

NoSQL Search RoadShow
San Francisco, June 6, 2013
dean@concurrentthought.com
[@deanwampler](https://twitter.com/deanwampler)
polyglotprogramming.com/talks

From Big Data to Big Information

Copyright © 2011-2013, Dean Wampler, All Rights Reserved

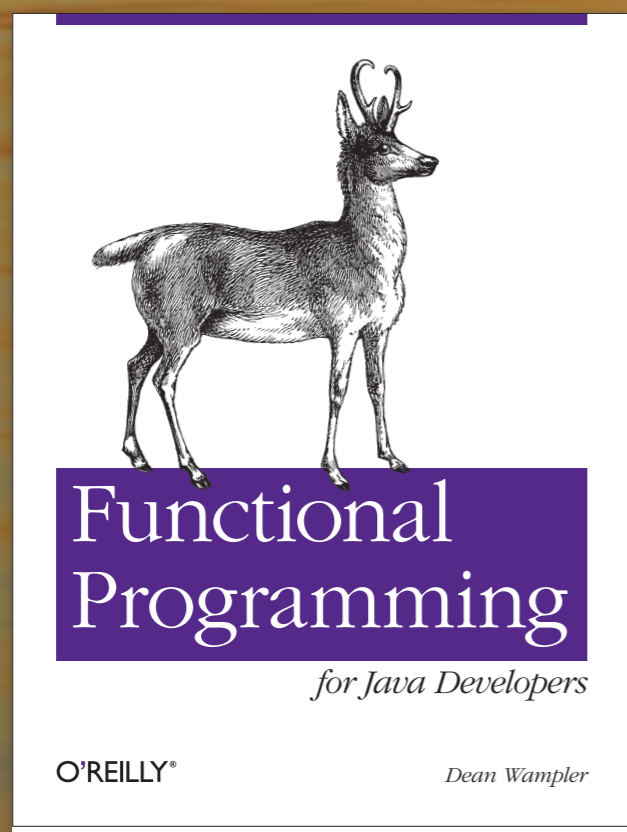


Thursday, June 6, 13

Copyright © Dean Wampler, 2011–2013, All Rights Reserved. Photos can only be used with permission. Otherwise, the content is free to use.

Photo: San Francisco Bay, just south of the airport in Burlingame, before sunrise.

Who am I?



Copyright © 2011-2013, Dean Wampler, All Rights Reserved

Thursday, June 6, 13

My books...

Photo: San Francisco Bay, Burlingame, around sunrise.

Why this talk?

Hadoop hype cycle:

But what gives us

actual *value*?

Thursday, June 6, 13

This talk reflects my experiences working on “Big Data” projects, mostly using Hadoop, with many clients. People sometimes jump on this bandwagon to avoid “being left behind” or to boost their career. Sometimes, it doesn’t make sense for their actual needs. Big Data itself doesn’t do you any good. It’s the information we extract that’s important, so let’s see how different technologies address specific problems.

Photo: Photo: San Francisco Bay, Burlingame, around sunrise.

What Is Big Data?



DevOps Borat @DEVOPS_BORAT

8 Jan

Big Data is any thing which is crash Excel.

Expand



DevOps Borat @DEVOPS_BORAT

6 Feb

Small Data is when is fit in RAM. Big Data is when is crash because is not fit in RAM.

Expand

Big Data

Data so big that traditional solutions are too slow, too small, or too expensive to use.



Hat tip: Bob Korbus

It's a buzz word, but generally associated with the problem of data sets too big to manage with traditional SQL databases. A parallel development has been the NoSQL movement that is good at handling semistructured data, scaling, etc.

3 Trends



Copyright © 2011-2013, Dean Wampler, All Rights Reserved

Thursday, June 6, 13

Three trends to organizer our thinking...

Photo: Gull on a pier near Fort Mason, SF

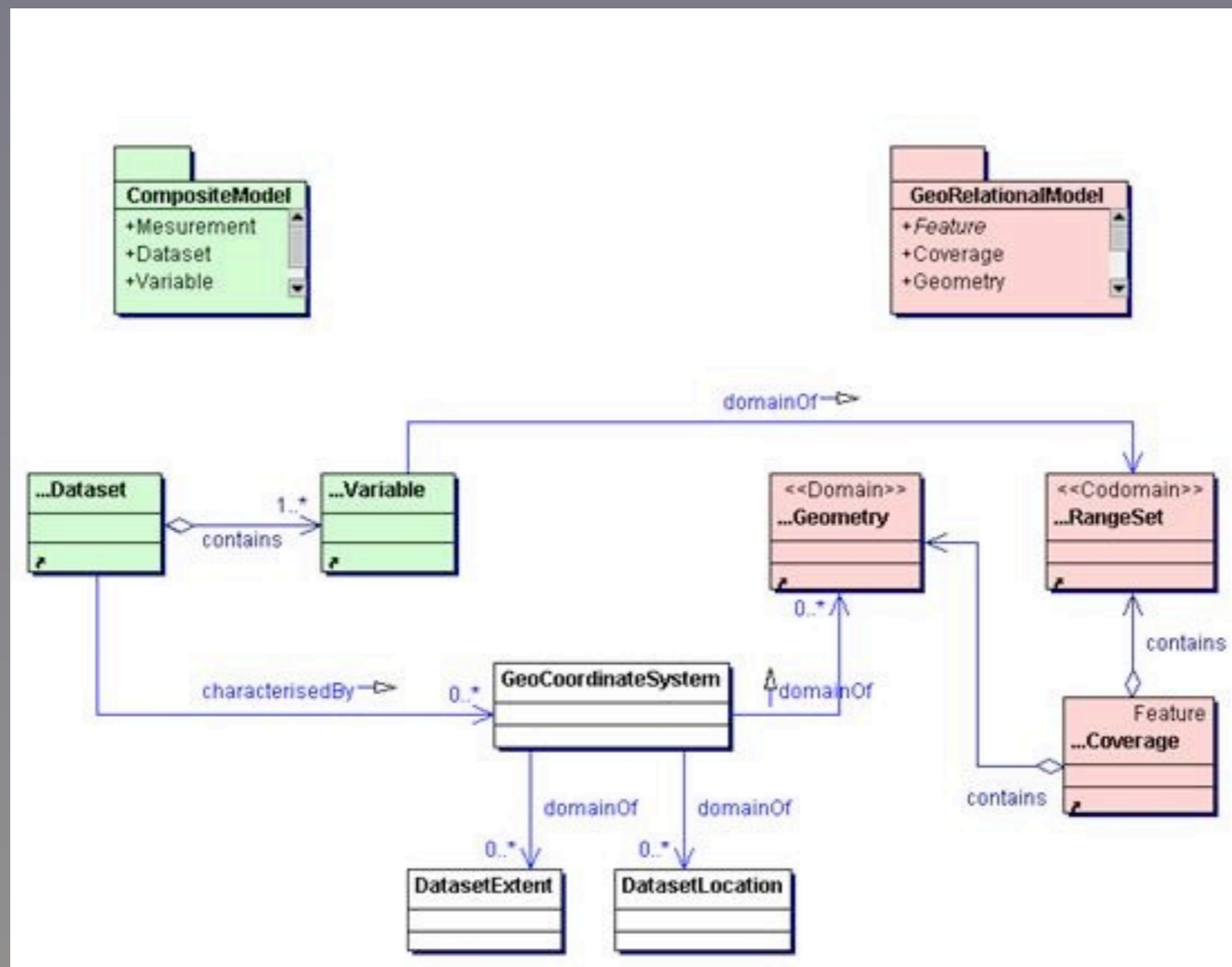
Data Size ↑



Thursday, June 6, 13

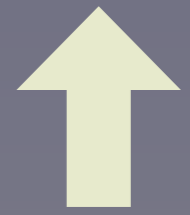
Data volumes are obviously growing... rapidly.
Facebook now has over 600PB (Petabytes) of data in Hadoop clusters!

Formal Schemas ↓



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



Thursday, June 6, 13

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...)

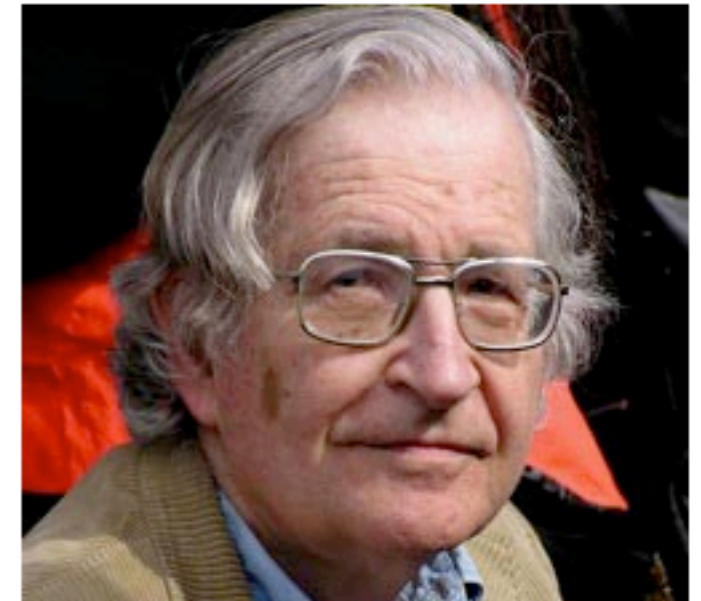
Probabilistic Models vs. Formal Grammars

[tor.com/blogs/...](http://www.tor.com/blogs/...)

