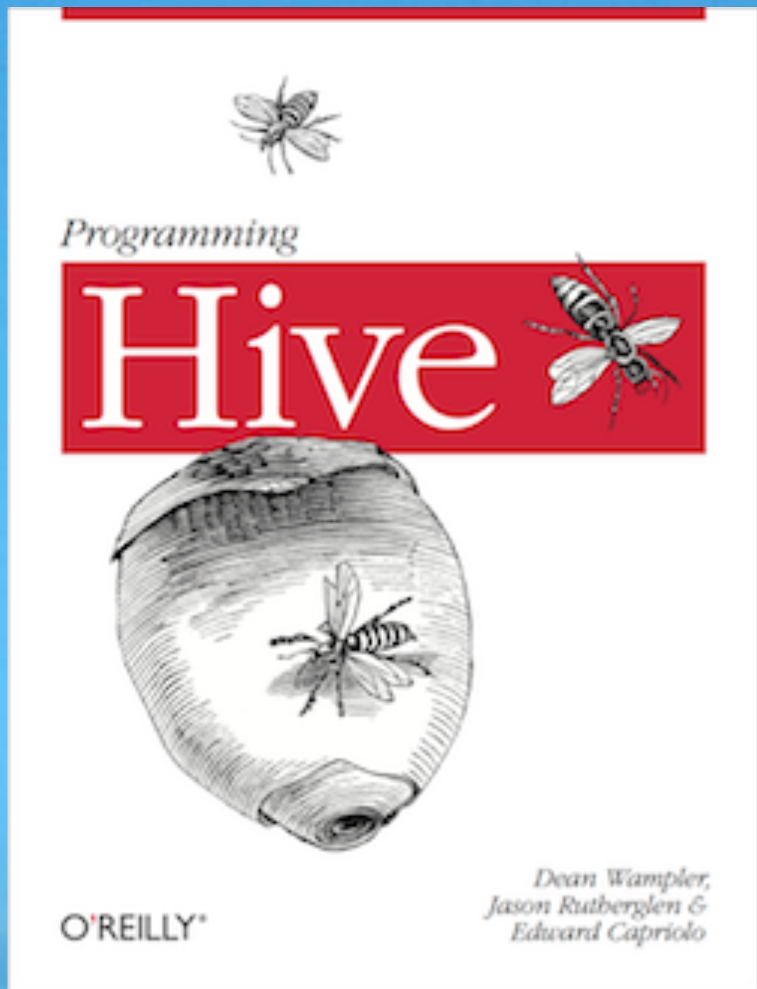
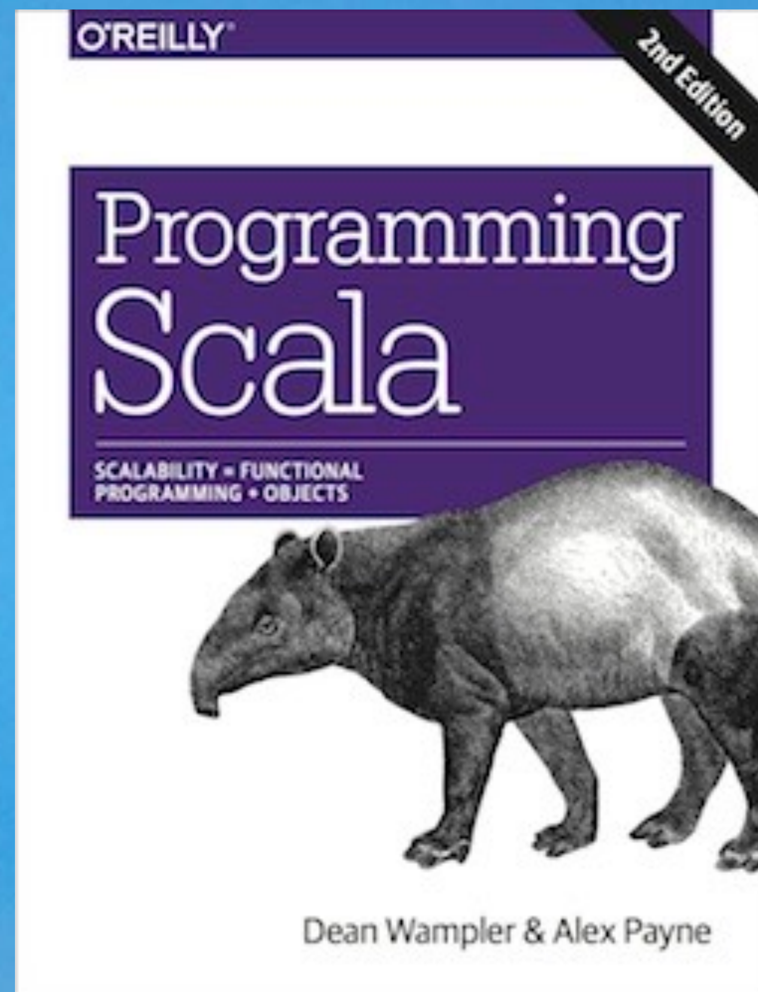
SQL Strikes Back

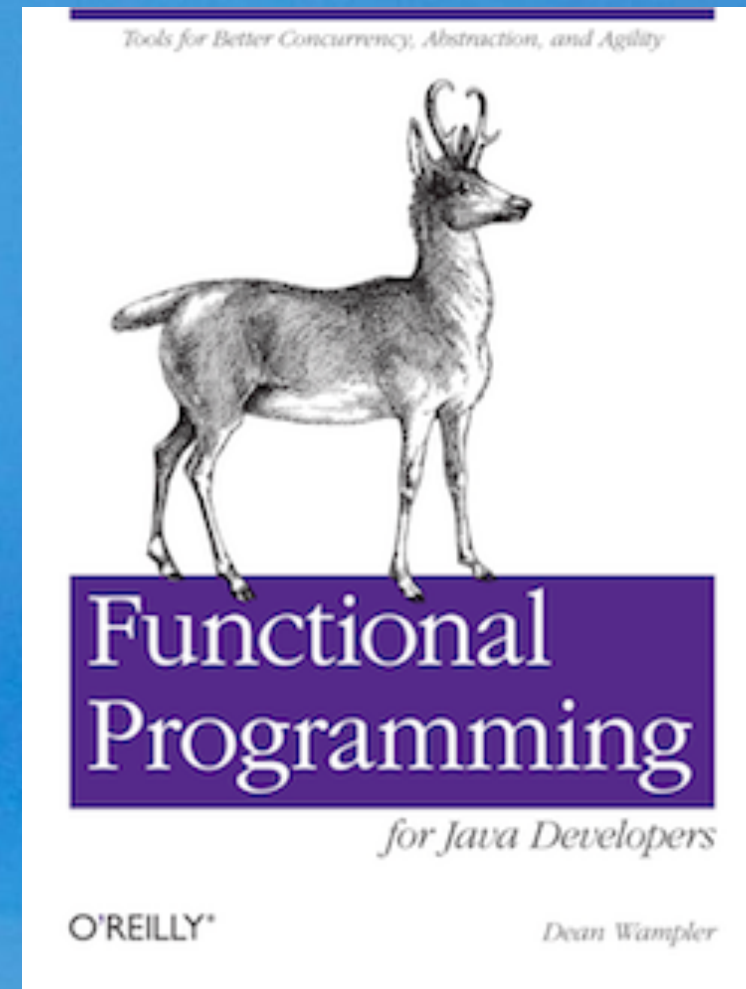
Recent Trends in Data Persistence and Analysis

Codemesh 2014
November 4, 2014
Dean Wampler, Ph.D
dean.wampler@typesafe.com

Photo: Before dawn above the Western USA







dean.wampler@typesafe.com
polyglotprogramming.com/talks
@deanwampler

Typesafe

Tuesday, November 4, 14

Typesafe provides products and services for building Reactive, Big Data applications

typesafe.com/reactive-big-data

Tuesday, November 4, 14

If you were at Eric Raymond's (@coderoshi's) talk...

