AsterixDB
more engine less trunk

Big Data Storage:
Should We Pop the (Software) Stack?

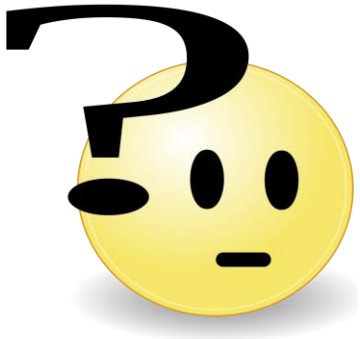
Michael Carey
Information Systems Group
CS Department
UC Irvine

UCIrvine
http://isg.ics.uci.edu

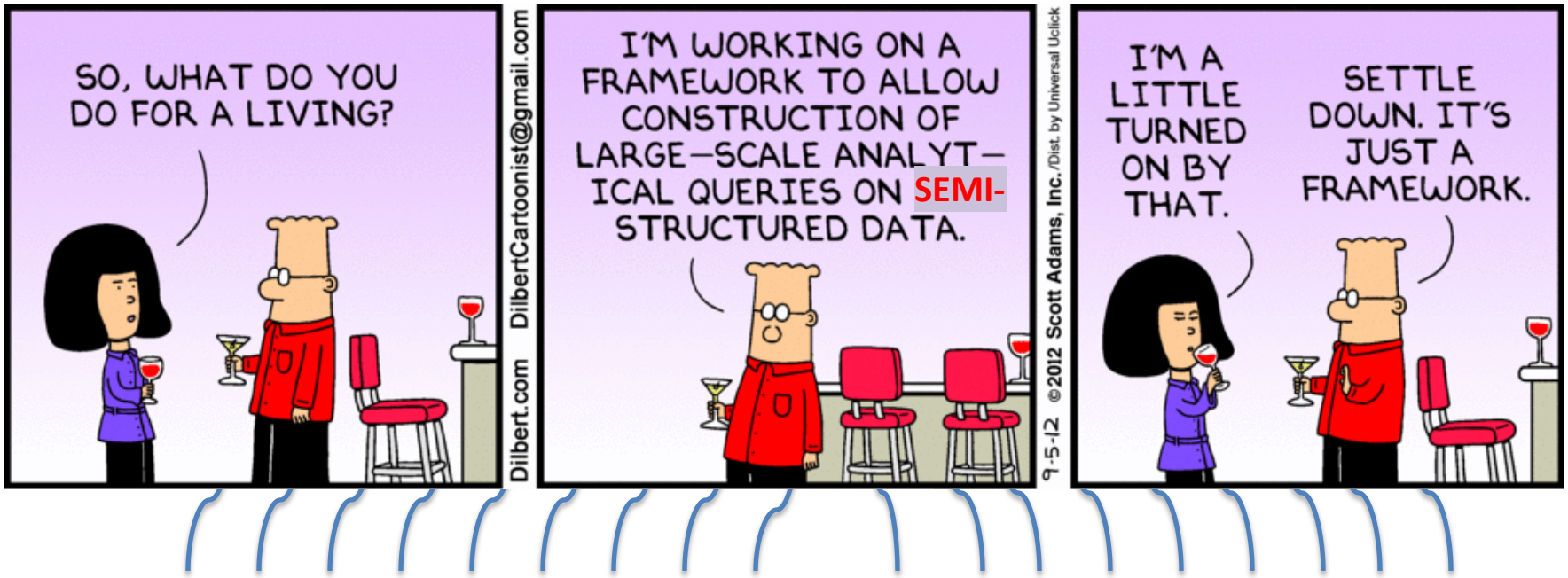
#AsterixDB

Rough Topical Plan

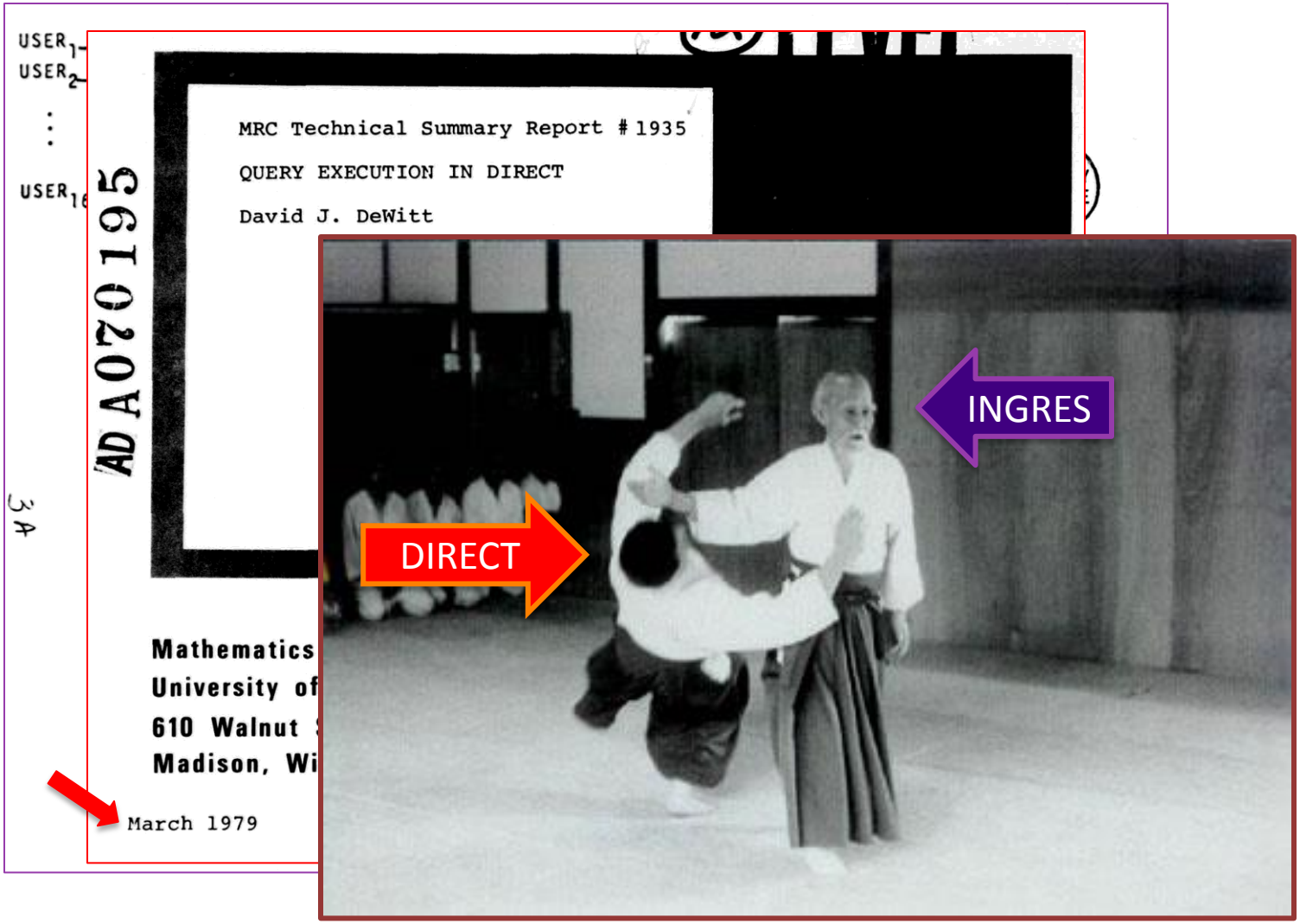
- Background and motivation (quick!)
- Big Data storage landscape (satellite view ☺)
  - Two points of view (plus cloudy skies)
- AsterixDB: a next-generation ***BDMS***
  - What we're doing (plus hedging our bets)
- Research plan, Q&A, and RFI...



