gender":"M"}
);
```

- **INSERT**, **DELETE**, and **UPDATE**
- Key-value store-like transactions (record-level atomicity)
- Index-consistent (via local indexes)

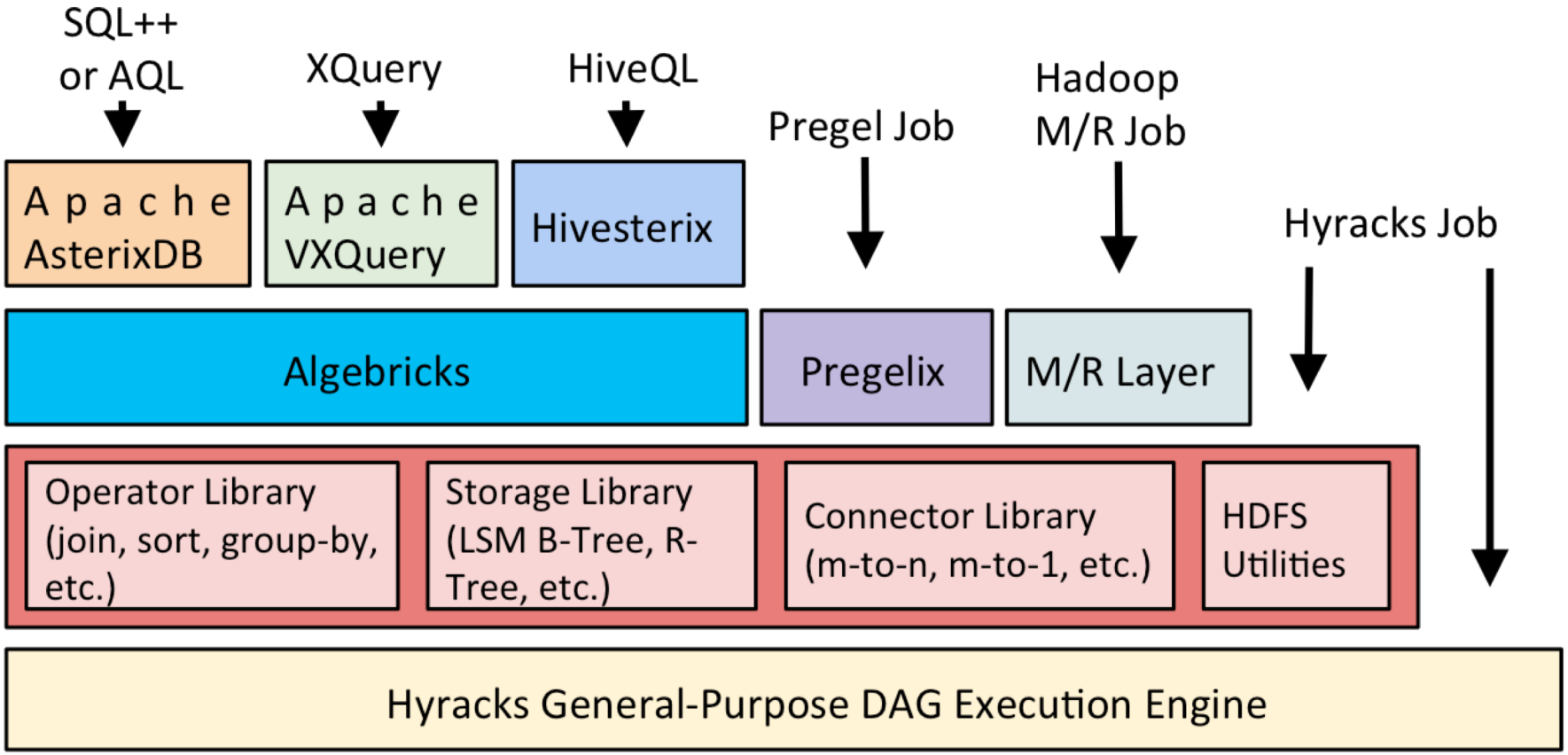
AsterixDB System Overview



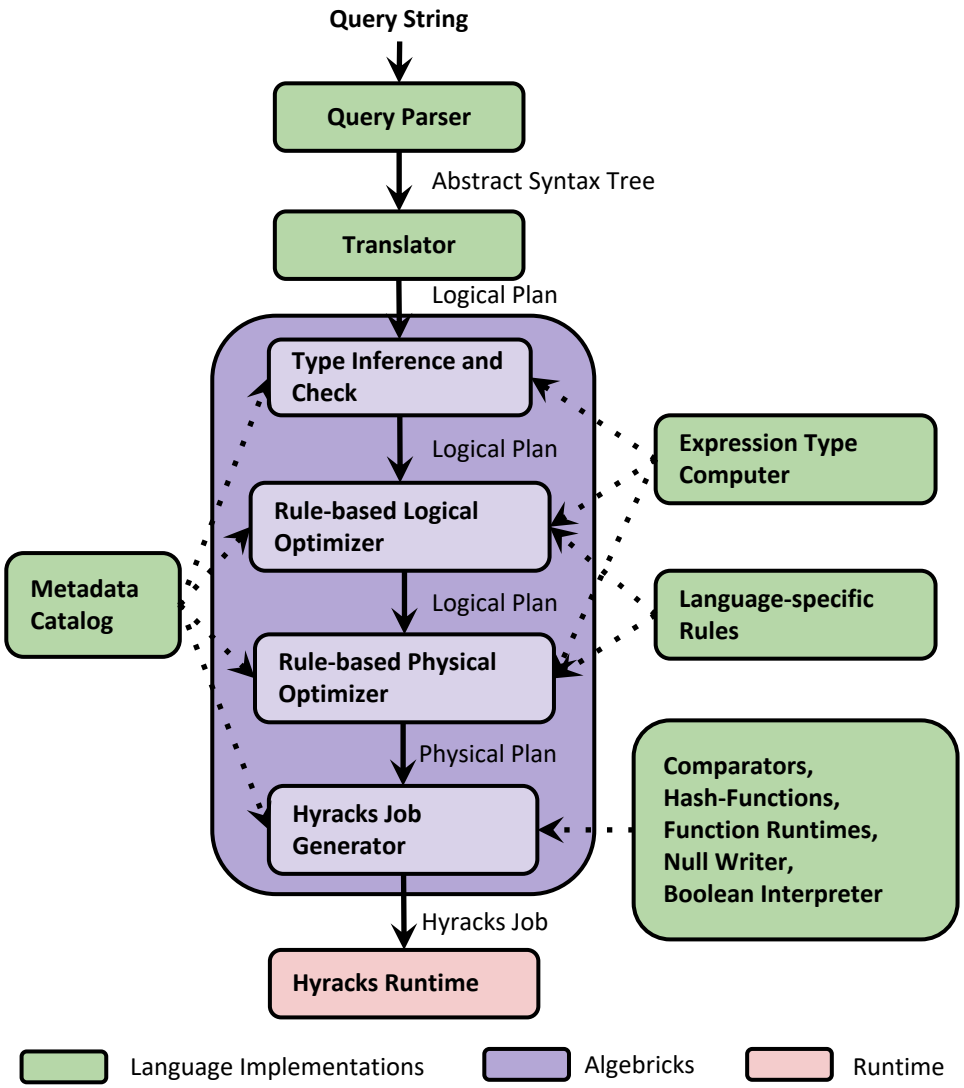
Storage and Memory Management



AsterixDB Software Stack



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

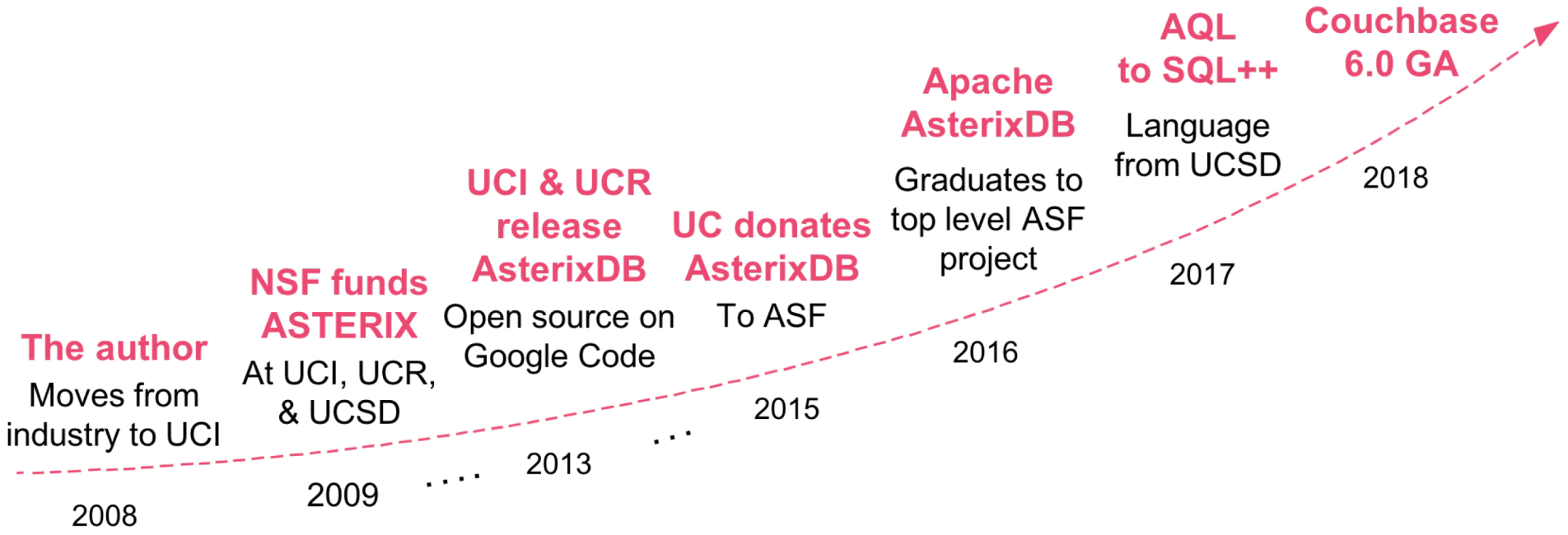
Plan for the Talk

- A Bit of Personal History
- AsterixDB System Overview
- History, Challenges, and Experiences
- Why Build Systems in Academia?
- Initial Commercial Impact
- A Balancing Act
- Closing Thoughts

ASTERIX* Project History

- Started as a large three UC campus NSF project
  - UC Irvine: Chen Li and I, plus Vinayak Borkar (a “pro”) and Nick Onose
  - UC Riverside: Vassilis Tsotras
  - UC San Diego: Yannis Papakonstantinou and Alin Deutsch

  (*Initial semistructured data focus was going to be XML… <:->>)