My cat Oberon.

Tuesday, November 4, 14

Three trends to organizer our thinking...

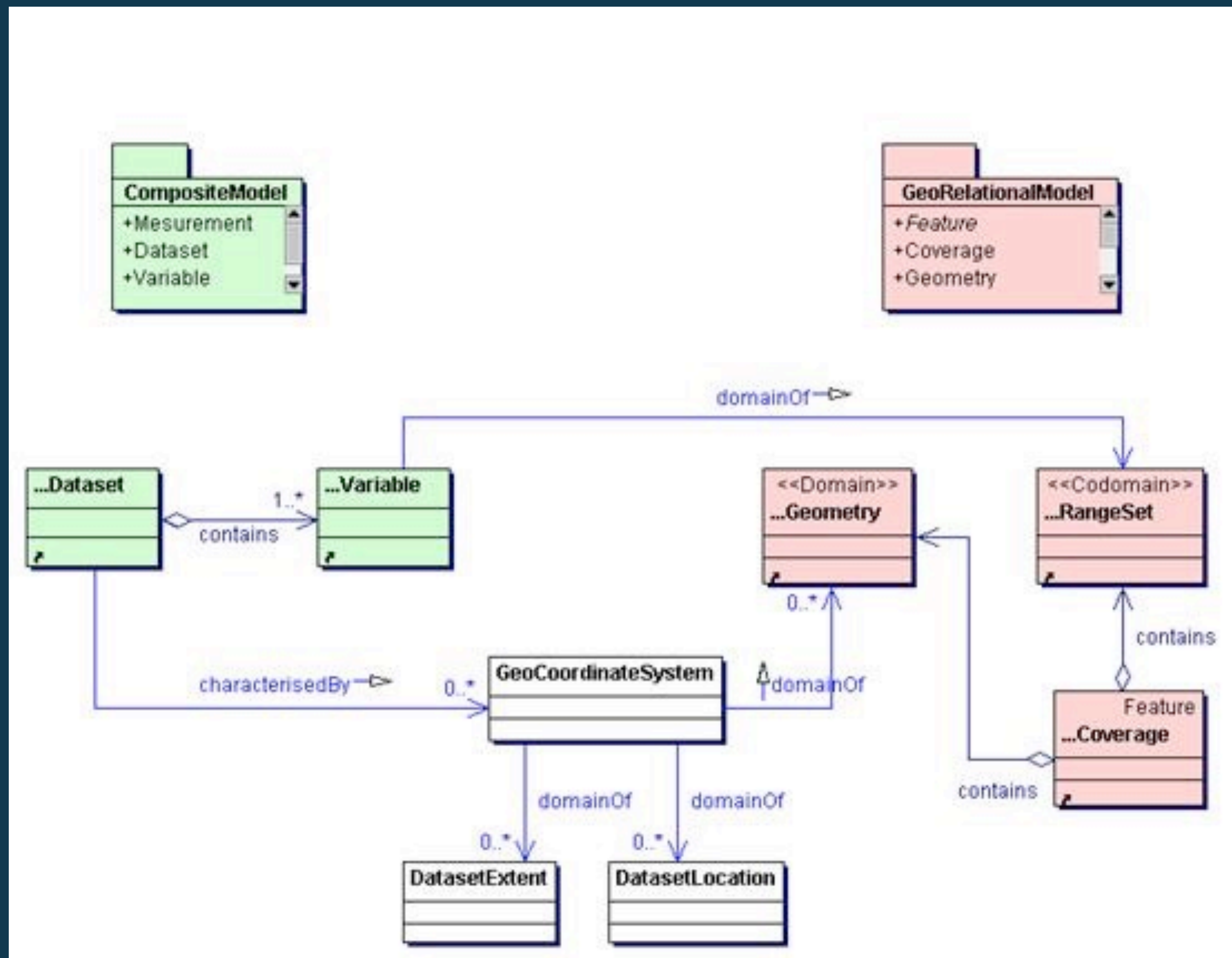
Photo: Dusk over the American Midwest, in Winter

Data Size ⬆

Tuesday, November 4, 14

Data volumes are obviously growing… rapidly.
Facebook now has over 600PB (Petabytes) of data in Hadoop clusters!

Formal Schemas ⬇



Tuesday, November 4, 14

There is less emphasis on "formal" schemas and domain models, i.e., both relational models of data and OO models, because data schemas and sources change rapidly, and we need to integrate so many disparate sources of data. So, using relatively-agnostic software, e.g., collections of things where the software is more agnostic about the structure of the data and the domain, tends to be faster to develop, test, and deploy. Put another way, we find it more useful to build somewhat agnostic applications and drive their behavior through data...

Data-Driven Programs ⬆

Tuesday, November 4, 14

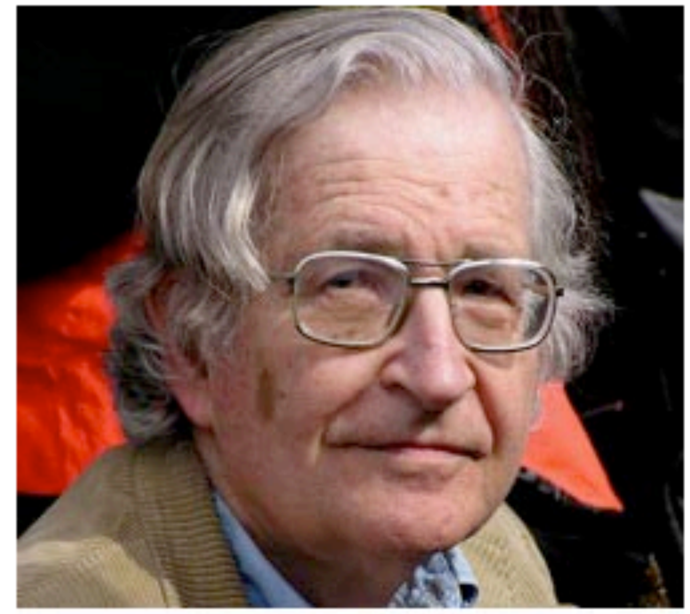
This is the 2nd generation "Stanley", the most successful self-driving car ever built (by a Google-Stanford) team. Machine learning is growing in importance. Here, generic algorithms and data structures are trained to represent the "world" using data, rather than encoding a model of the world in the software itself. It's another example of generic algorithms that produce the desired behavior by being application agnostic and data driven, rather than hard-coding a model of the world. (In practice, however, a balance is struck between completely agnostic apps and some engineering towards for the specific problem, as you might expect...)