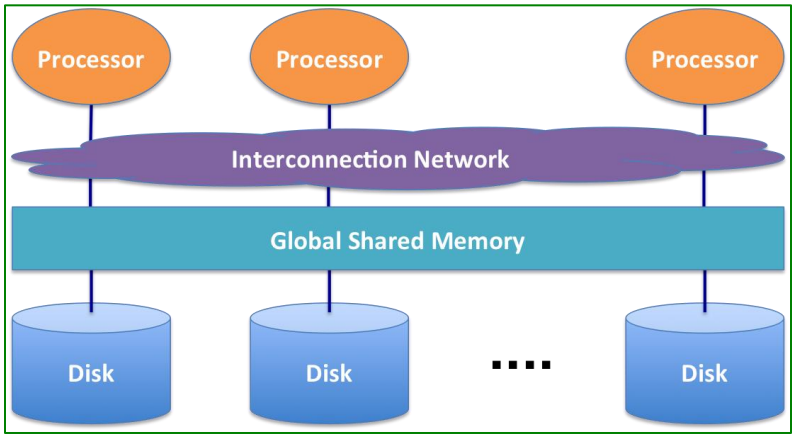
Big Data: It's Everywhere…



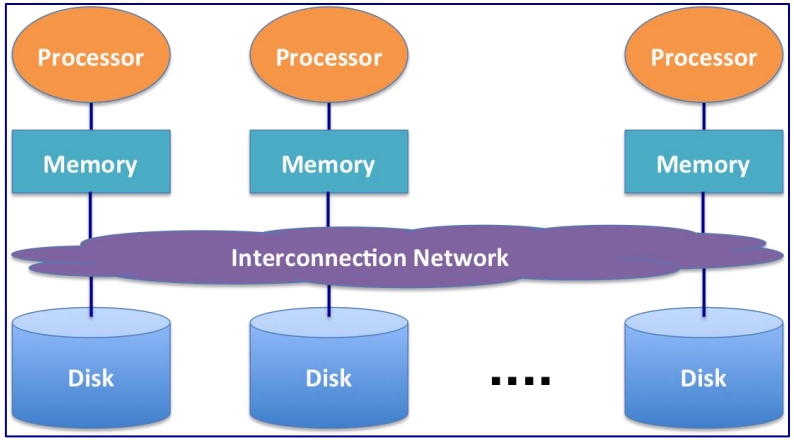
Ancient DB History I: ***DIRECT***



DB History II: *Shared What?* Wars

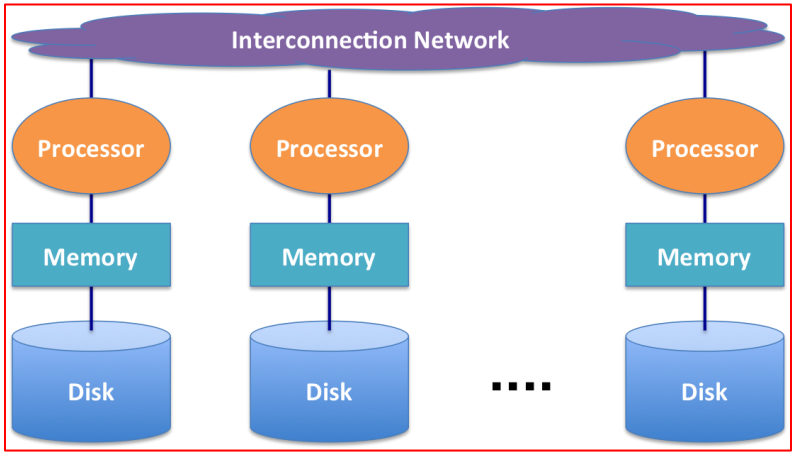


Shared-everything



Shared-disk

Shared-nothing

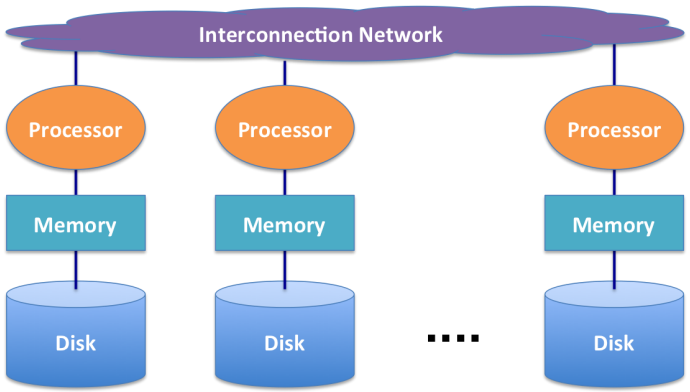


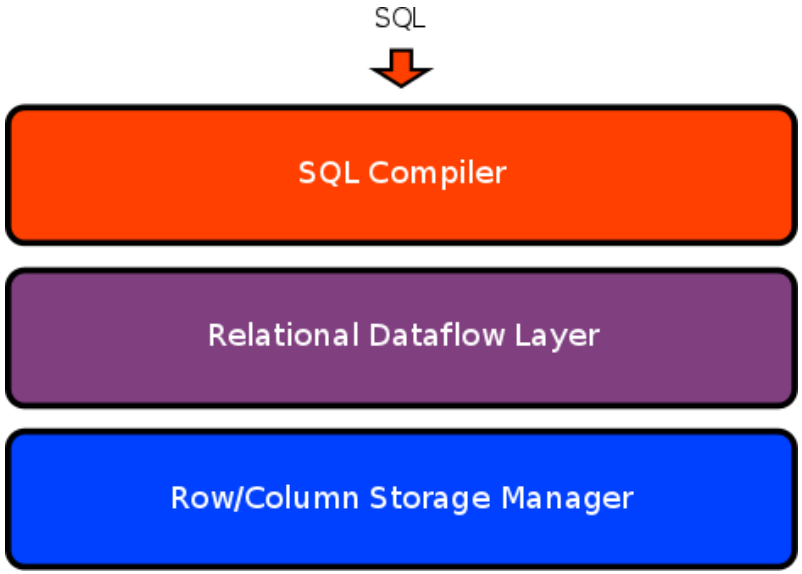
(1980's)

?

Big Data in the *Database* World