A Few Historical Highlights

- Some important early decisions (2009-13)
  - JSON was coming → ***ADM*** and ***AQL*** instead of *XML* and *XQuery*
  - UCR decided to have their students commute to UCI on Fridays
  - Made pilgrimages to industry to discuss our project and solicit feedback (including eBay, Facebook, Google, Yahoo!, Teradata, HTC, Netflix, LinkedIn, and Twitter)
  - Solicit industrial gifts (Oracle: "How about a position in Oracle Labs instead?" → Till Westmann, now V.P., Apache AsterixDB)
- Some second phase highlights (2013-17)
  - NSF infrastructure funding (UCI/UCR)
  - Started UCI/UCR Big Active Data (BAD) NSF research project
  - First introduced to Couchbase (more later)
  - Moved from "UC open source" → Apache Software Foundation
  - Adopted SQL++ from Yannis Papakonstantiou (UCSD) after multiple years of justifying AQL over SQL ***(Algebricks to the rescue…!)***

Funding Challenges

- Some reviewers in the DB research universe have "issues" when it comes to systems research projects
  - "Where's the innovation?"
  - "This is just engineering!"
  - "They should really start a company to do this."
- Bias towards "innovative" over "incremental" work
  - I have this great new idea for inverted doubly twisted torus trees!
  - There may be a reason nobody has actually done that (☺)…
  - A significant improvement in something existing is important!
  - We'll double click on this later…

Funding Challenges *(cont.)*

- The *Bandwagon* problem
  - Hadoop/MapReduce
    - "This is a very ambitious proposal, would strongly encourage trying to build atop Hadoop, even if its current filesystem and MR engine are not ideal for addressing this kind of problem. The MR engine is a layer on top (one that Pig uses behind the scenes), while the Distributed File System APIs and HDFS implementation are broadly useful. Work done on top of Hadoop is more likely to be picked up and integrate with other applications."