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

Thursday, June 6, 13

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>

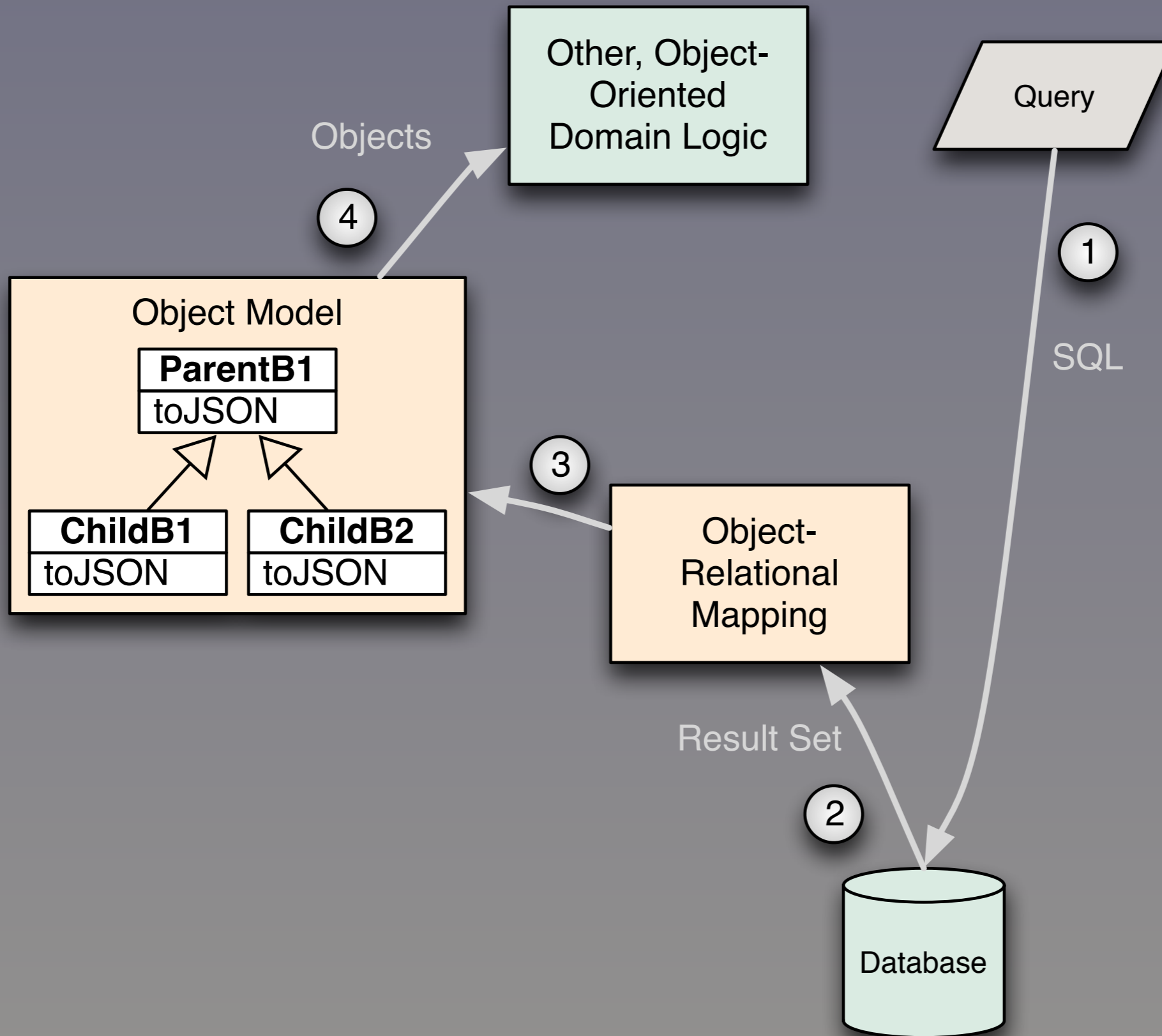
Big Data Architectures



Copyright © 2011-2013, Dean Wampler, All Rights Reserved

Thursday, June 6, 13

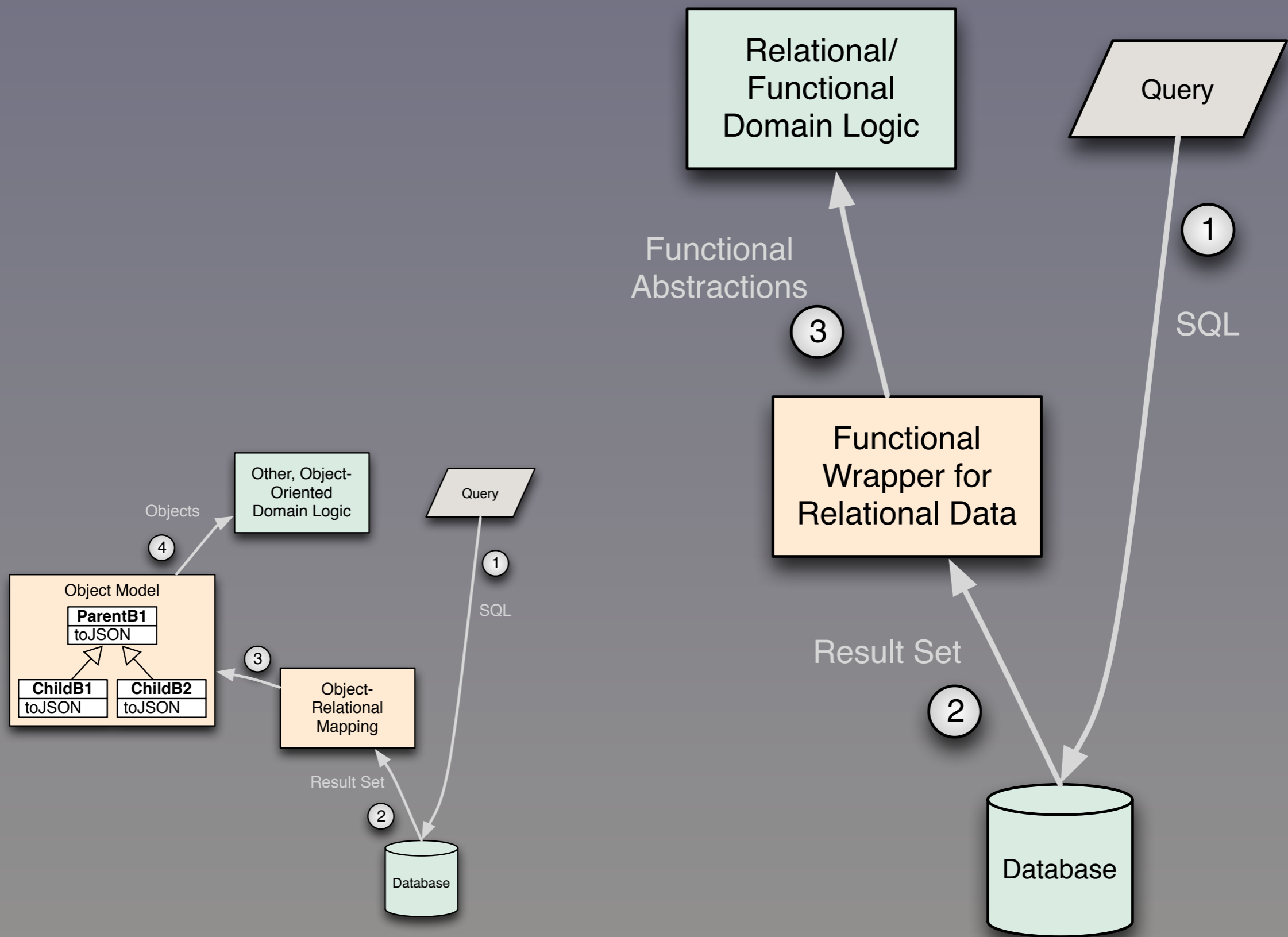
What should software architectures look like for these kinds of systems?
Photo: Light on the Boardwalk and Coit Tower.



Thursday, June 6, 13

Traditionally, we've kept a rich, in-memory domain model requiring an ORM to convert persistent data into the model. This is resource overhead and complexity we can't afford in big data systems. Rather, we should treat the result set as it is, a particular kind of collection, do the minimal transformation required to exploit our collections libraries and classes representing some domain concepts (e.g., Address, StockOption, etc.), then write functional code to implement business logic (or drive emergent behavior with machine learning algorithms...)

The toJSON methods are there because we often convert these object graphs back into fundamental structures, such as the maps and arrays of JSON so we can send them to the browser!



Thursday, June 6, 13

But the traditional systems are a poor fit for this new world: 1) they add too much overhead in computation (the ORM layer, etc.) and memory (to store the objects). 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.

- Focus on:

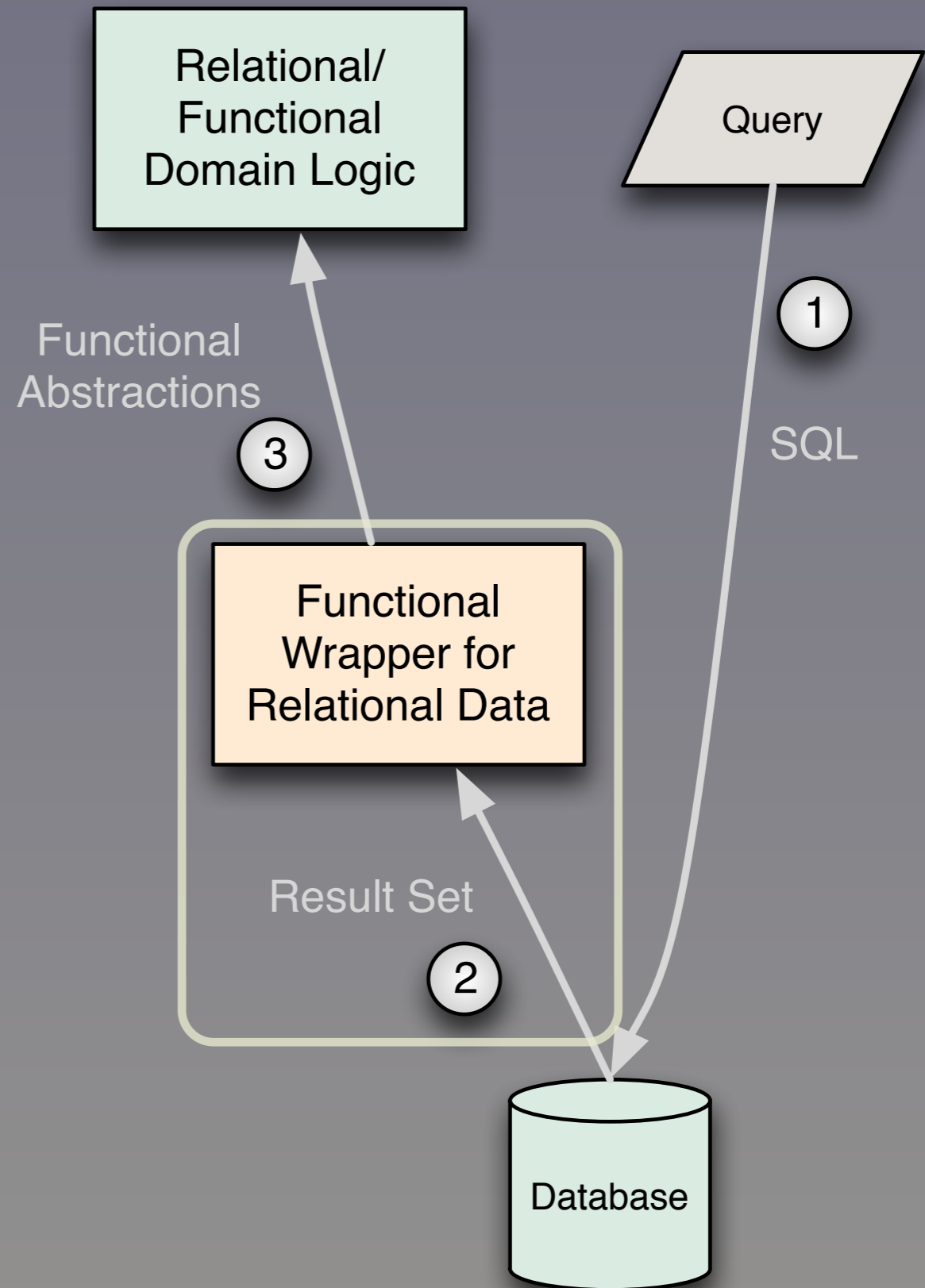
- Lists

- Maps

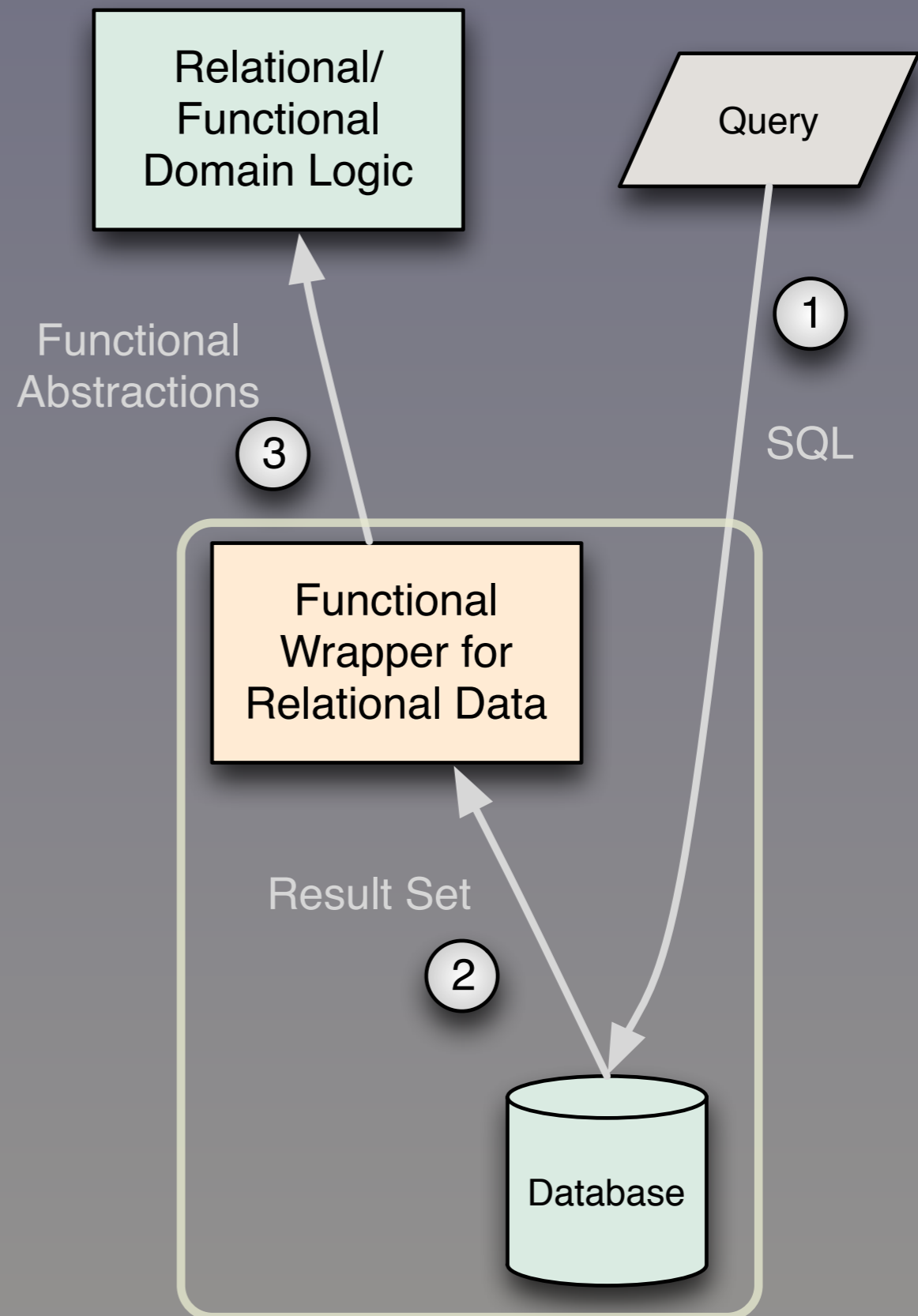
- Sets

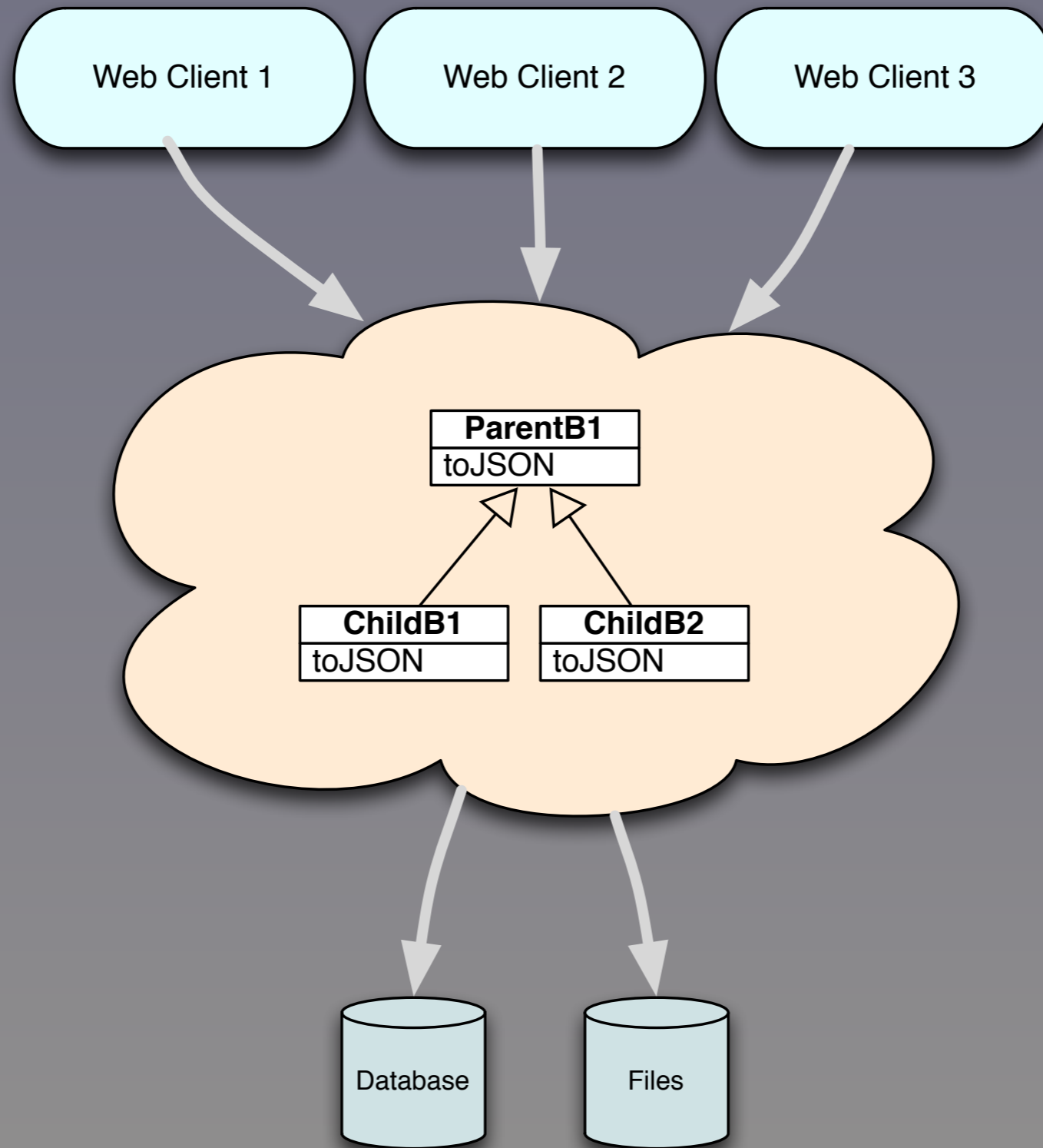
- Trees

- ...



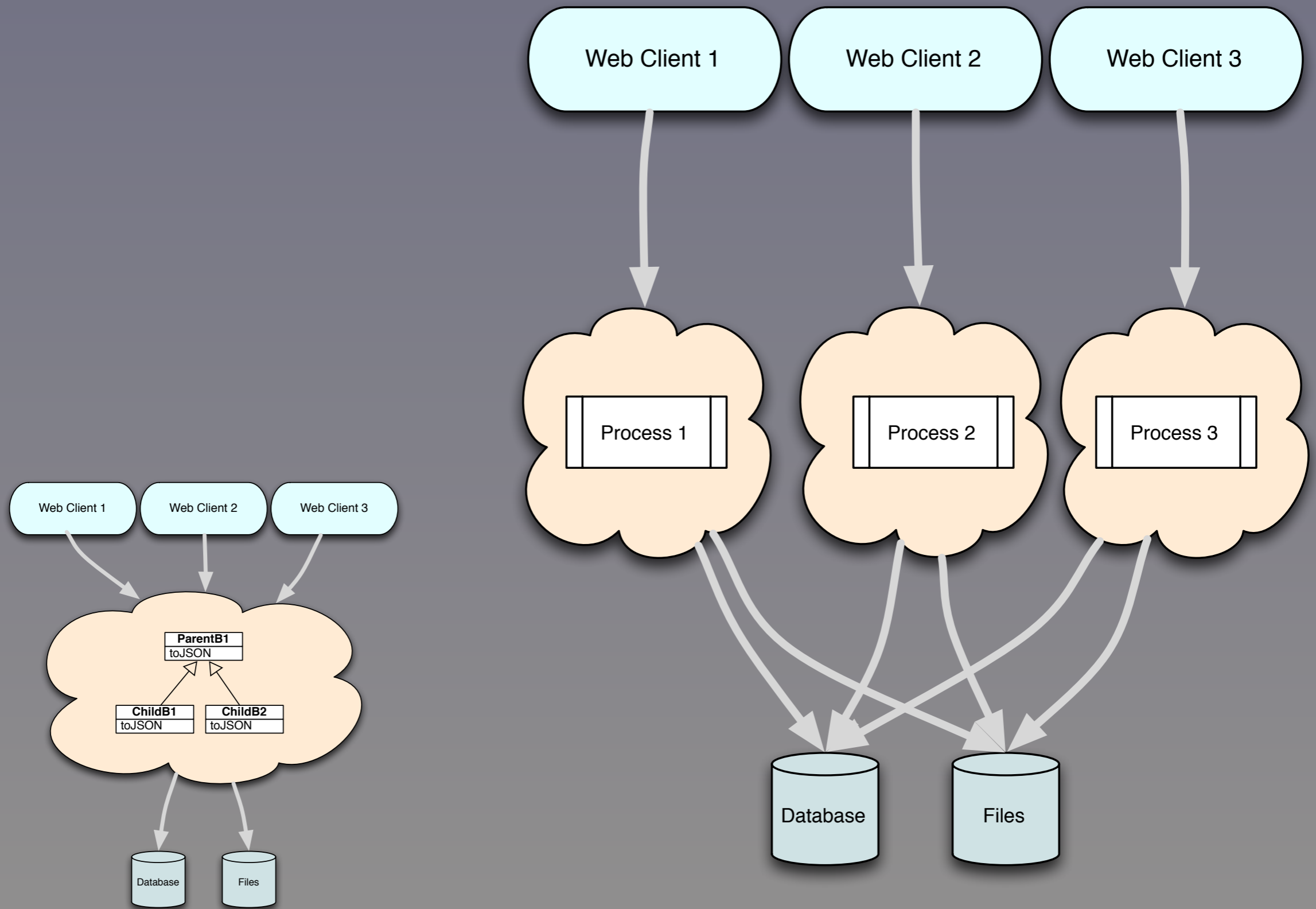
- NoSQL?
- Cassandra, HBase
- Riak, Redis
- MongoDB
- Neo4J
- ...