Probabilistic Models vs. Formal Grammars

Norvig vs. Chomsky and the Fight for the Future of AI

KEVIN GOLD

When the Director of Research for Google compares one of the most highly regarded linguists of all time to Bill O'Reilly, you know it is *on*. Recently, Peter Norvig, Google's Director of Research and co-author of the most popular artificial intelligence textbook in the world, wrote a webpage extensively criticizing Noam Chomsky, arguably the most influential linguist in the world. Their disagreement points to a revolution in artificial intelligence that, like many revolutions, threatens to destroy as much as it improves. Chomsky, one of the old guard, wishes for an elegant theory of intelligence and language that looks past human fallibility to try to see simple structure underneath. Norvig, meanwhile, represents the new philosophy: truth by statistics,



Chomsky photo by Duncan Rawlinson and his Online Photography School. Norvig photo by Peter Norvig

Typesafe

Tuesday, November 4, 14

An interesting manifestation of the last two points is the public argument between Noam Chomsky and Peter Norvig on the nature of language. Chomsky long ago proposed a hierarchical model of formal language grammars. Peter Norvig is a proponent of probabilistic models of language. Indeed all successful automated language processing systems are probabilistic.

http://www.tor.com/blogs/2011/06/norvig-vs-chomsky-and-the-fight-for-the-future-of-ai

Why NoSQL?

Challenges

- Unprecedented data set sizes.
- Cost containment.
- Availability more important than consistency.
- Not all data is (or needs to be) relational.

These challenges (among others) drove the emergence of NoSQL.

Unprecendeted Data Set Sizes

- Existing relational databases could not easily scale to handle the sets amassed by Google, Amazon, eBay, Facebook, Yahoo!, Twitter, ...
  - At the time (the 90's), little work had been done to scale relational DBs horizontally.
  - CPUs and networks were slower, disks were smaller.

Cost Containment

- Even when existing relational databases could be scaled to handle the data sets, the resources required were often very high.
  - DevOps time.
  - # of Servers.
  - Performance.
- JBOD - Just a bunch of disks is as cheap as you can get.

JBOD, like HDFS, is the extreme end of the scale.

Availability Über Alles

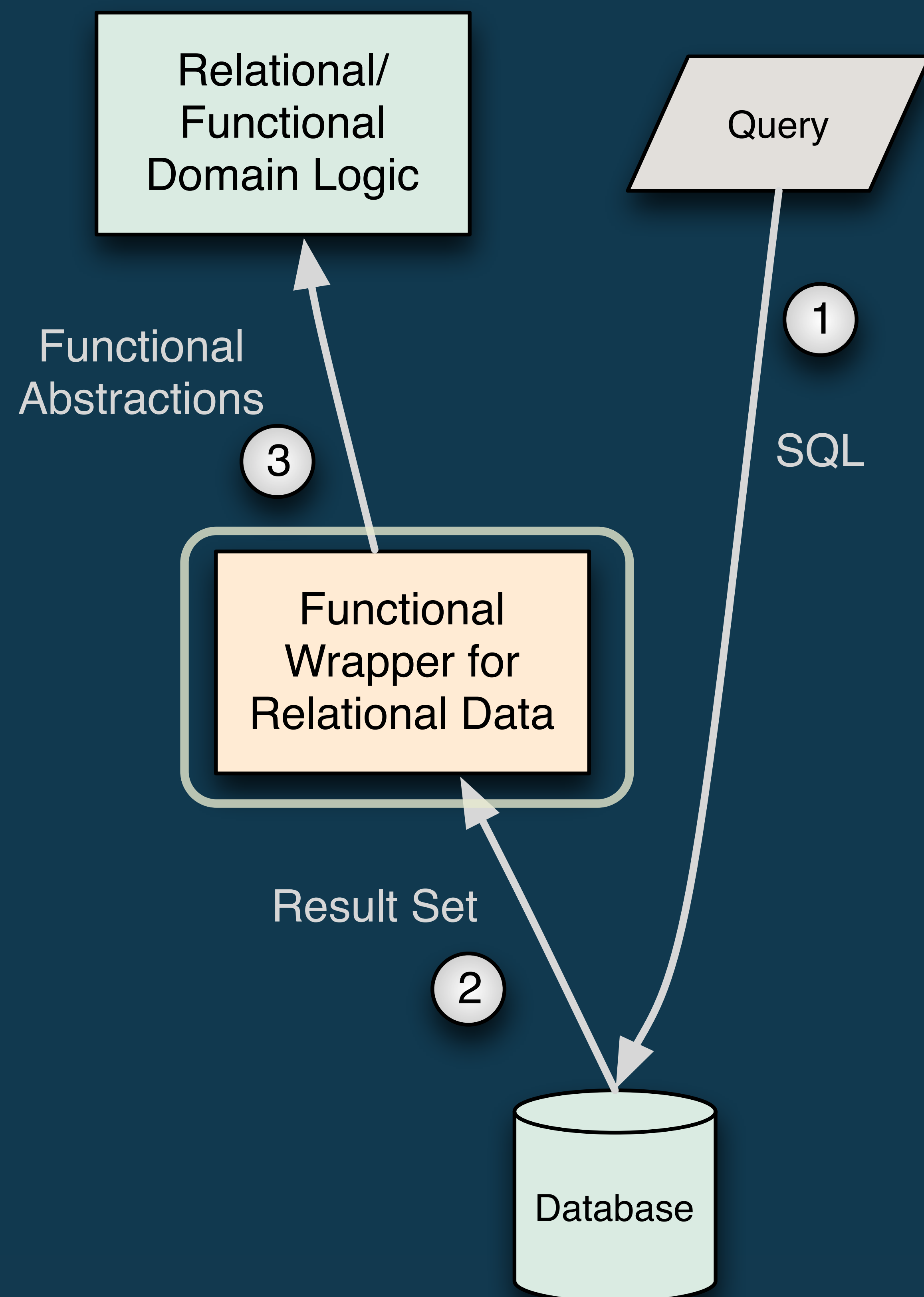
- Sometimes remaining available is more important than (immediate) consistency.
  - E.g., Amazon would rather show you a stale catalog and take your money, than wait until a network connection to the SKU database is fixed.
- Eventual Consistency - Tolerate transient inconsistencies.
  - Fix post hoc if needed.
- CAP theorem

Not All Data Is Relational

- The alternative data models are optimal (or good enough...) for many problems.
  - Key-Value store.
  - Document store (e.g., XML or JSON).
  - Column store.
- Data is free-form to structured.
  - Storing "as is" and gluing different data sets together is a core strength of Hadoop.