    - Later vindicated by Impala and Tez runtime efforts
  - "One Size No Longer Fits All"
    - AsterixDB is counter-cultural – but reviewers are unfortunately inherently cultural creatures
    - Side note: we bit off a bigger problem than Spark/Flink (due to storage, indexing, transactions, …), so it's been a longer road
- The big systems budget problem
  - Really need staff as well as students; harder than it once was!

Publishing Challenges

- *Damned if you do, damned if you don't!*
  - Some are critical of work done in a particular system's context
  - A legitimate worry, but broader applicability arguments can be made
- *How hard can that be?*
  - Some reviewers seem clueless about the answer to this question
  - *Ex:* One review for a paper on 4-5 different LSM-based spatial indexes for AsterixDB suggested – in a *2-3 month revision cycle* – adding two more index types and running all the same experiments on them
    - To the best of our knowledge, those methods have never seen the outside of a simulator (loading, concurrency control, recovery, …, plus of course integration into the query optimizer)
    - Our student was accused of "lack of effort" for not doing this additional nine months worth of work
    - That reviewer turned two positive reviewers negative in the final phase ☹

Publishing Challenges *(cont.)*

- The *"BMW model"* (*B*uild, *M*easure, *W*rite)
  - *Goal*: Solve interesting problems and build "cool stuff"
  - Papers are for communicating results; not the goal (!)
  - We therefore tell our students:
    1. Build it (in our case, in the AsterixDB context)
    2. Measure it to see how well it actually works
    3. THEN you can write about it (and look for a paper venue)
  - Note that this doesn't optimize their "paper metrics"
- On that note, two of my personal "software heroes" are
  - Barbara Liskov (MIT): CLU, Argus, … (*ACM Turing Award*)
  - John Ousterhout (now Stanford): networked OS, VLSI tools, log-structured FS's, high-performance distributed systems, …
  - Check out their annual paper production rates and then ponder their impacts. (Just sayin'…)

Plan for the Talk

- A Bit of Personal History
- AsterixDB System Overview
- History, Challenges, and Experiences
- Why Build Systems in Academia?
- Initial Commercial Impact
- A Balancing Act
- Closing Thoughts

Why Build Systems in Academia?

- Maybe because you are “called to do it”
  - I.e., you find it rewarding to build something that didn’t exist before and see it working (and maybe even being used by others)
- Maybe because there are a number or potential benefits
  - Help you avoid things that aren’t worth doing
  - Help guide your focus towards things that definitely matter (well, at least to someone)
  - Let’s double click on this…

1. To Make Sure It's Possible

- "Multidatabase systems" were hot in the late 80's and early 90's – *a.k.a.* federated databases
  - Some groups prototyped systems and then wrote papers, while others did paper designs and wrote papers
- One topic area was transactions (concurrency control), and a focus of some papers was on how to federate heterogeneous systems (mixes of locking, timestamps, and optimistic CC)
  - Assumed exchange of readsets and writesets (records/pages)
  - Unfortunately this is not how the actual systems worked…
    - Most DBMSs used locking and had query-based APIs (e.g., SQL)
- Working in the context of an actual system can help to avoid misdirected effort

2. To Make Sure It's Beneficial

- A few years ago, we heard "everyone knows that ***X*** is the best way to index spatial data" – for different ***X***!
  - Inspired by this opportunity, a student at UCI undertook a study of potential LSM-based spatial secondary index structures
- The results of our study were "disappointing"
  - There were differences if you looked (only) at the *index time*
  - However, for a *query*, 2nd-ary lookup time was not the main cost
    - First do the secondary index lookup
    - Then sort the resulting primary keys (or OIDs)
    - Finally, fetch the actual data records ← *(this is where the time goes)*
- R-trees generally did well over the whole study, so…
  - We made a few R-tree improvements (e.g., points vs. objects)
  - We tossed the other index structures that we'd explored
  - Note that this *was* an important and impactful outcome!
- Working in a real system's context can help you to avoid "polishing a round ball"

3. To Make Sure It's Complete

- You've just invented a great new index structure (inverted doubly twisted torus trees)! But wait, you aren't done – don't forget about updates, concurrency control, recovery, and optimizer integration…
  - Otherwise your idea won't work (at least not yet) in a real system or in a real deployment
- I once asked Goetz Graefe why you almost never find systems with Linear Hashing in addition to B+ trees
  - We know how to *efficiently* load a B+ tree
  - Given a modest amount of memory, both will *only* have buffer misses for leaf pages when searched in practice
  - So why would you want *two of these* in your code base?