Thursday, June 6, 13

In a broader view, object models tend to push us towards centralized, complex systems that don't decompose well and stifle reuse and optimal deployment scenarios. FP code makes it easier to write smaller, focused services that we compose and deploy as appropriate.

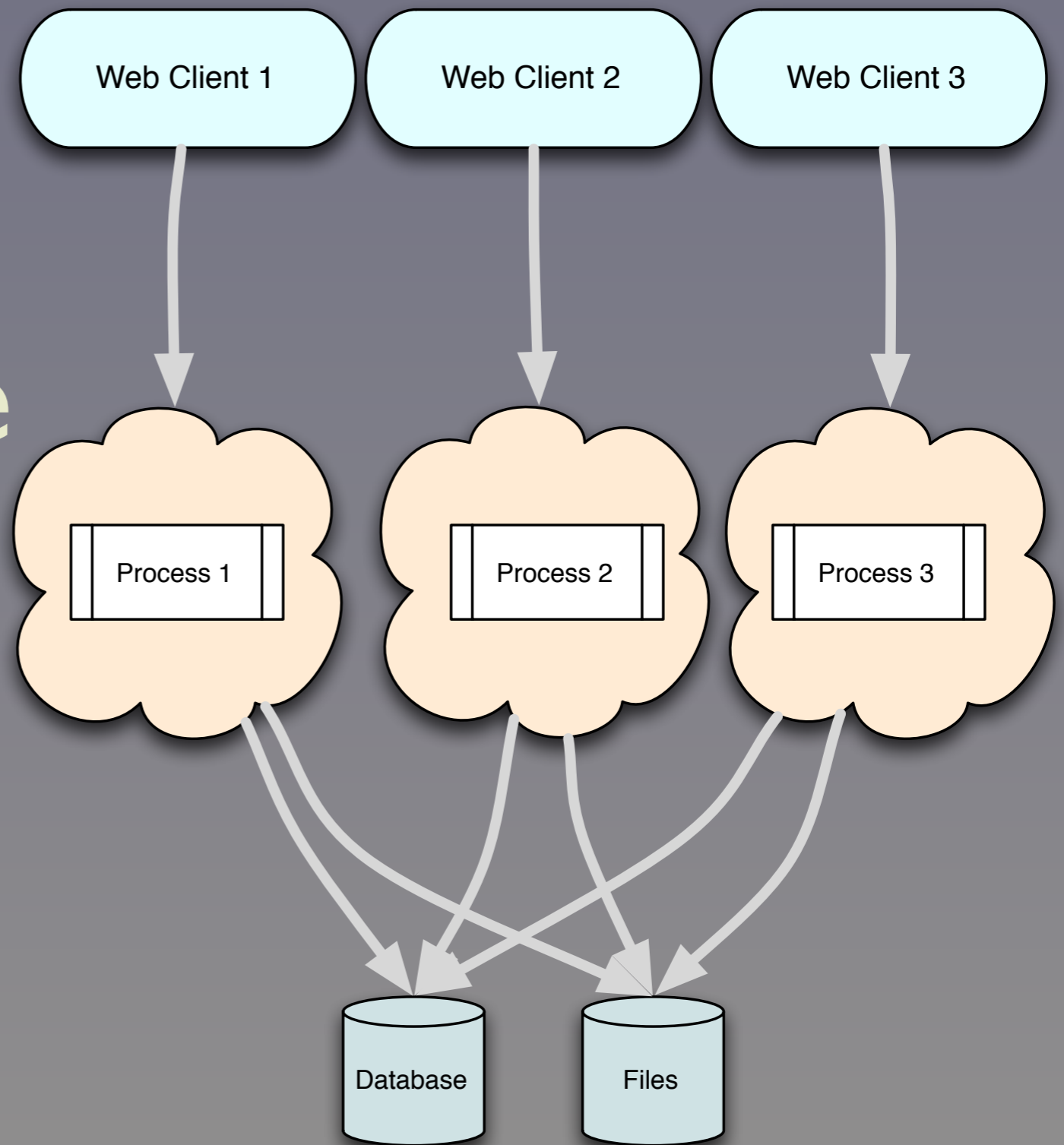


Thursday, June 6, 13

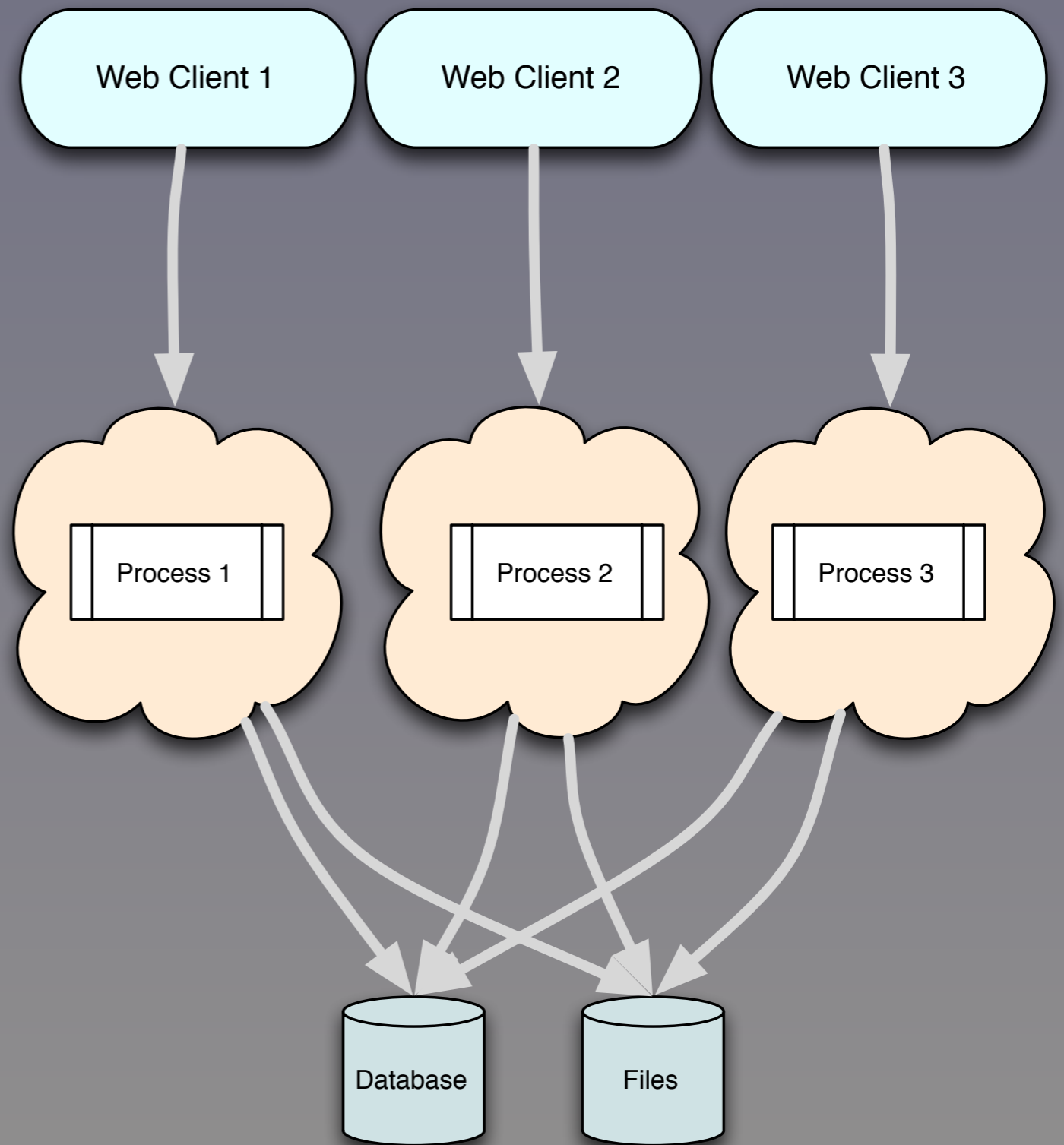
In a broader view, object models tend to push us towards centralized, complex systems that don't decompose well and stifle reuse and optimal deployment scenarios. FP code makes it easier to write smaller, focused services that we compose and deploy as appropriate. Each "ProcessN" could be a parallel copy of another process, for horizontal, "shared-nothing" scalability, or some of these processes could be other services...

Smaller, focused services scale better, especially horizontally. They also don't encapsulate more business logic than is required, and this (informal) architecture is also suitable for scaling ML and related algorithms.