Architectures for NoSQL Applications

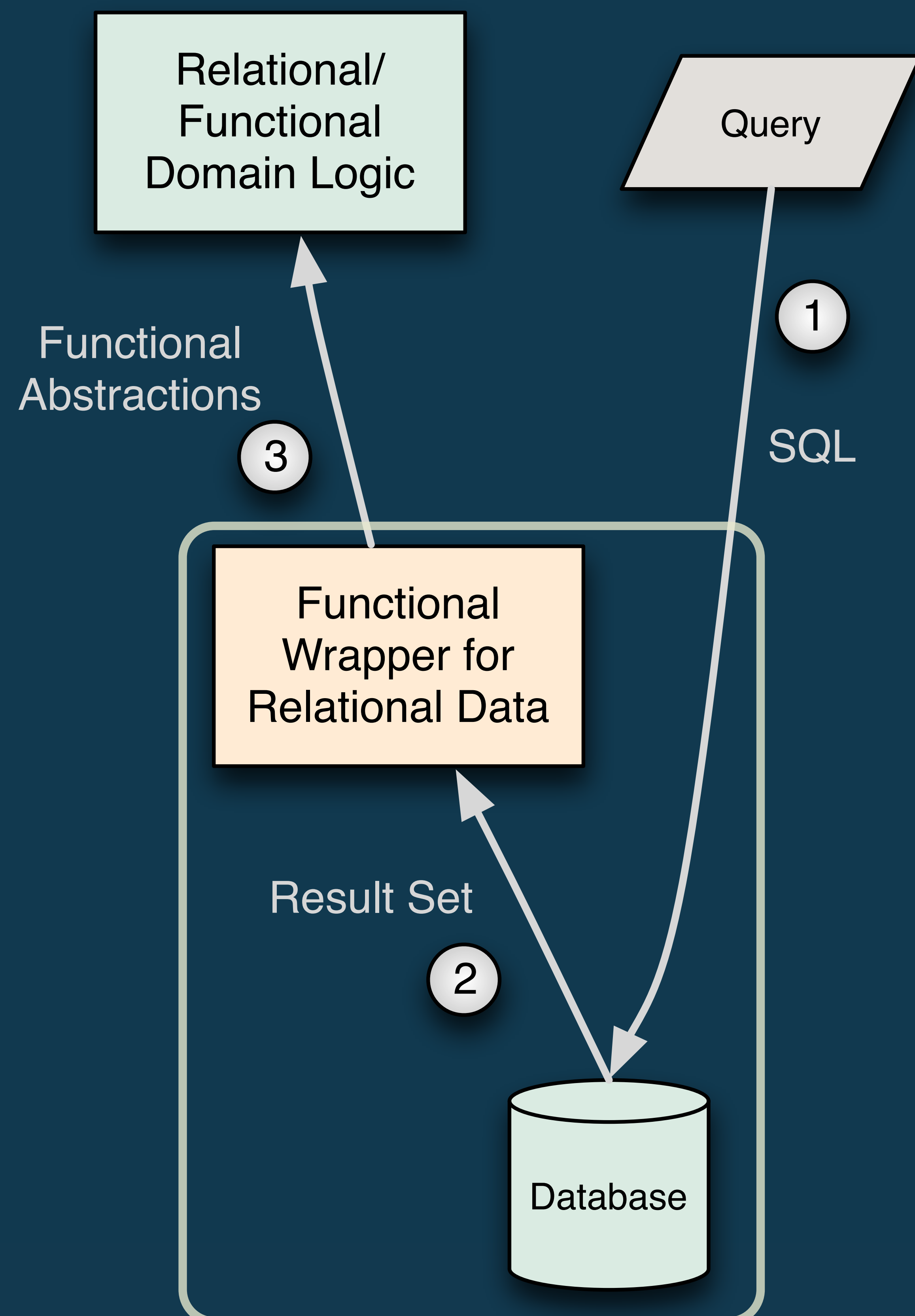
- Focus on:
  - Lists
  - Maps
  - Sets
  - Trees
  - ...



Tuesday, November 4, 14

Most of what we do with data is mathematical transformation, so we're far more productive (and runtime efficient) if we embrace fundamental data structures used throughout (lists, sets, maps, trees) and build rich transformations into those libraries, transformations that are composable to implement business logic.

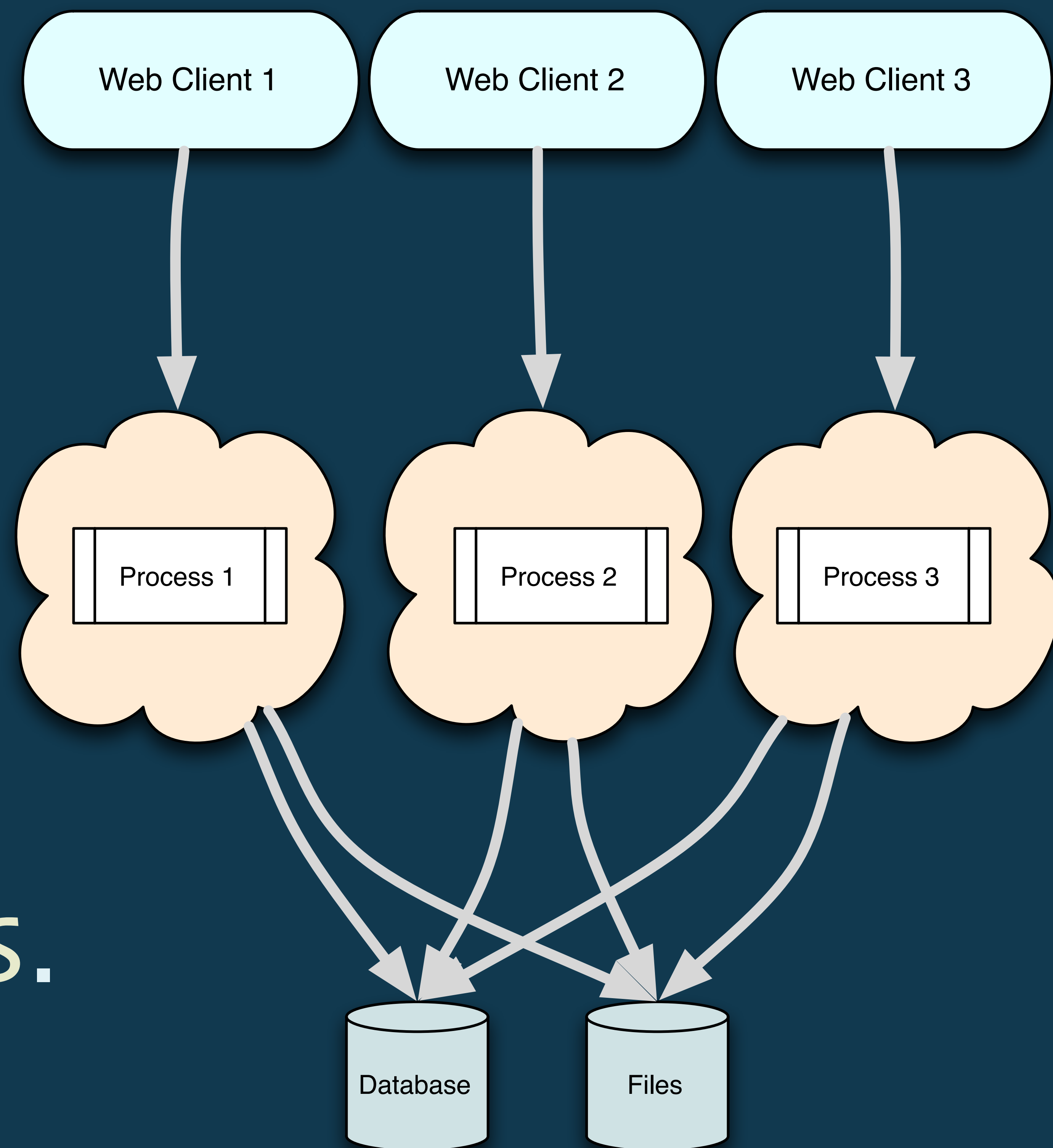
- NoSQL?
  - Cassandra, HBase
  - Riak, Redis
  - MongoDB
  - ...