4. To Make Sure It's Helpful

- Whenever we invent a new feature, we think:
  - "How can you not want *that*"?
- We learned a lot from a student of Prof. Gloria Mark's at UCI who bravely tried using AsterixDB early in its life
  - She was studying stress and (computer) multitasking in millennials
  - She needed to time-bin a set of time-interval-based observations, and also to properly handle bin-crossing interval data
  - She needed to export as well as import (i.e., round trip) CSV data
  - She also found a few bugs, optimizer blind spots, etc.
- AsterixDB is better as a result of meeting their needs!
  - Another example is Chen Li's Cloudberry project's use of AsterixDB for spatio-temporal social media data analytics (including scale, uptime, and lifecycle)

5. To Get a Free Roadmap

- In my first academic phase, at Wisconsin, for some of our simulation-based work, it sometimes felt like we were inventing problems (because we kind of were)
  - We were basically guessing about what folks might need (e.g., in some work on "firm real-time database systems"
  - It was sometimes disconcerting to see others pick up our assumptions ("But wait! We made that up…!")
- My next-phase experiences with delivering products in industry were very different
  - People will use your systems in ways you didn't think of
  - People will want features you don't yet have
  - Now you can select topics/ideas from a queue of user-generated ideas rather than having to ask "What next?" in a vacuum
- If you build and share a system you can benefit from the same effect – it can help to drive your research