- Smaller, more focused middleware

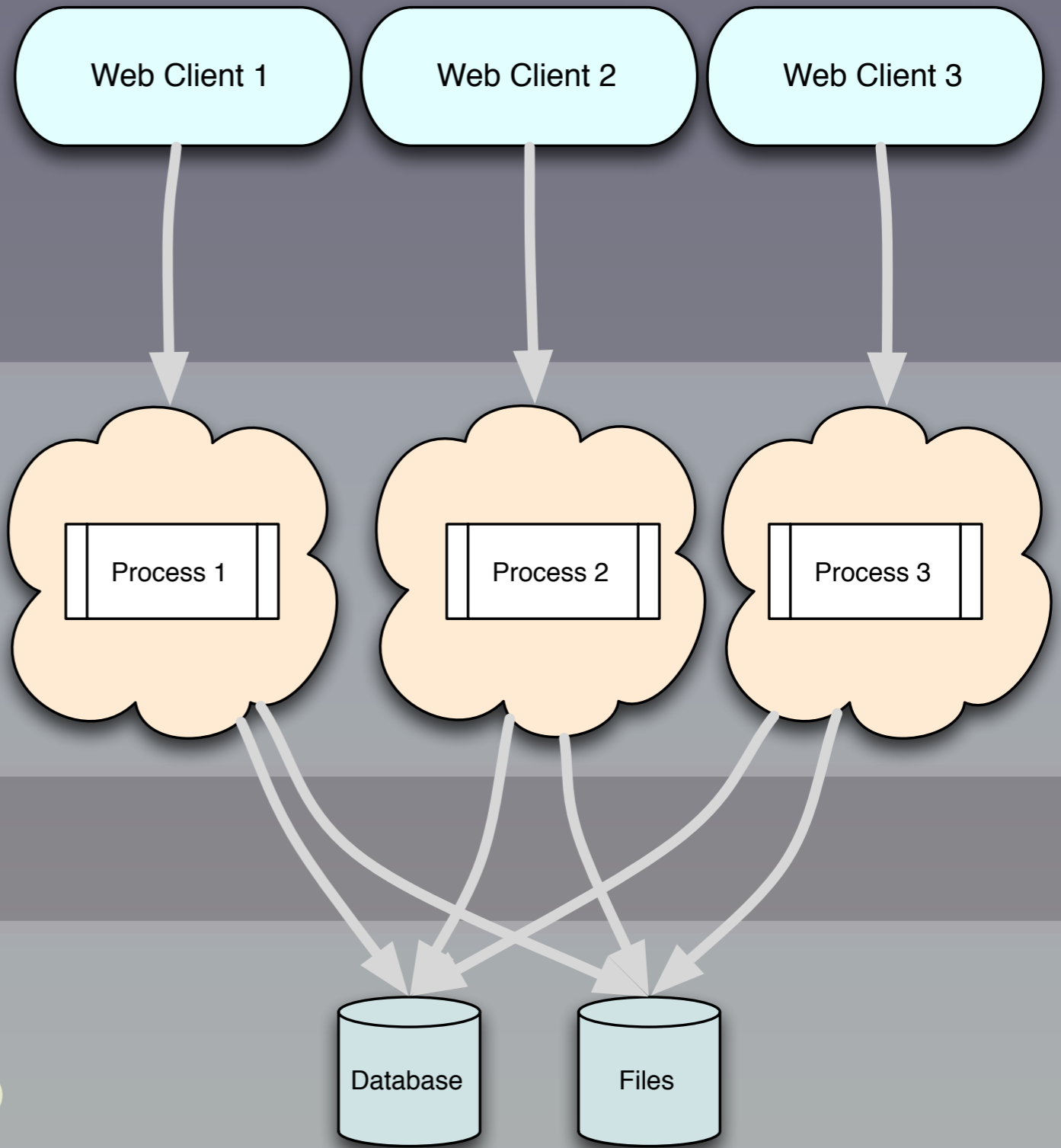


- Data Size ↑
- Formal Schema ↓
- Data-Driven Programs ↑



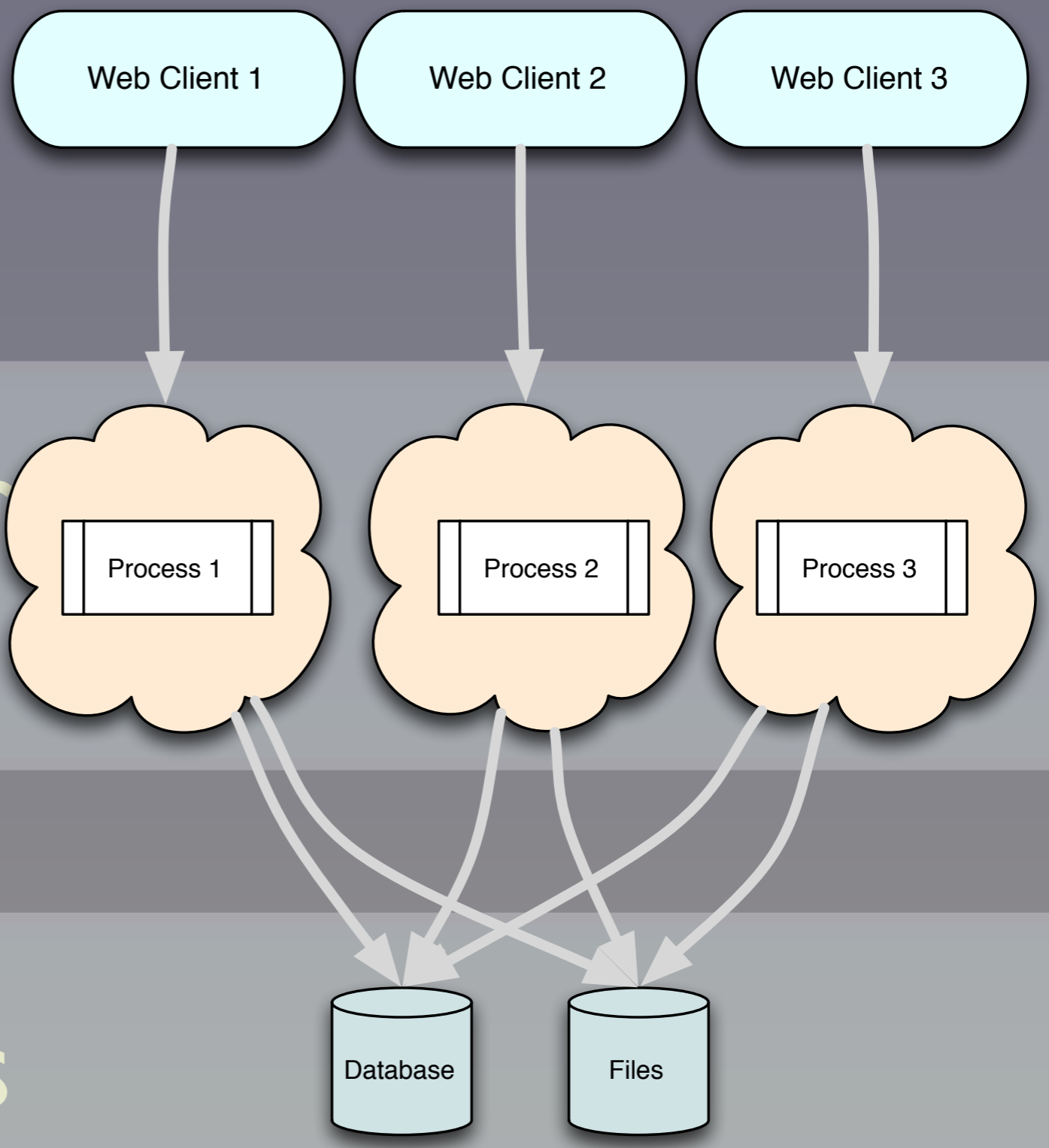
- MapReduce

- Distributed FS



- Hadoop, other middleware

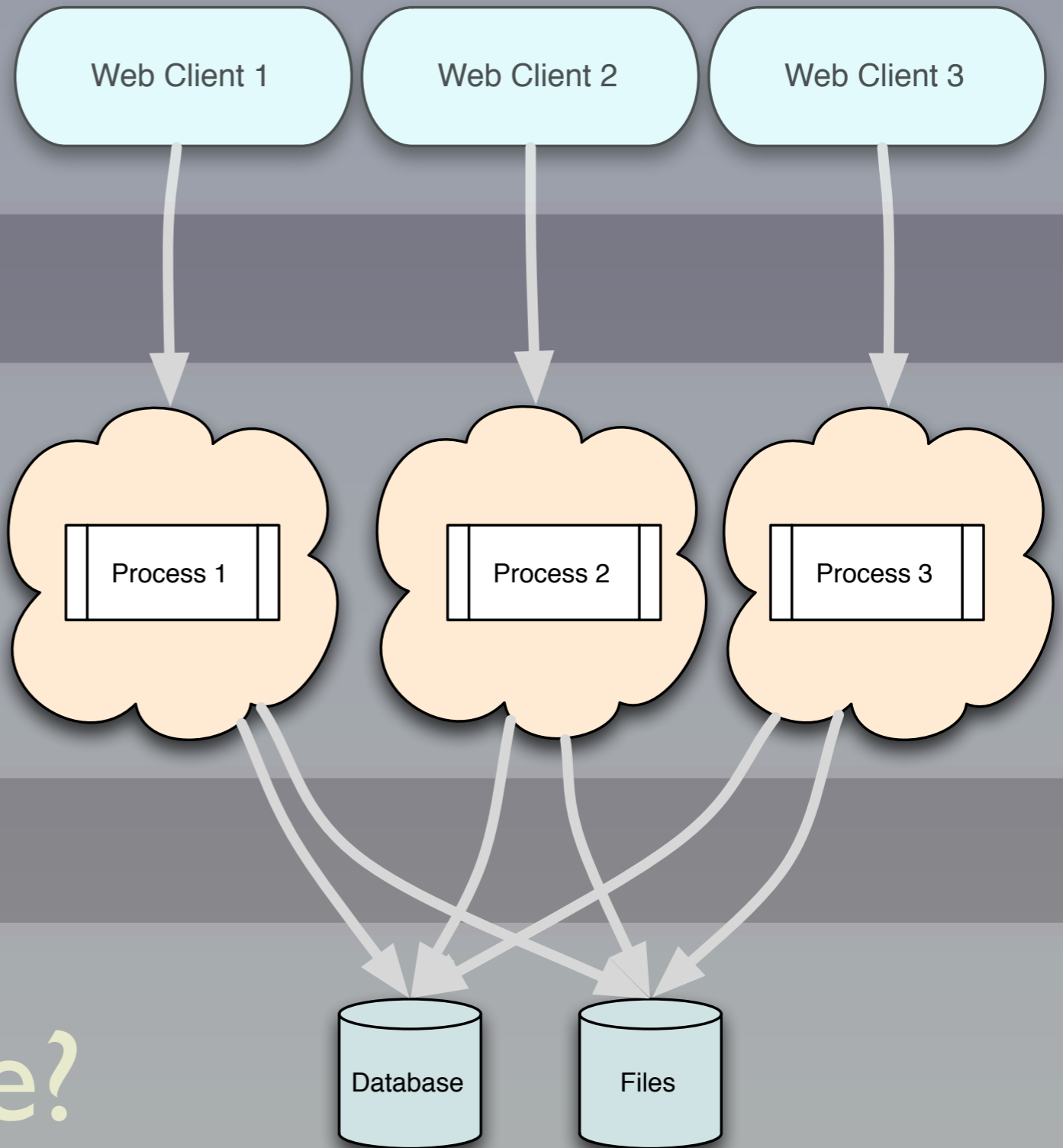
- NoSQL stores




- JSON

- Node.js?

- JSON database?





What Is MapReduce?