- Enterprise data warehouses
  - 1980's: Shared-nothing parallel DBMSs
  - 2000's: Enter new players (Netezza, Aster Data, DATAllegro, Greenplum, Vertica, ParAccel, ...)
- Scalable OLTP
  - 1980's: Tandem's NonStop SQL



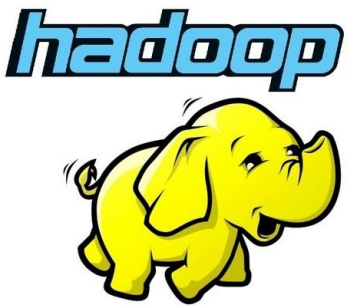


Notes:

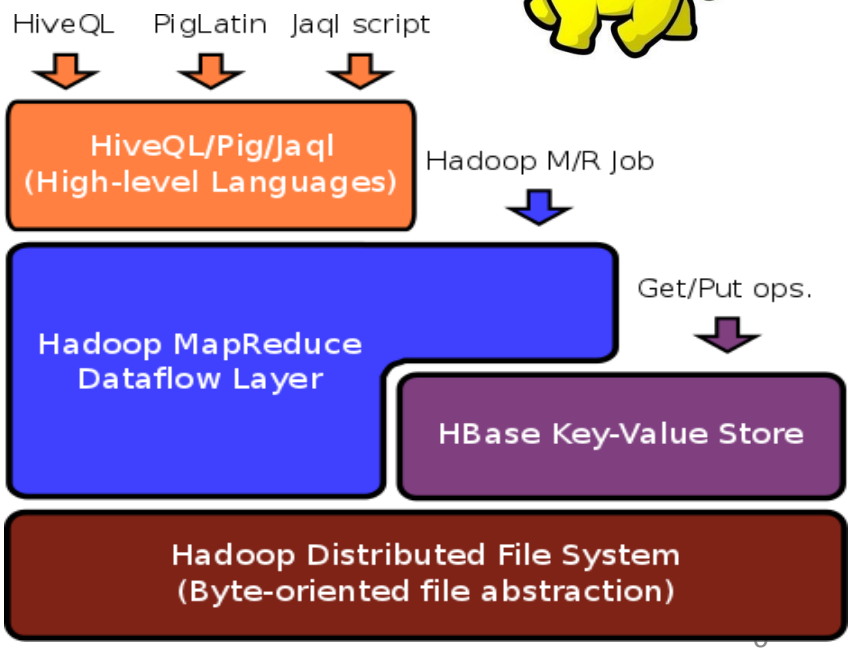
- One storage manager per machine
- Upper layers orchestrate query execution
- ***One way in/out:*** through the SQL door