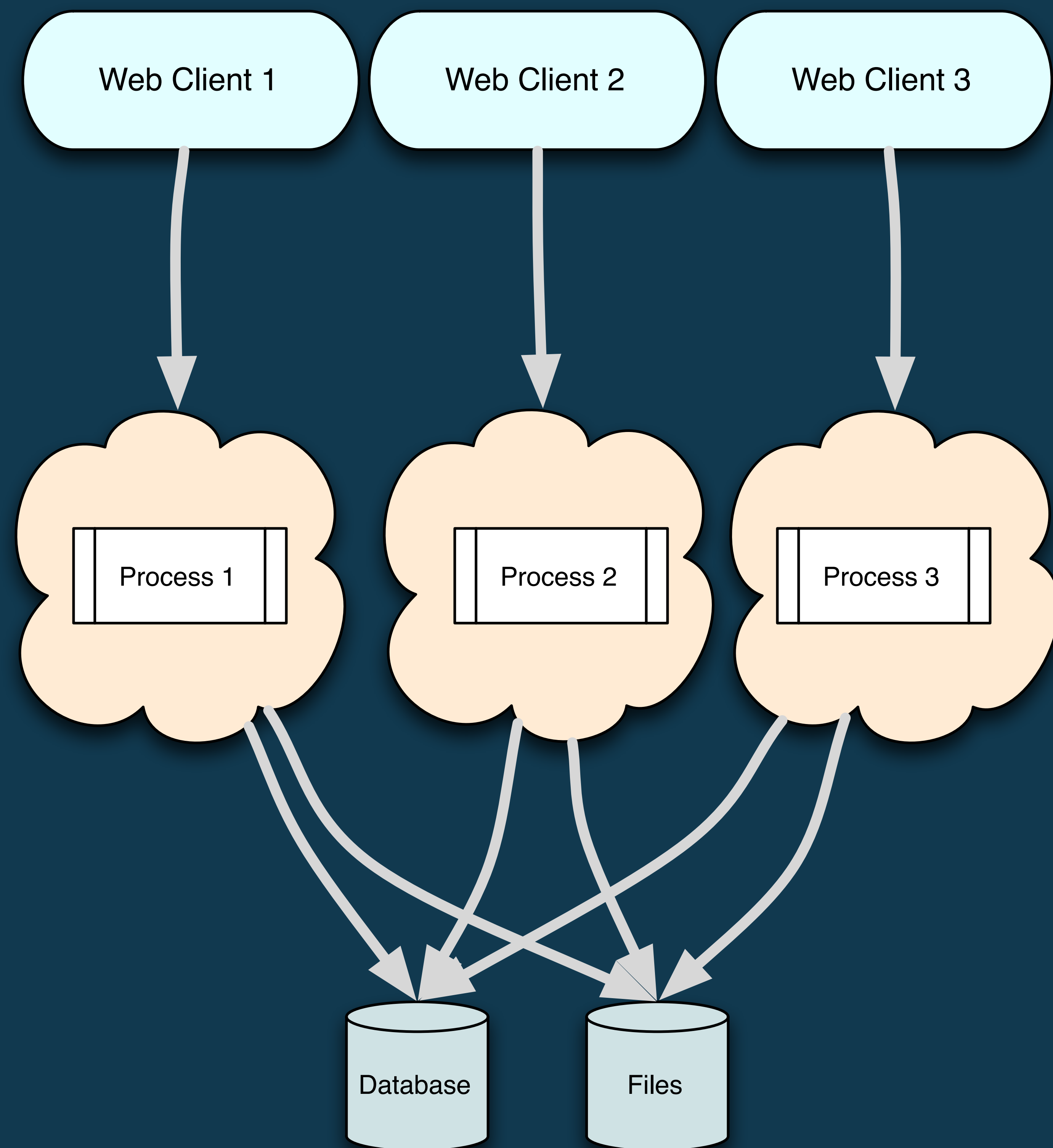
Tuesday, November 4, 14

NoSQL databases fit this model well, with focused abstractions for their data model.

- FP naturally leads to microservices.