Copyright © 2011-2013, Dean Wampler, All Rights Reserved

Thursday, June 6, 13

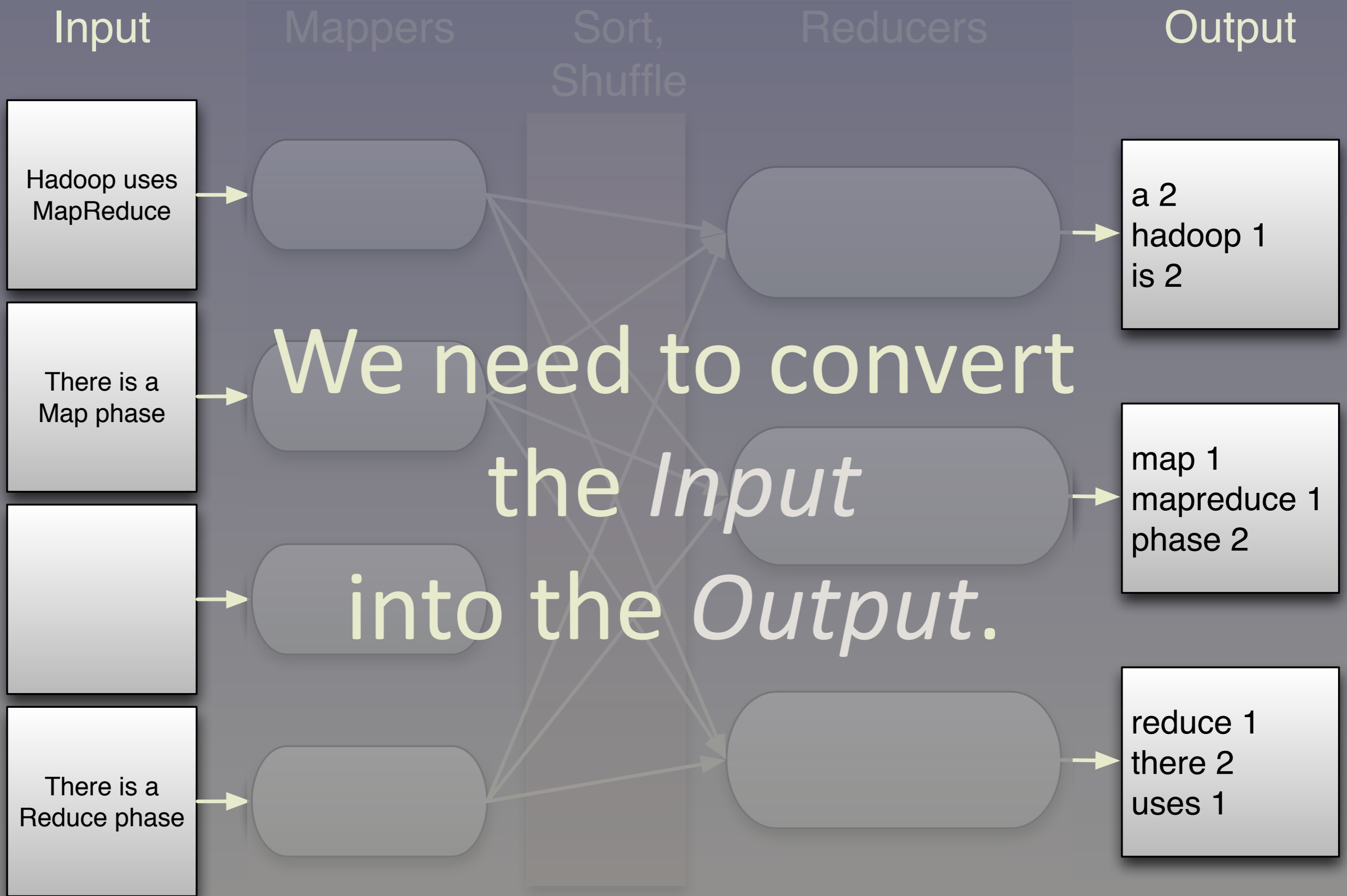
Let's be clear about what MR actually is.

Photo: Near the Maritime Museum

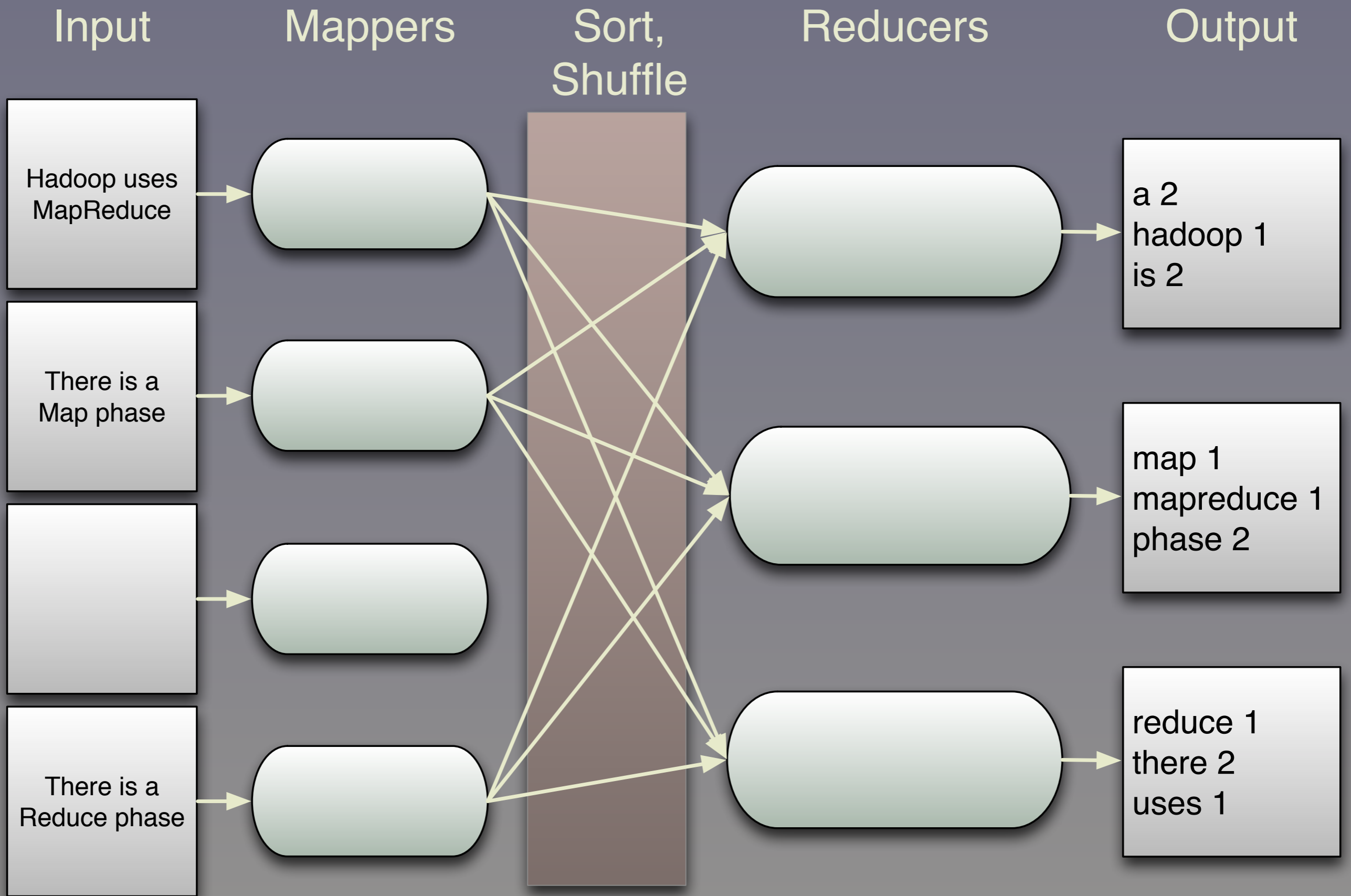
MapReduce in Hadoop

Let's look at a
MapReduce algorithm:
WordCount.

(The *Hello World* of big data...)



Four input documents, one left empty, the others with small phrases (for simplicity...). The word count output is on the right (we'll see why there are three output "documents"). We need to get from the input on the left-hand side to the output on the right-hand side.