Later: Big Data in the ***Systems*** World

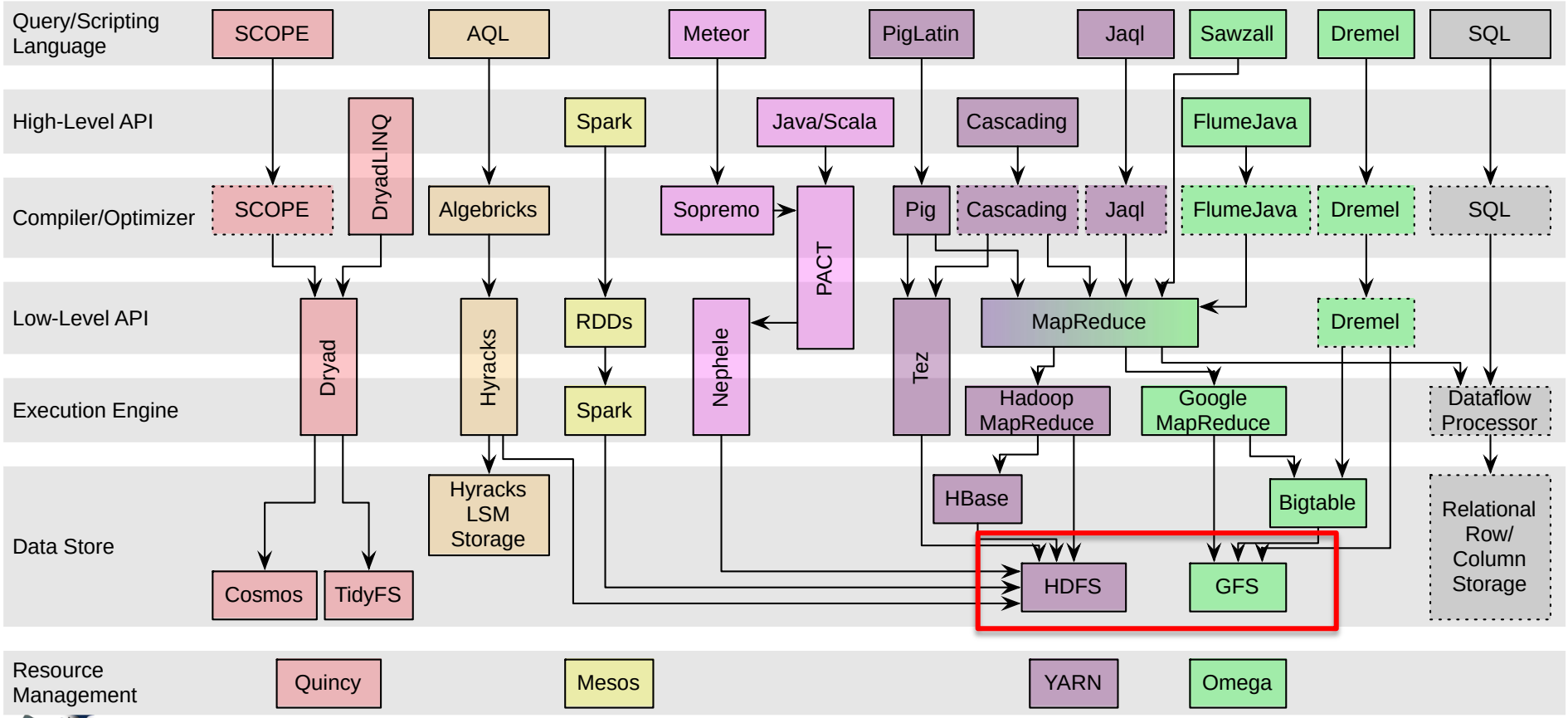
- Out to index and query the Web, Google laid a new foundation in the early 2000's
  - Google File System (GFS): Files spanning many machines with 3-way replication
  - MapReduce (MR): *"Parallel programming for dummies"* (UDFs + parallel framework)



- Yahoo!, FB, et al read the papers
  - HDFS and Hadoop MapReduce
  - Declarative HLLs: Pig, Hive, ...
  - HLLs now heavily preferred to MR
- Also key-value stores ("NoSQL")
  - Social sites, online games, …
  - BigTable/HBase, Dynamo/Cassandra, MongoDB, …