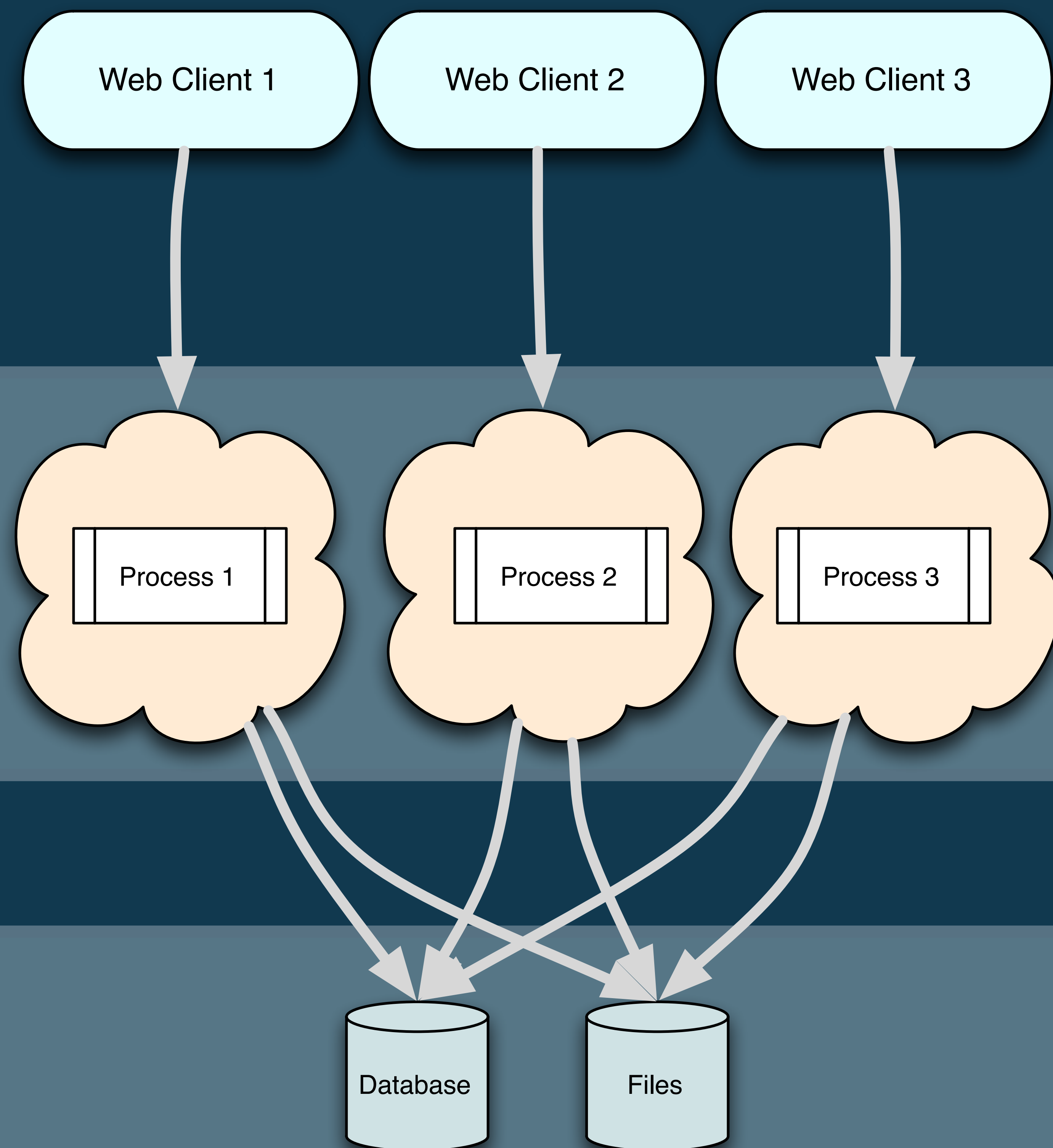
- Data Size ⬆
- Formal Schema ⬇
- Data-Driven Programs ⬆



Tuesday, November 4, 14

And this structure better fits the trends I outlined at the beginning of the talk.

- MapReduce
- HDFS or DBs



Tuesday, November 4, 14

And MapReduce + a distributed file system, like Hadoop's MapReduce and HDFS, fit this model.

TL;DR

- HDFS
  - General-purpose file storage
  - Wide variety of formats possible.
  - Table scans and writes.
  - Unsuitable for record-level CRUD.
    - batch mode
- NoSQL
  - Specific-purpose storage
  - Preferred data model (usually)
  - Designed for record-level CRUD.
    - "Events".

And fads...

Would a Hipster choose a SQL Database??

Signs you've gone too far:

- "I moved all that transaction logic into the application."
- "I'm using a key-value store and parsing the value blobs into objects."
- "I like writing queries in JSON!"

The Tipping Point...

A Facebook Story

Common Use for Hadoop: Data Warehouse Replacement

Tuesday, November 4, 14

Data warehouse systems hit scalability limits and cost/TB concerns at large scale...

“Our data warehouse can only store 6-months of data unless we upgrade for a $1M.”

Data warehouse systems hit scalability limits and cost/TB concerns at large scale...

Traditional Data Warehouse

- Pros
  - Mature
  - Rich SQL, analytics functions
  - Scales to “mid-size” data
- Cons
  - Expensive per TB
  - Can’t scale to Hadoop-sized data sets

Data warehouses tend to be more scalable and a little less expensive than OLTP systems, which is why they are used to “warehouse” transactional data and perform analytics. However, their $/TB is ~10x the cost on Hadoop and Hadoop scales to larger data sets.

Solution #1: Replace the Data Warehouse with NoSQL?

Tuesday, November 4, 14

Could you use a NoSQL store instead??

But SQL is critical to Data Warehouses

Tuesday, November 4, 14

NoSQL does give you the more cost-effective storage, but SQL is very, VERY, important for most DW applications and the analysts who use DWs, so your "NoSQL" store would need a powerful query tool to support common DW scenarios. However, DW experts usually won't tolerate anything that isn't SQL.

Solution #2: Replace the Data Warehouse with Hadoop?

Tuesday, November 4, 14

Could you use a Hadoop somehow instead??

Data Warehouse vs. Hadoop?

- Data Warehouse
  - +Mature
  - +Rich SQL, analytics
  - –Scalability
  - –$$/TB
- Hadoop
  - –Maturity vs. DWs
  - +Growing SQL
  - +Massive scalability
  - +Excellent $$/TB

Tuesday, November 4, 14

Data warehouses tend to be more scalable and a little less expensive than OLTP systems, which is why they are used to "warehouse" transactional data and perform analytics. However, their $/TB is ~10x the cost on Hadoop and Hadoop scales to larger data sets.

✓ Solution #2: Replace the Data Warehouse with Hadoop?

+ SQL

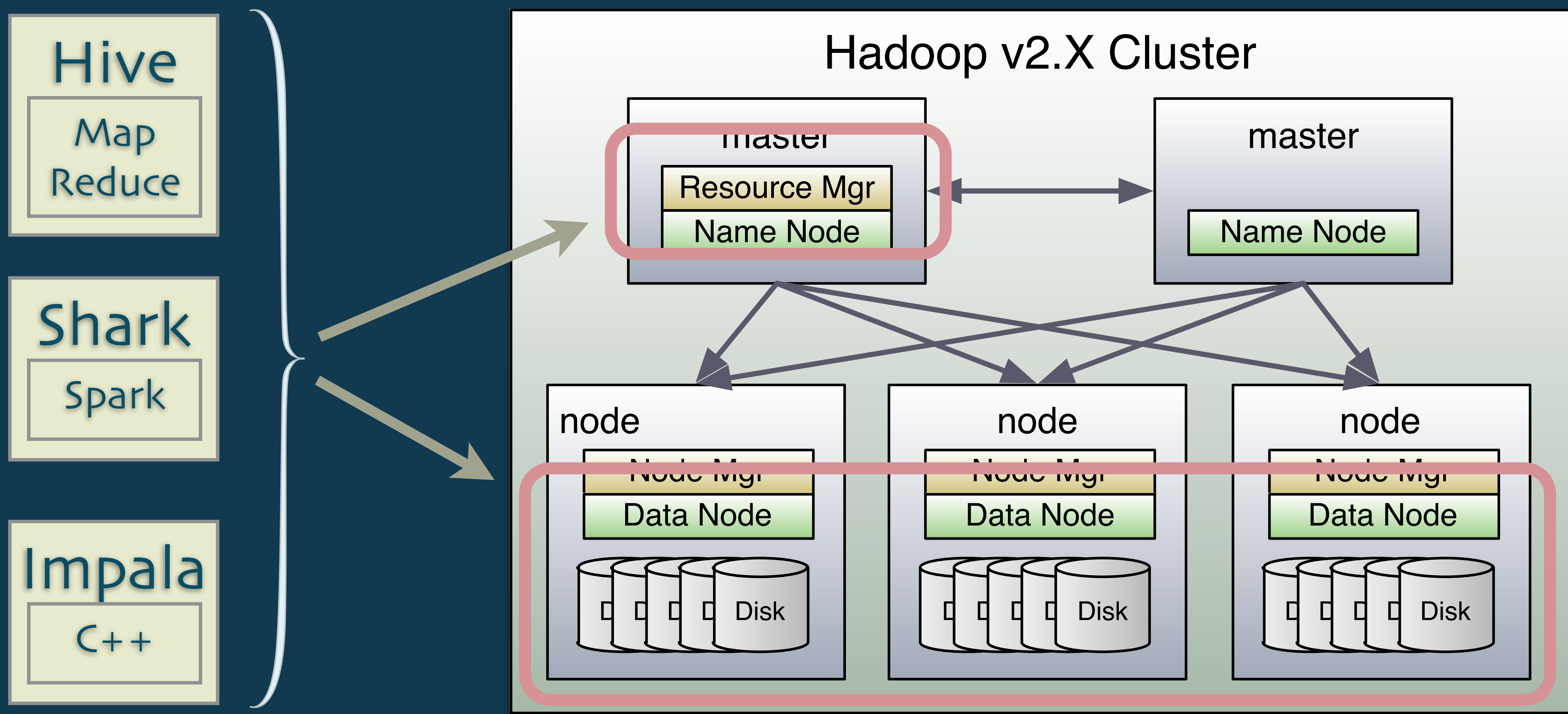
Could you use a Hadoop somehow instead??