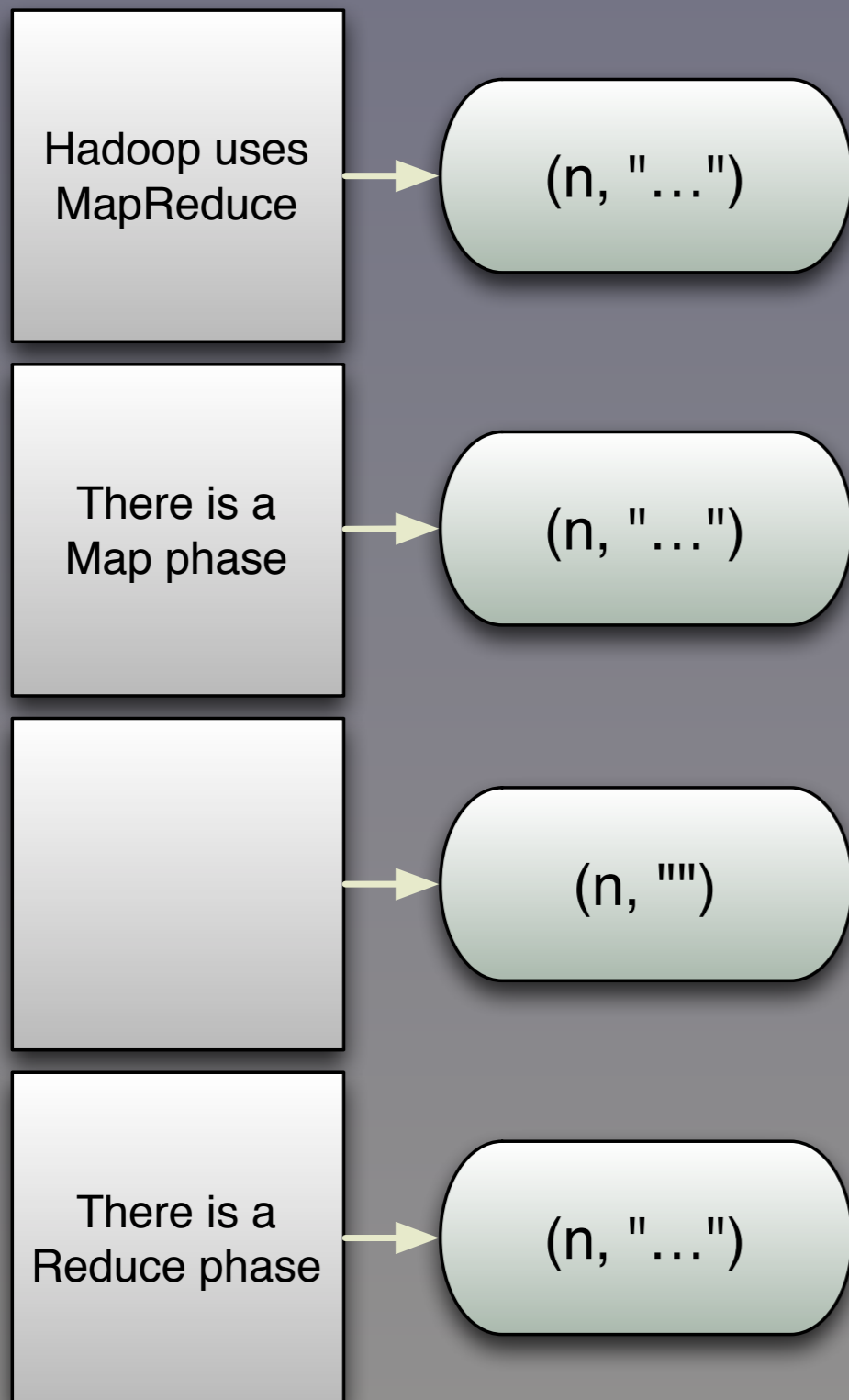


Thursday, June 6, 13

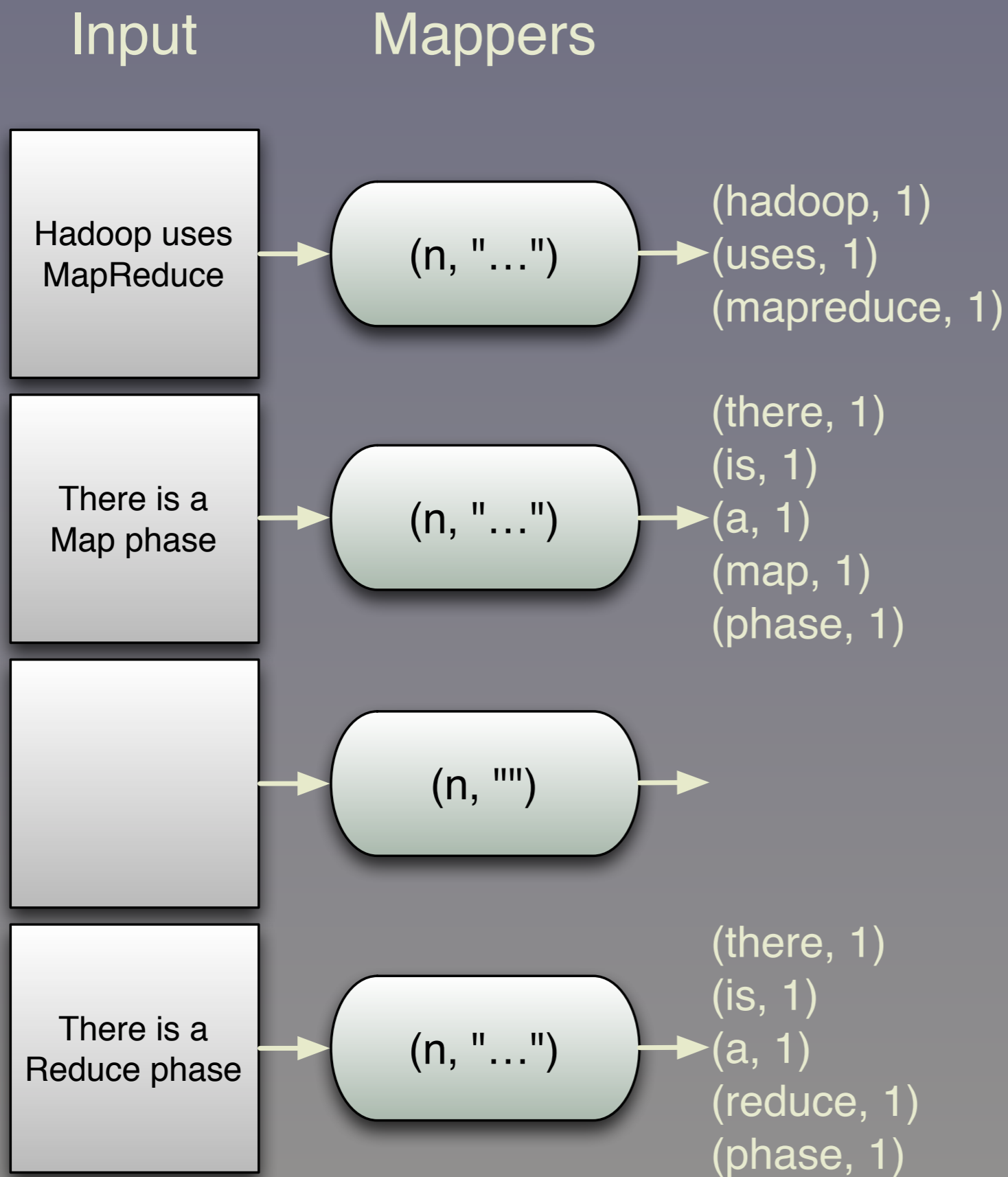
Here is a schematic view of the steps in Hadoop MapReduce. Each Input file is read by a single Mapper process (default: can be many-to-many, as we'll see later). The Mappers emit key-value pairs that will be sorted, then partitioned and "shuffled" to the reducers, where each Reducer will get all instances of a given key (for 1 or more values). Each Reducer generates the final key-value pairs and writes them to one or more files (based on the size of the output).

Input

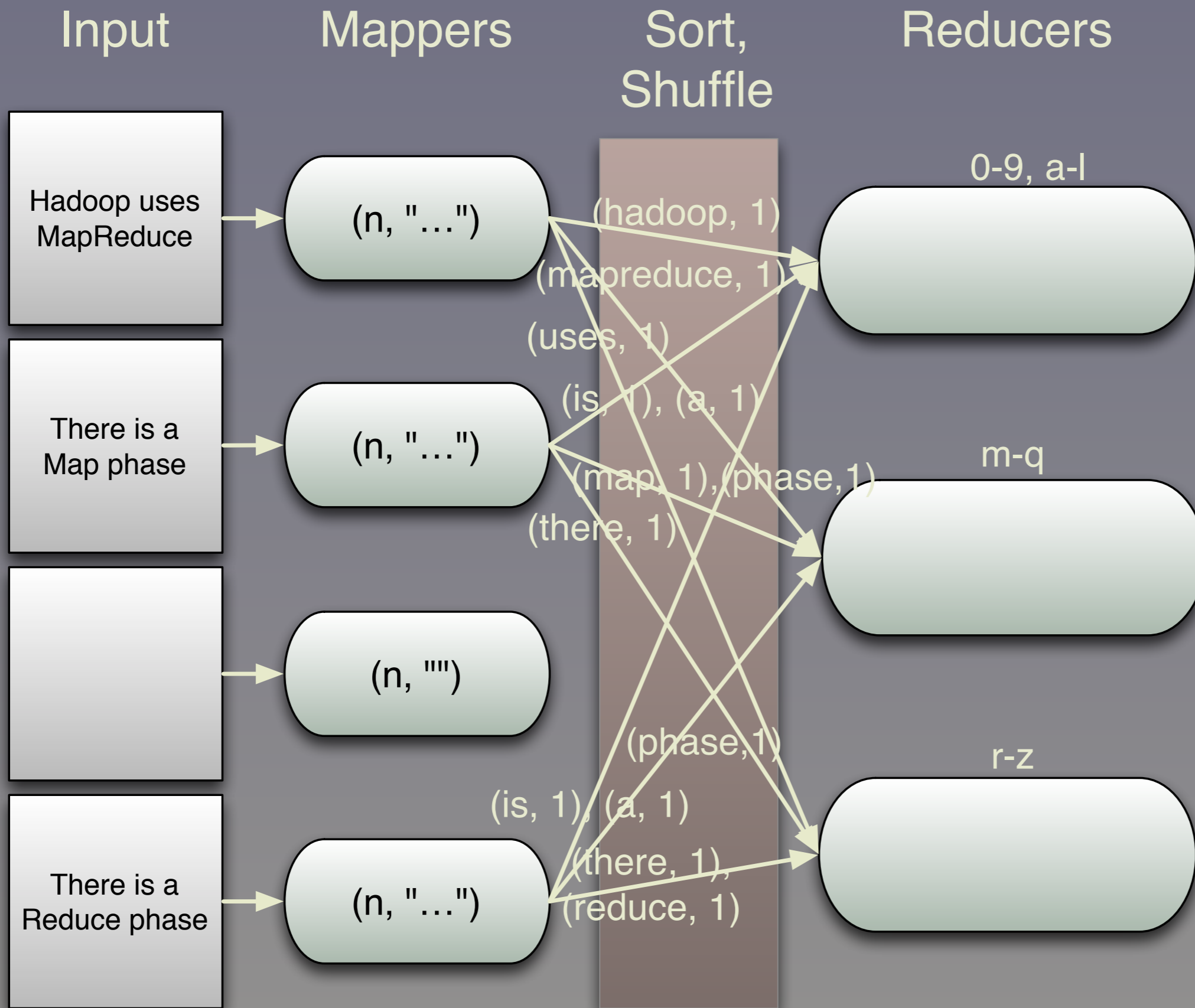
Mappers



Each document gets a mapper. All data is organized into key-value pairs; each line will be a value and the offset position into the file will be the key, which we don't care about. I'm showing each document's contents in a box and 1 mapper task (JVM process) per document. Large documents might get split to several mapper tasks. The mappers tokenize each line, one at a time, converting all words to lower case and counting them...