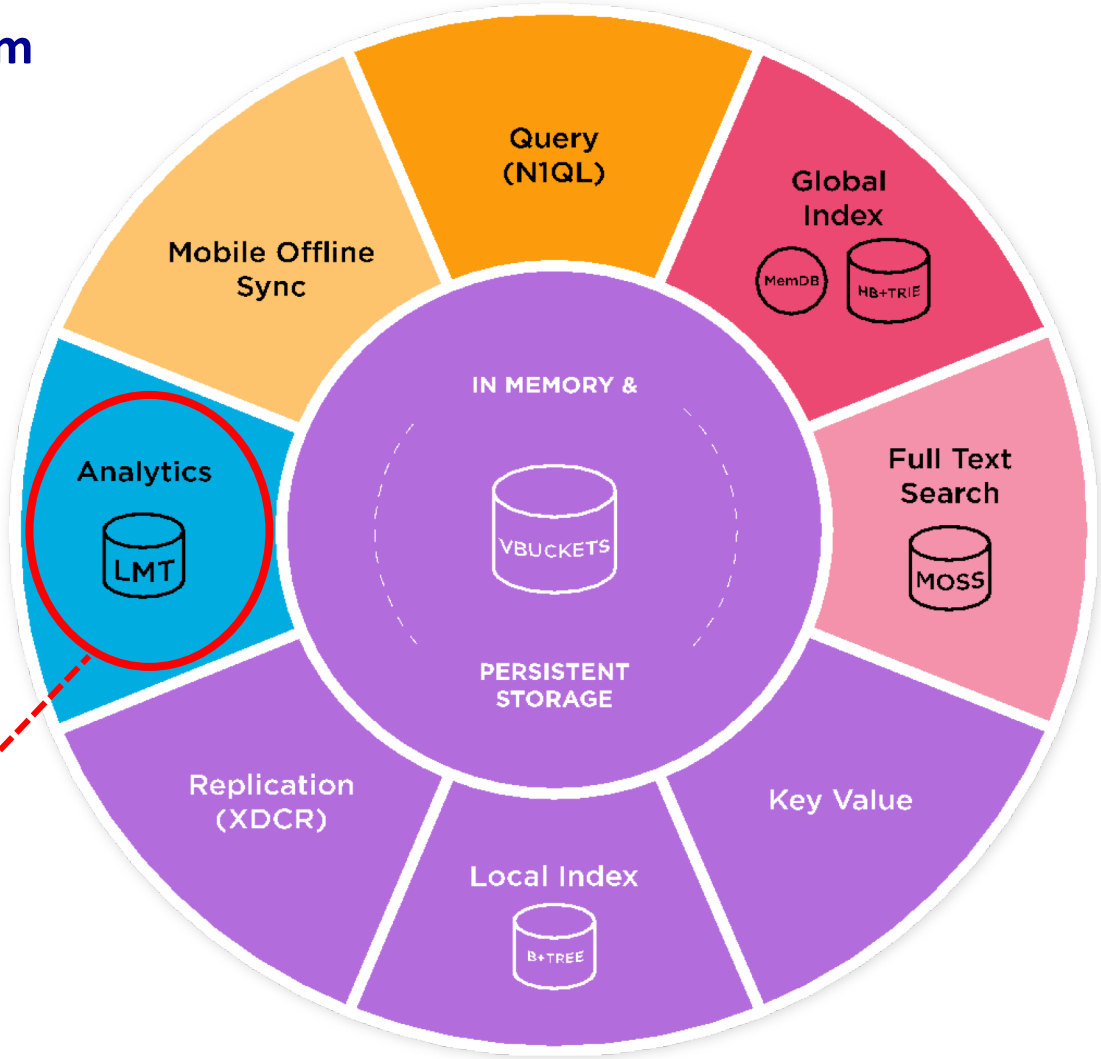
Plan for the Talk

- A Bit of Personal History
- AsterixDB System Overview
- History, Challenges, and Experiences
- Why Build Systems in Academia?
- Initial Commercial Impact
- A Balancing Act
- Closing Thoughts

Commercial Use: NoSQL 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***

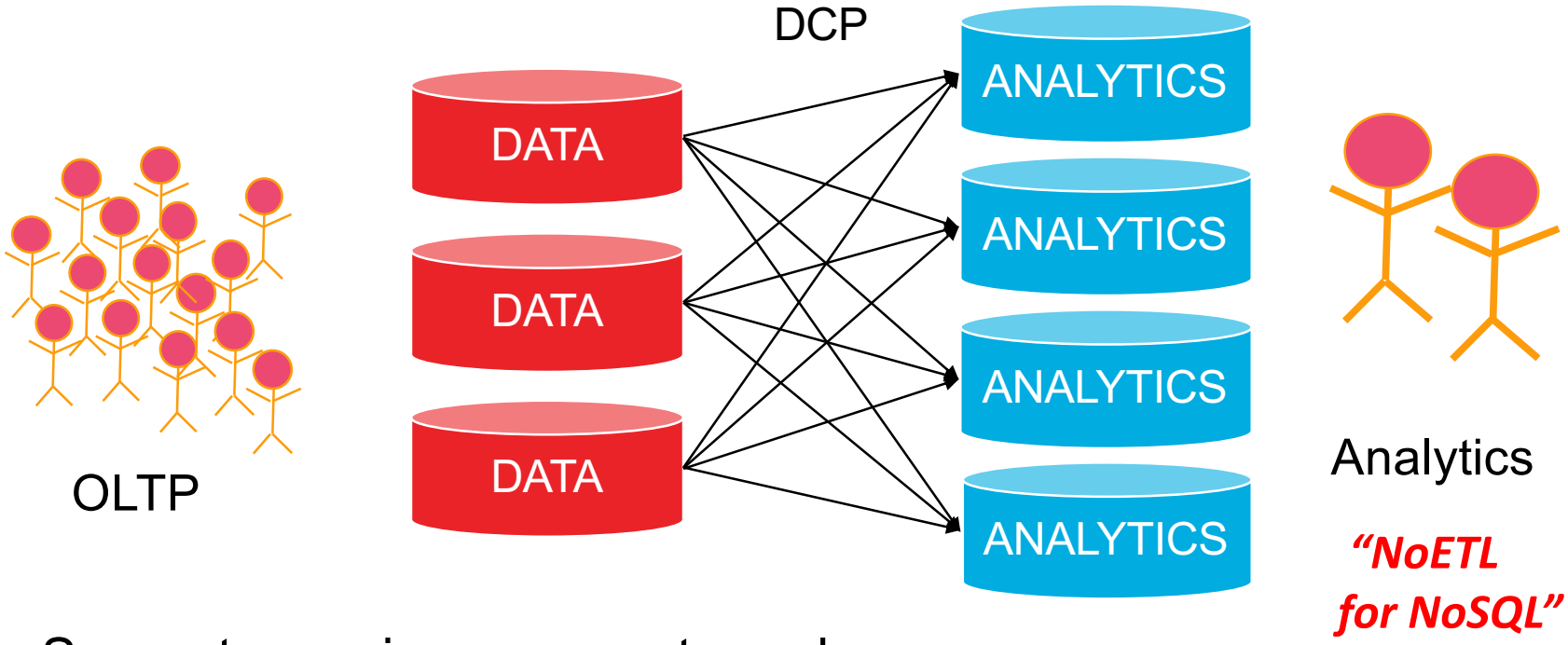


Apache AsterixDB and Couchbase

- Introduced by a mutual friend in 2015 or so
  - Couchbase had evolved a key-value cache/store into a scalable document DBMS with a SQL-inspired query language (N1QL)
    - Target applications wanted low latency for many small queries
  - AsterixDB had a scalable document DBMS with an XQuery-inspired query language (AQL, oops ☺)
    - Target applications wanted parallel evaluation of large queries
  - *A Reese's "peanut butter & chocolate" opportunity*
    - Common vision of where database systems should go next – relax the relational rules (1NF, schemas) but keep the queries and optimization

- Eventually got to help build a team inside Couchbase
  - Couchbase engineers are major Apache AsterixDB contributors
  - They played a huge role in the migration to SQL++ (N1QL for Analytics)
  - Core engine work continues to happen completely in open source
  - Commercial value-add includes cluster dynamics and management

Couchbase Analytics Service



- Separate services, separate nodes
  - Performance isolation (HTAP-like but for JSON)
  - Separate scale-out based on OLTP / Analytics needs
  - Parallel (M:N) connectivity for performance