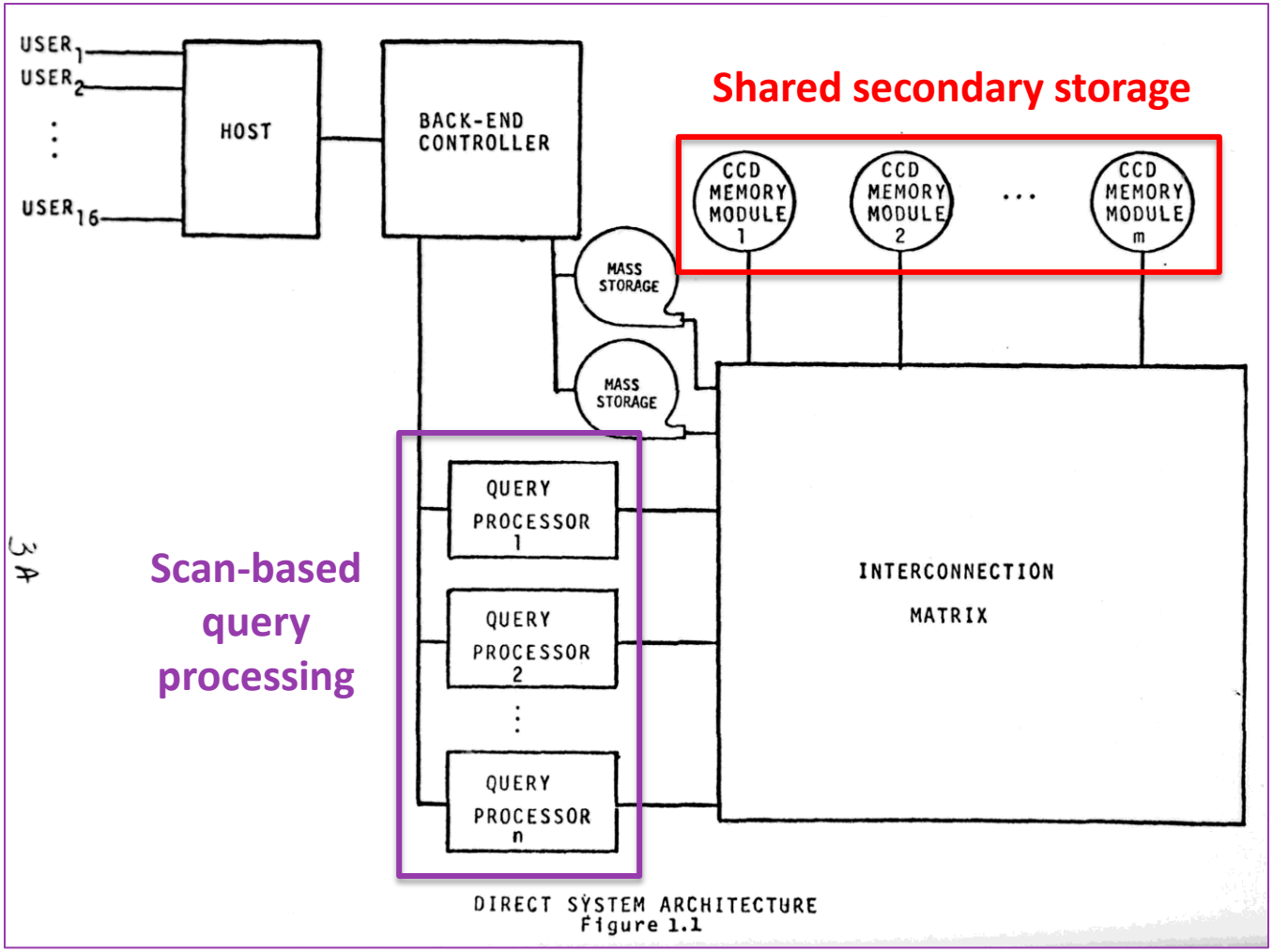


No Shortage of "NoSQL" Big Data Analysis Platforms!



UCIrvine
http://isg.ics.uci.edu

Remember History? *(DIRECT)*



DIRECT SYSTEM ARCHITECTURE
Figure 1.1

One More Bit of History

the wrong service is provided or
severe performance problems
. When possible, we offer some

ission to copy without fee all or part of
material is granted provided that the cop-
re not made or distributed for direct
mercial advantage, the ACM copyright
e and the title of the publication and its
appear, and notice is given that copying
permission of the Association for Com-

we look at the services provided by buffer pool management; the file system; scheduling, process management, and interprocess communication; and consistency control. We then conclude with a discussion of the merits of including all files in a paged virtual memory.

The examples in this paper are

this cache. A file read (e.g., rea
in Figure 1) returns data dir
from a block in the cache, if poss
otherwise, it causes a block t
"pushed" to disk and replaced b
desired block. In Figure 1 we s
block Y being pushed to make r
for block X. A file write sir
moves data into the cache; at s

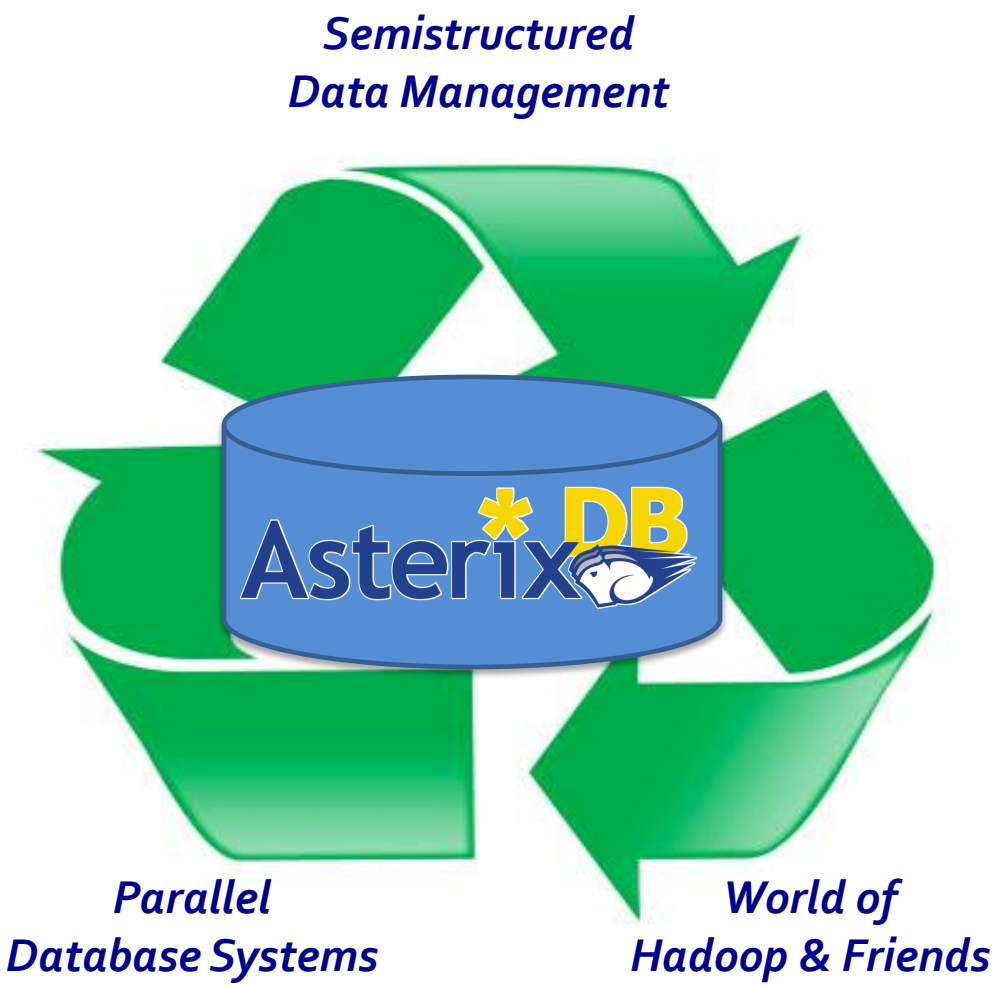
ting system services are examined
pplicability to support of database
hese services include buffer pool
em; scheduling, process manage-
communication; and consistency

system services in many existing systems are either too slow or inappropriate. Current DBMSs usually provide their own and make little or no use of those offered by the operating system. It is important that future operating system designers become more sensitive to DBMS needs.