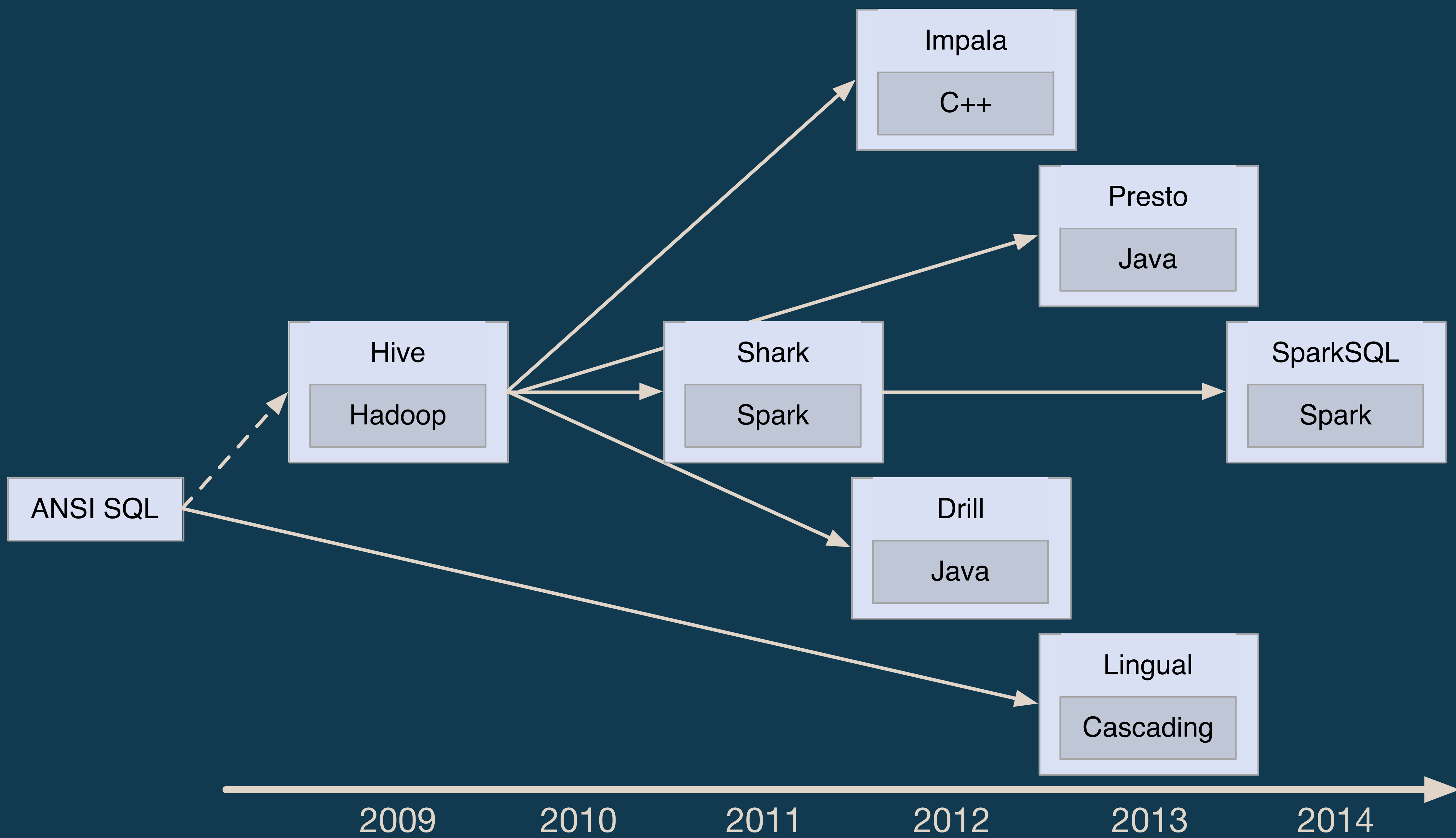
SQL on Hadoop

Facebook had data in Hadoop. Facebook's Data Analysts needed access to it...

Facebook was an early adopter of Hadoop and they have remained one of the biggest users in terms of data volumes, 100s of PB at this point!

… so they created Hive.


Hive
Map Reduce
Shark
Spark
Impala
C++
Hadoop v2.X Cluster
master
Resource Mgr
Name Node
master
Name Node
node
Node Mgr
Data Node
Disk
node
Node Mgr
Data Node
Disk
node
Node Mgr
Data Node
Disk



Tuesday, November 4, 14

SQL has been a rich area of innovation and development in Hadoop.
Some of the tools in the Hive family are mixing in more ANSI SQL constructs.

Claim:
Hadoop wouldn’t be popular if Hive didn’t exist.

Hive

- SQL dialect, HiveQL
  - Not very ANSI-compliant.
  - Evolving to fix this.
- Uses MapReduce back end.
  - So annoying latency.
  - But latest version moves to Tez.
- First SQL on Hadoop.
- Developed by Facebook.

Shark

- HiveQL front end.
- Spark back end.
  - Significantly better performance; +30x better than MapReduce.
- Developed by Berkeley AMP as part of the Spark project.
- Now deprecated...

SparkSQL

- Still HiveQL front end.
  - Or will be once they catch up.
- New query engine, Catalyst.
  - Better modularity, code quality than Hive.
- Separate "client-server" integration with Hive.
- Native support for JSON & Parquet.