Plan for the Talk

- A Bit of Personal History
- AsterixDB System Overview
- History, Challenges, and Experiences
- Why Build Systems in Academia?
- Initial Commercial Impact
- A Balancing Act
- Closing Thoughts

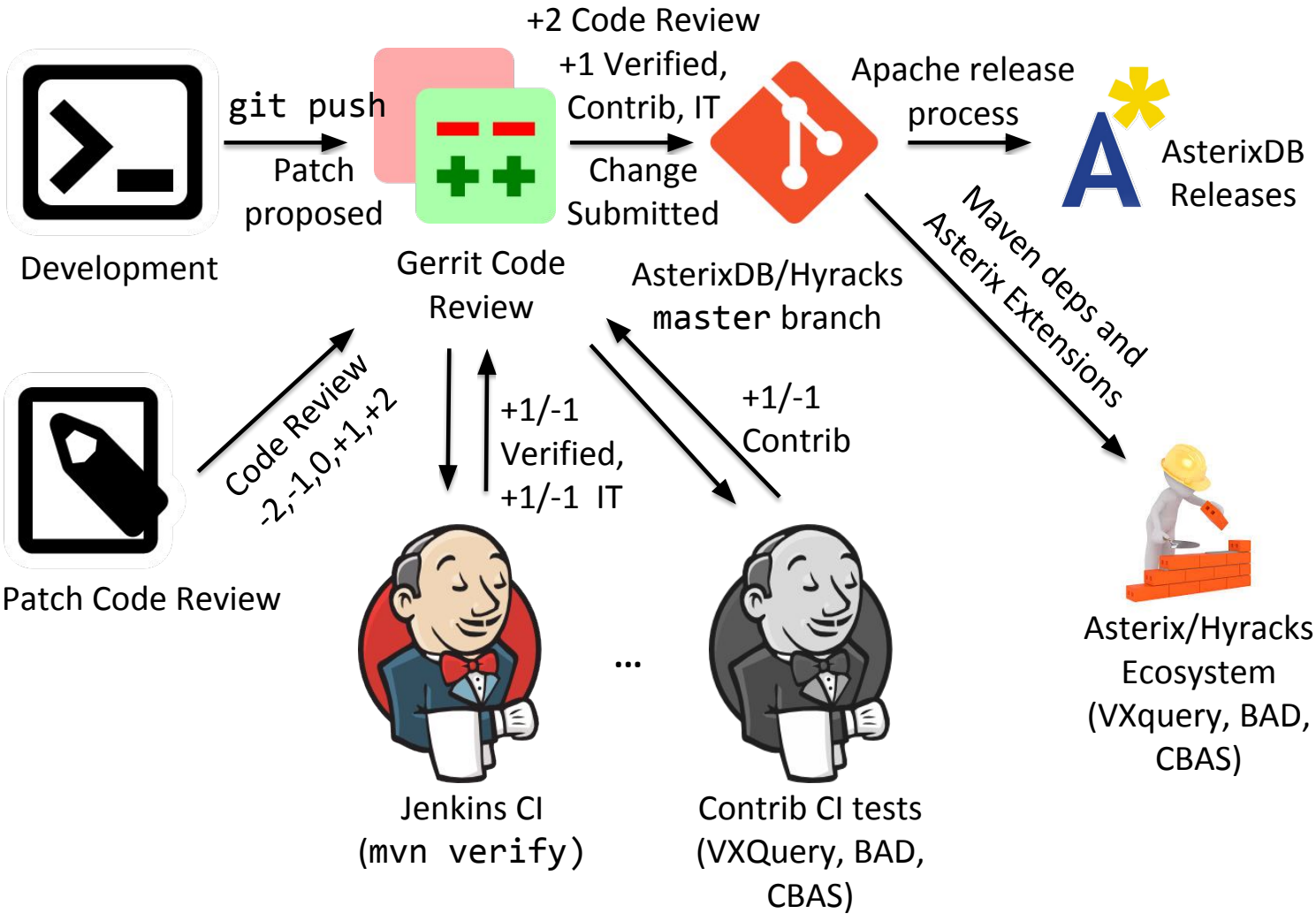
The Apache AsterixDB Balancing Act

- Now serving three user bases
  - *Apache open source community*: Apache Software Foundation "brand" implies certain quality expectations
  - *Couchbase customers*: "You get what you pay for" has a very different *meaning for them*
  - *Universities using AsterixDB:* In use for education or research at UCI, UCR, UCSD, and U Washington as well as various institutions in China, Korea, India, Saudi Arabia, Germany, and Norway
- University projects don't always survive the transition
  - No longer a "benevolent dictatorship" – took some adjustment
  - Significantly raised the quality bar for new code contributions
    - Industrial-strength code management, reviewing, testing, …
    - Reviews include a mix of industrial and non-industrial committers

The AsterixDB Balancing Act *(cont.)*

- Introduced the notion of an *extension*
  - VXQuery: utilizes Hyracks and Algebricks to provide a parallel implementation of XQuery
  - BAD (Big Active Data): extends all of AsterixDB with “active” features, e.g., channels for data-centric pub/sub
  - CBAS (Couchbase Analytics): uses AsterixDB for JSON analytics
  - Extensions participate in the automated testing process; would be nice to do more here for orthogonality (S/W research topic?)
- Want everyone to keep on winning!