A DBMS would prefer a small efficient operating system with only

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"

⋮

ASTERIX Data Model (ADM)

```
create dataverse TinySocial;
use dataverse TinySocial;

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 type EmploymentType as open {
   organization-name: string,
   start-date: date,
   end-date: 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)

```
create dataverse TinySocial;
use dataverse TinySocial;

create type MugshotUserType as {
   id: int32
}
```

```
create type EmploymentType as open {
   organization-name: string,
   start-date: date,
   end-date: 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***

Other DDL Features

```
create index msUserSinceIdx on MugshotUsers(user-since);
create index msTimestampIdx on MugshotMessages(timestamp);
create index msAuthorIdx  on MugshotMessages(author-id) type btree;
create index msSenderLocIndex on MugshotMessages(sender-location) type rtree;
create index msMessageIdx on MugshotMessages(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"="{hostname}://{path}"), ("format"="delimited-text"), ("delimiter"="|"));

create feed socket_feed using socket_adaptor
    (("sockets"="{address}:{port}"), ("addressType"="IP"),
     ("type-name"="MugshotMessageType"), ("format"="adm"));
connect feed socket_feed to dataset MugshotMessages;
```

External data highlights:

- Equal opportunity access
- "Keep everything!"
- Data ingestion, *not* streams
- Queries unchanged

ASTERIX Query Language (AQL)

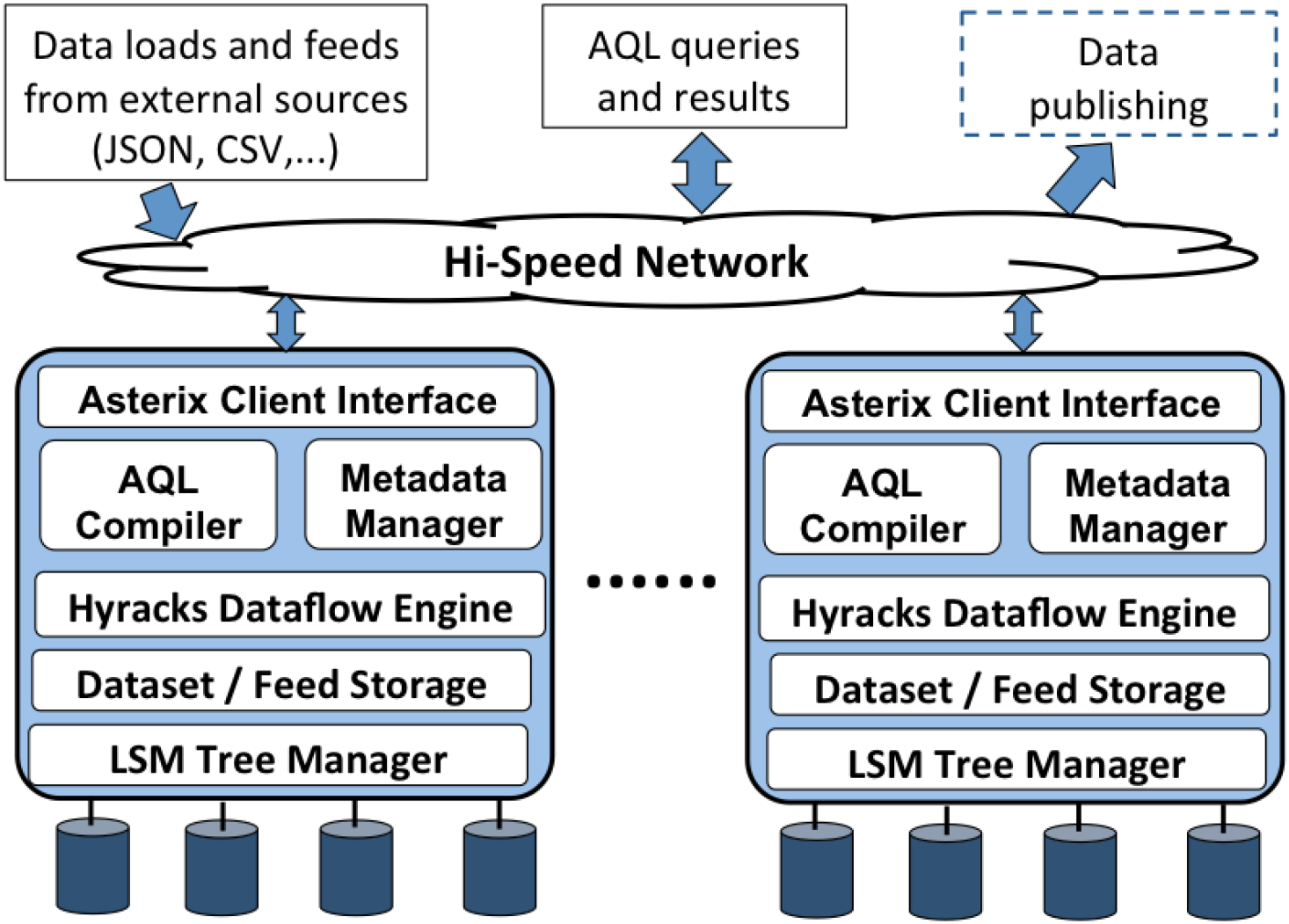
- *Ex:* Identify active users and group/count them by country:

```
with $end := current-datetime()
with $start := $end - duration("P30D")
from $user in dataset MugshotUsers
where some $logrecord in dataset AccessLog
  satisfies $user.alias = $logrecord.user
  and datetime($logrecord.time) >= $start
  and datetime($logrecord.time) <= $end
group by $country := $user.address.country with $user
select {
  "country" : $country,
  "active users" : count($user)
}
```