Impala

- HiveQL front end.
  - With a richer library of functions.
- C++ and Java back end.
- Provides up to 100x better performance than Hive's MapReduce!
- Developed by Cloudera.

See http://www.cloudera.com/content/cloudera/en/products/cloudera-enterprise-core/cloudera-enterprise-RTQ.html.

Lingual

- ANSI SQL front end.
- Cascading back end.
  - Same strengths/weaknesses for runtime performance as Hive.
  - But new Tez backend in Beta.

http://www.cascading.org/lingual/ Cascading is based on MapReduce, but a faster backend based on Tez is in beta. There's also a standalone "local mode" for small jobs and development.
Tez is a next-generation compute engine that fixes many issues with MapReduce.

Word Count in Hive

```
CREATE TABLE docs (line STRING);
LOAD DATA INPATH '/path/to/docs'
INTO TABLE docs;

CREATE TABLE word_counts AS
SELECT word, count(1) AS count FROM
(SELECT explode(split(line, '\W+'))
 AS word FROM docs) w
GROUP BY word
ORDER BY word;
```


Tuesday, November 4, 14


“NewSQL”

NewSQL

- Term coined April 2011 by The 451 Group
  - http://blogs.the451group.com/information_management/2011/04/06/what-we-talk-about-when-we-talk-about-newsql/
- New projects/vendors that:
  - attempt to improve the relational model for distributed systems.
  - attempt to improve vertical scalability.

Typesafe

Tuesday, November 4, 14

It's a broad category and 451 says we shouldn't take it too seriously, just like the term NoSQL.

Examples

- For a comprehensive list:
  - http://www.scalebase.com/the-story-of-newsql/

Spanner and F1

- Google's SQL database.
- Globally-distributed transactions.

VoltDB

- In-memory DB, aims for speed.
- Suitable for streaming apps.
- SQL queries.
- Co-founded by Michael Stonebraker.
- http://voltdb.com/

NuoDB

- Distributed SQL & ACID transactions.
- http://www.nuodb.com/

Clustrix

- Distributed SQL.
- Real-time analytics.
- Often deployed in the cloud, like AWS.
- http://clustrix.com

MariaDB Galera Cluster

- Distributed MariaDB (fork of MySQL).
- Why not improve “legacy” SQL databases?

https://downloads.mariadb.org/mariadb-galera/

webscale.org

- Joint effort of Facebook, Google, LinkedIn, and Twitter engineers to improve MySQL scalability.
- http://webscalesql.org/

Other Notables

- TokuDB - High performance storage engine for use with MongoDB or MySQL.
- GenieDB & ScaleBase - Globally distributed MySQL as a service.
- ScaleDB - Streaming inserts & analytics with SQL.
- TransLattice - Globally distributed, transactional SQL based on Postgres.

NoSQL + SQL

- Cassandra Query Language (CQL).
- Hive + HBase.
- etc.

Typesafe

Tuesday, November 4, 14

Photo: San Francisco Bay

There will always be SQL...

- But just as NoSQL expanded our persistence options, what other approaches might we see for working with data?
  - Not all of the following are persistence tools...

Graphs

- Lots of data is ideally modeled as a graph:
  - Facebook friends.
  - Twitter followers.
  - ...

Pet Peeve

- Graph examples that are just object models.
- Model relationally instead.
- Acid test: do standard graph traversal algorithms make sense? (connected components, minimum spanning tree, cliques, ...)

Tuesday, November 4, 14

Use the right tool for the job. Let's not make graphs a fad...

Graphs

- But performant, distributed graph technology is still a research problem.
- Examples:
  - Titan - Supports Cassandra, HBase storage
  - Pregel - Google's graph engine.
    - Bulk Synchronous Parallel (BSP)
  - GraphX - Built on Spark + distributed FS.
  - Apache Hama, Giraph
  - ...

Tuesday, November 4, 14

I'm not mentioning Neo4j because it's not a distributed graph database.

Logic Programming

- Datalog
  - declarative logic programming.
  - A subset of Prolog.
  - Alternative to SQL's relational model.

Logic Programming

- Examples
  - Datomic - See Stuart Sierra's talk earlier today.
  - Cascalog - Clojure/Cascading/Hadoop.

```
(defn lowercase [w] (.toLowerCase w))

(?<- (stdout) [?word ?count]
 (sentence ?s)
   (split ?s :> ?word1)
 (lowercase ?word1 :> ?word)
   (c/count ?count))
```

Typesafe

Tuesday, November 4, 14

This is the famous "word count" algorithm written in Cascalog

Probabilistic Programming

- Languages for Probabilistic Graphical Models.
- Used to model systems that are inherently probabilistic.

http://en.wikipedia.org/wiki/Bayesian_network
http://en.wikipedia.org/wiki/Markov_chain
PGMs are essential tools for many machine learning and artificial intelligence systems. But they require some expertise to build, both mastery of the PGM concepts and implementing them in conventional programming languages There is growing interest in designing languages that encapsulate this complexity.

Probabilistic Programming

- Bayesian Networks
  - Model probable causes leading to outcomes.
  - Observe outcomes and infer causes.
  - Examples:
    - Medical diagnosis.
    - Fault isolation.
    - Weather forecasting.

http://en.wikipedia.org/wiki/Bayesian_network
http://en.wikipedia.org/wiki/Markov_chain
PGMs are essential tools for many machine learning and artificial intelligence systems. But they require some expertise to build, both mastery of the PGM concepts and implementing them in conventional programming languages There is growing interest in designing languages that encapsulate this complexity.

Probabilistic Programming

- Markov Chains
  - Model sequences of events where the probability of the next event depends on one or more previous events.
    - Monte Carlo Markov Chain - Assume only the current event matters when predicting the next event.
  - Example: Self-driving cars!

http://en.wikipedia.org/wiki/Bayesian_network
http://en.wikipedia.org/wiki/Markov_chain
PGMs are essential tools for many machine learning and artificial intelligence systems. But they require some expertise to build, both mastery of the PGM concepts and implementing them in conventional programming languages There is growing interest in designing languages that encapsulate this complexity.

Probabilistic Programming

- What would "probabilistic queries" look like?
- BlinkDB
  - A massively parallel, approximate query engine for interactive SQL queries over large data sets.
  - Allows users to trade-off query accuracy for response time.

Typesafe

Not exactly a probabilistic model, but along the lines of situations where "absolute answers" are not necessary.

http://blinkdb.org/
Another Berkeley Univ. project.

Dataflow Programming

- Tools like Cascading/Scalding and Spark encourage a dataflow-style model.
  - Works in batch and streaming modes.
  - Functional Reactive Programming is an implementation of dataflow programming.

CRDTs

- Convergent Replicated Data Types.
  - http://pagesperso-systeme.lip6.fr/Marc.Shapiro/papers/RR-6956.pdf
  - Approach for sharing mutable state at scale.
  - Requires:
    - Operation where concurrent updates commute.
    - All replicas execute all updates in causal order.

Tuesday, November 4, 14