  - Couchbase: big head start, with ongoing benefits from contributions from the research side of the community
  - Universities: students benefit from the wisdom/experience of the pros, and the code base is much stronger since moving to Apache
  - Apache AsterixDB community: all of the above

AsterixDB Code Management



Plan for the Talk

- A Bit of Personal History
- AsterixDB System Overview
- History, Challenges, and Experiences
- Why Build Systems in Academia?
- Initial Commercial Impact
- A Balancing Act
- Closing Thoughts

Mid-Flight for Apache AsterixDB

- DBMS platforms take a *long* time to mature
  - E.g., Postgres project (now PostgreSQL) was just getting ready to start at UC Berkeley as I was walking out the door with my Ph.D.
  - Data ownership is a whole different thing than query processing
    - Robustness and failure-tolerance
    - Cluster dynamics (accidental and on purpose)
    - Upgrades and storage formats
- AsterixDB's balancing act is just one model
  - Patterson (Berkeley) "new project every 5 years" model
  - Widom (Stanford) "pick one thing to vary, and prototype" model
  - Only time will tell how sustainable our model is…
- Closing thoughts/recommendations
  - Building software in academia is frustrating but fun/rewarding
  - I believe the "BMW model" is an important lesson for students
  - Don't afraid to be counter-cultural if that's what you believe in!

Questions?

- Asterix UCI/UCR research project home
  - http://asterix.ics.uci.edu/
- Apache AsterixDB community home
  - http://asterixdb.apache.org/