***AQL highlights*:**

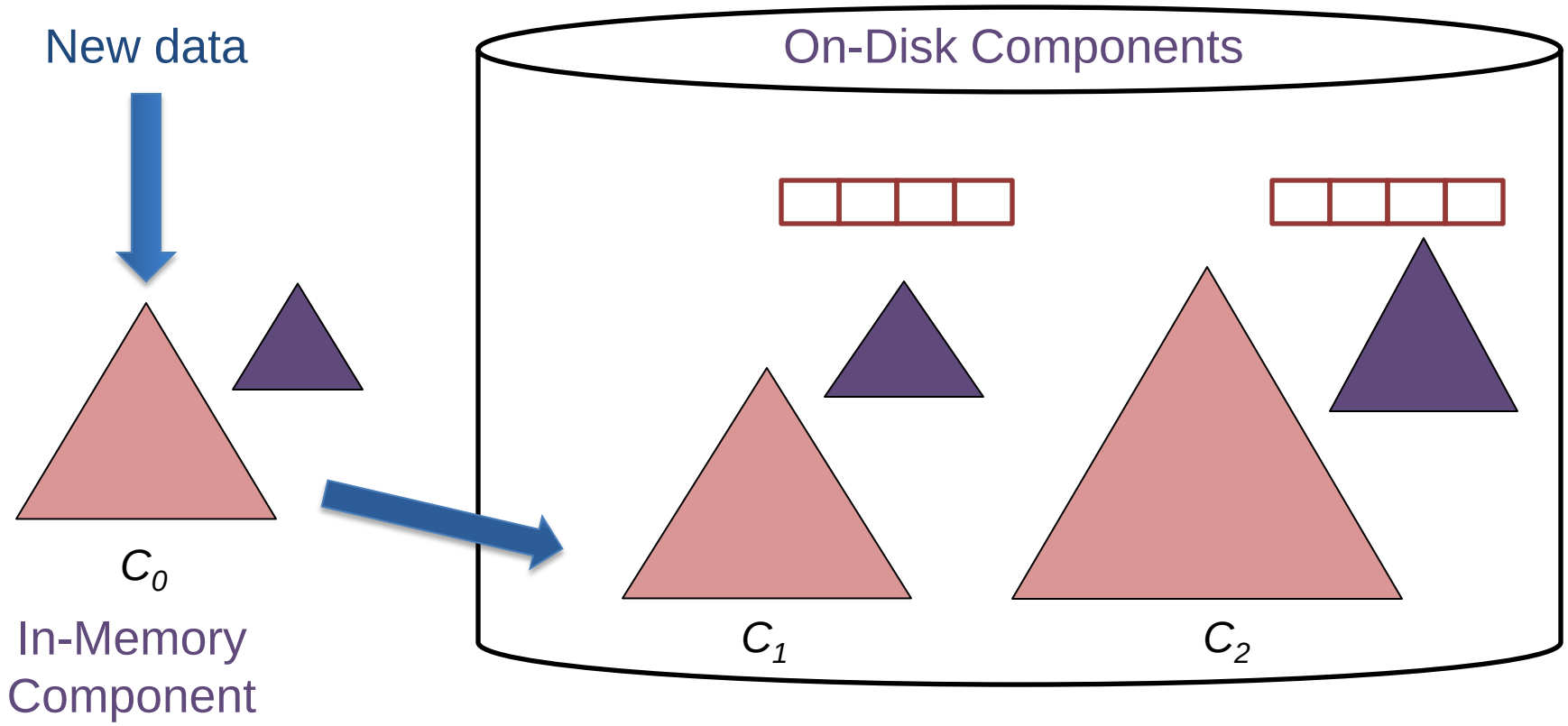
- Lots of other features (see website!)
- Spatial predicates and aggregation
- Set-similarity matching
- And plans for more…

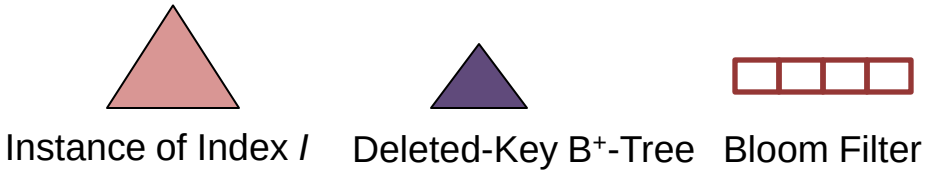
AsterixDB System Overview



UCIrvine
http://isg.ics.uci.edu

Local LSM-Based Storage & Indexes





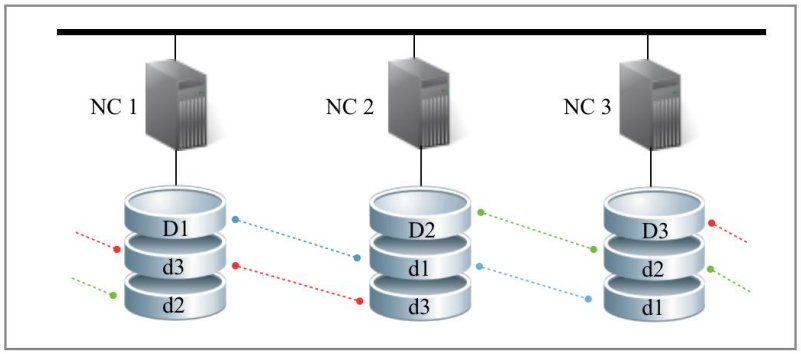
LSM-ified Indexes:

- B+ trees
- R trees (secondary)
- Inverted (secondary)

Distributed Storage in AsterixDB

- Hash-partitioned, shared-nothing, local drives
  - Partitioning based on primary key (hashing)
  - Secondary indexes local to, and consistent with, corresponding primary partitions (all LSM-based)
- Also offer external dataset feature (for HDFS)
  - Multiple (Hive) formats, secondary index support
  - Index partitions co-located with data (if possible)
  - Developed for space and “IT comfort” reasons

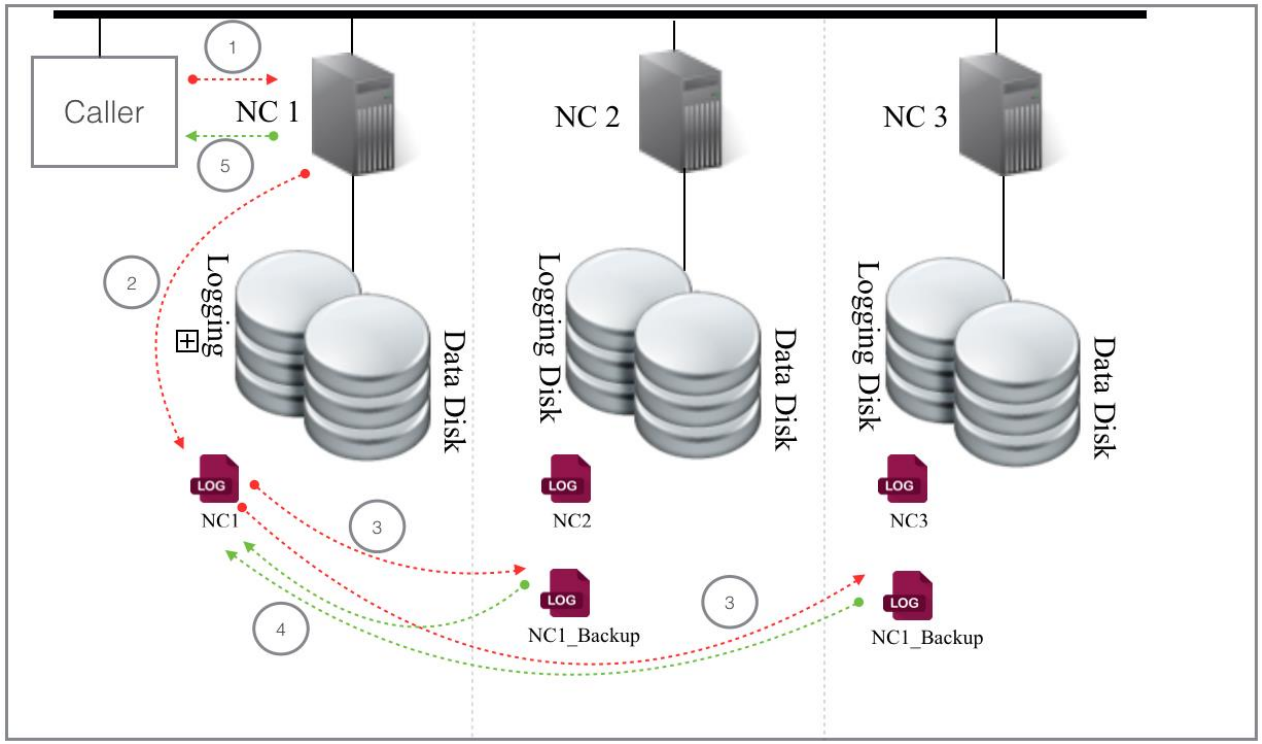
Data Replication in AsterixDB (WIP)



Chained Declustering

Log-Based Replication

(synchronous, recovery-only copies kept)



Hedging Our Bets

- We're currently porting our LSM-based storage system to also work on *top* of HDFS (and YARN)
  - Might somehow feel more "comforting" (and/or "environmentally friendly") to Big Data IT shops
  - Another path to replication and high availability
- Interesting experiments lie ahead
  - Revisit Stonebraker-like OS issues (modern version)
  - Bake-off: Distributed record management *vs.* DFS
  - Just how well does HDFS do *w.r.t.* locality of writes?

What About the Cloud?

- Computing may be elastic, but data is not...!
  - Native storage → hard to expand & contract
  - Seems to make the argument for a shared-disk-like approach based on cloud storage facilities
- Experimentation is needed
  - E.g., Google persistent disks (in Google Cloud)?
  - Performance implications will be interesting to explore...

Some AsterixDB Use Cases

- Recent/projected use case areas include:
  - Behavioral science
  - Social data analytics
  - Cell phone event analytics
  - Education (MOOC analytics)
  - Power usage monitoring
  - Public health
  - Cluster management log analytics
- Let's take a quick pick at the first two…
  - Time permitting!

Current Status

AsterixDB
more engine less trunk

- 4 year initial NSF project (250+ KLOC @ UCI/UCR)
- AsterixDB BDMS is 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
  - Rich set of data types (including text, time, location)
  - Fuzzy and spatial query processing
  - NoSQL-like transactions (for inserts/deletes)
  - Data feeds and external indexes will appear soon
- Now in Apache incubation mode!

For More Info

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

Open source code base:

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

The New Kid on the Block!

http://asterixdb.ics.uci.edu