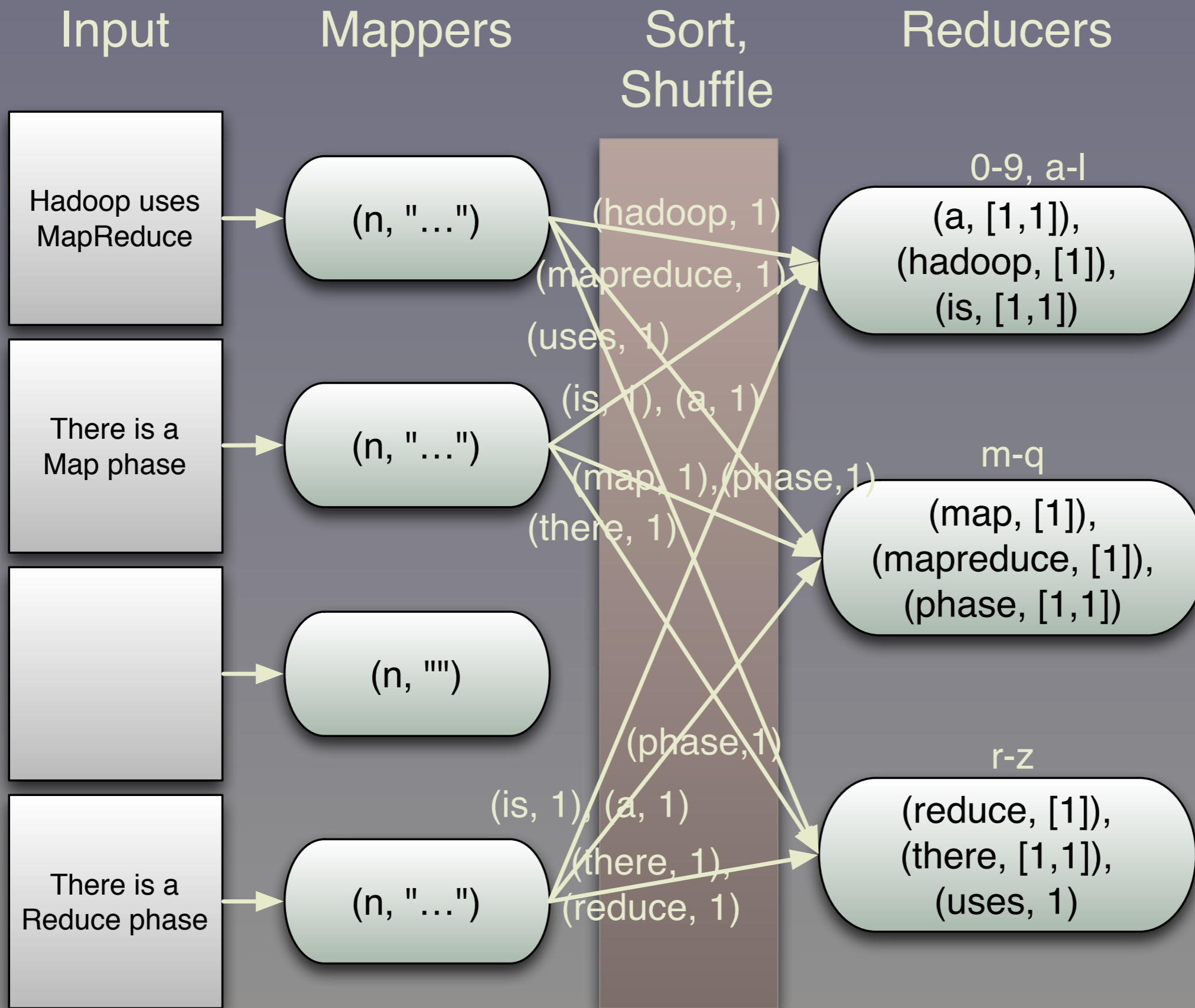


The mappers emit key-value pairs, where each key is one of the words, and the value is the count. In the most naive (but also most memory efficient) implementation, each mapper simply emits (word, 1) each time “word” is seen. However, this is IO inefficient! Note that the mapper for the empty doc. emits no pairs, as you would expect.

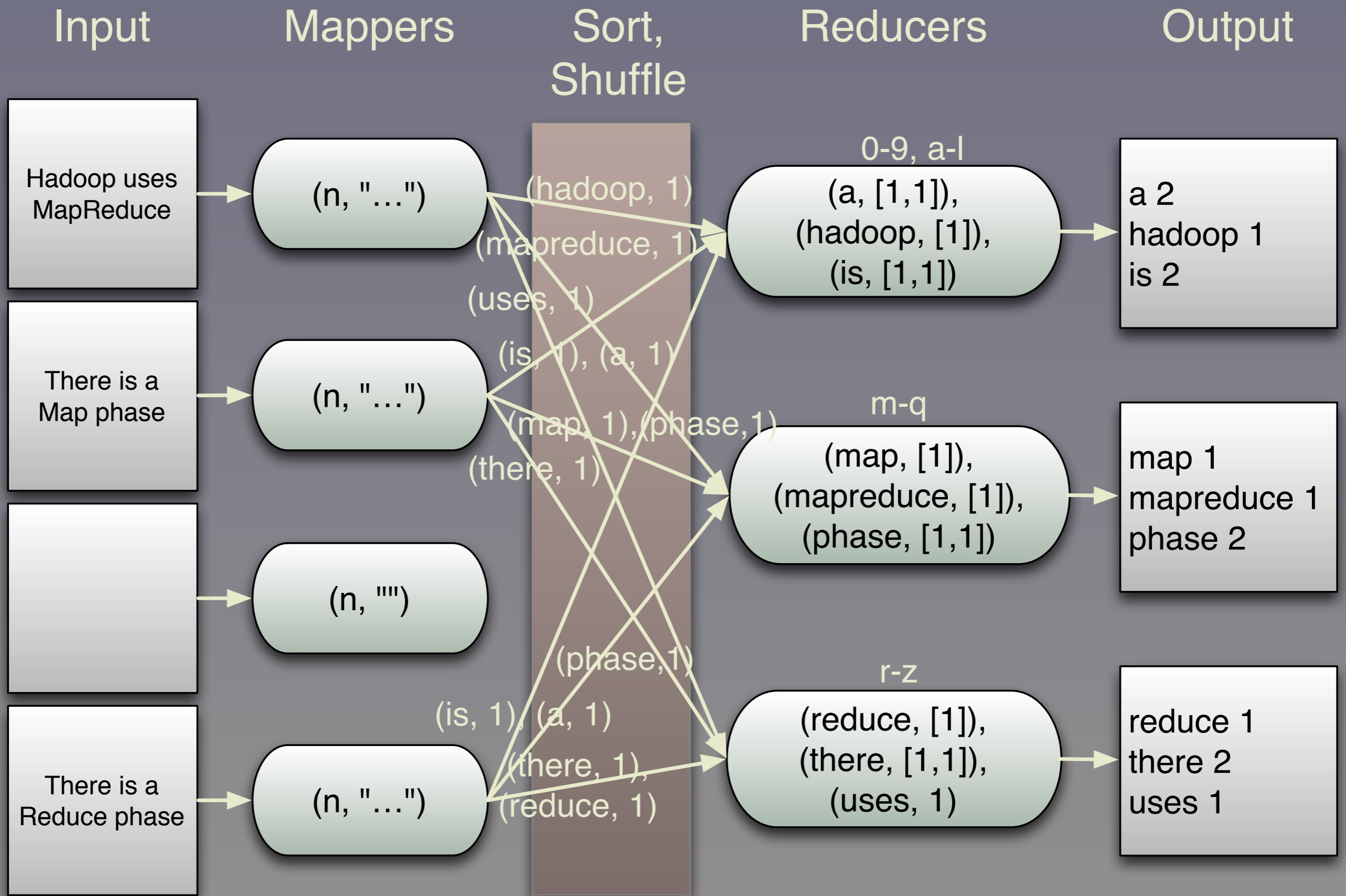


The mappers themselves don't decide to which reducer each pair should be sent. Rather, the job setup configures what to do and the Hadoop runtime enforces it during the Sort/Shuffle phase, where the key-value pairs in each mapper are sorted by key (that is locally, not globally) and then the pairs are routed to the correct reducer, on the current machine or other machines.

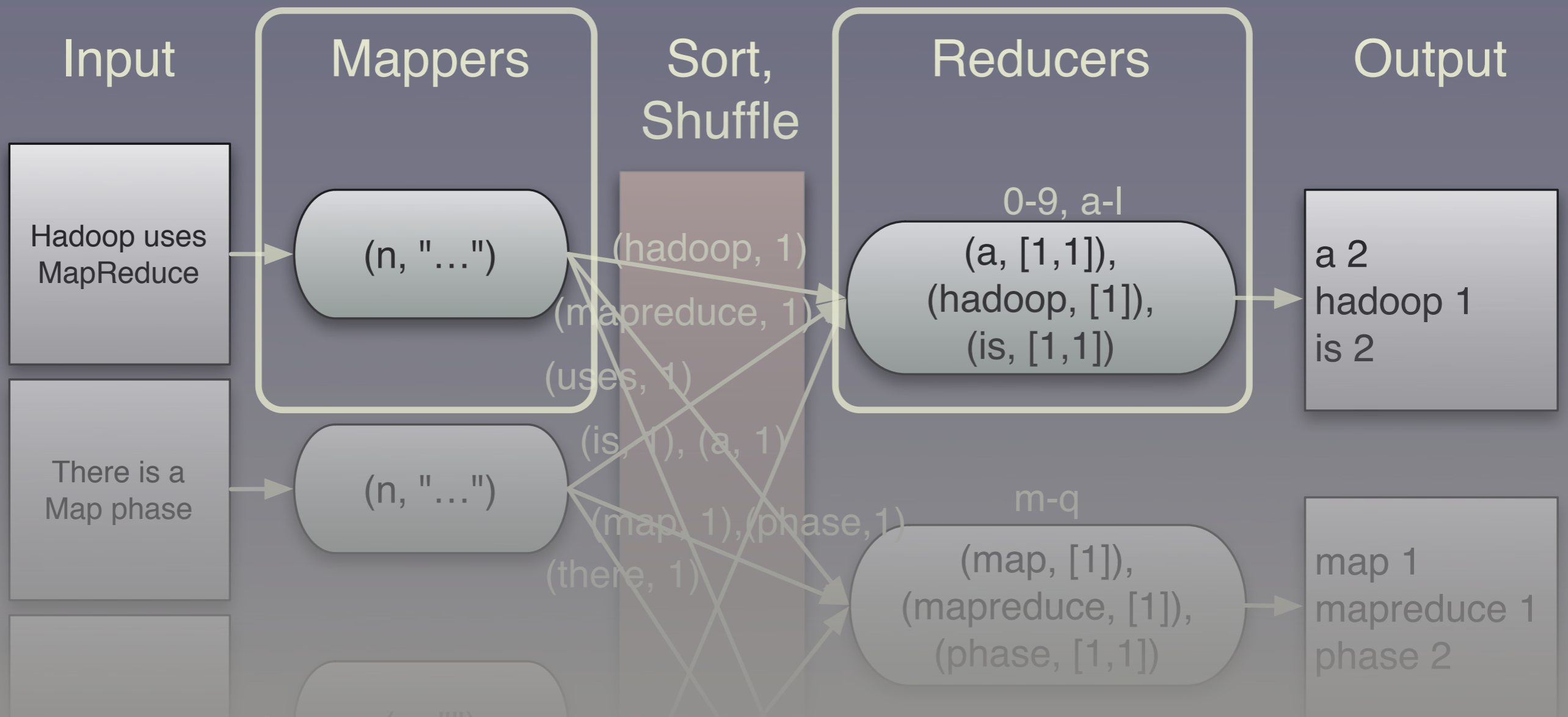
Note how we partitioned the reducers, by first letter of the keys. (By default, MR just hashes the keys and distributes them modulo # of reducers.)



The reducers are passed each key (word) and a collection of all the values for that key (the individual counts emitted by the mapper tasks). The MR framework creates these collections for us.



The final view of the WordCount process flow. The reducer just sums the counts and writes the output. The output files contain one line for each key (the word) and value (the count), assuming we're using text output. The choice of delimiter between key and value is up to you, but tab is common.



Map:

- Transform *one* input to *0-N* outputs.

Reduce:

- Collect *multiple* inputs into *one* output.

To recap, a “map” transforms one input to one output, but this is generalized in MapReduce to be one to 0-N. The output key-value pairs are distributed to reducers. The “reduce” collects together multiple inputs with the same key into

Arguably, Hadoop is our
best, generic* tool for
scaling Big Data
horizontally
(at least today).

As I'll argue it's definitely more first-generation than "perfect". By generic, I mean beyond the more focused goals of a database, SQL or NoSQL, in the sense that it provides a very open-ended compute model and primitive storage, on which you can build a database, which is exactly what HBase is! However, HBase doesn't use MapReduce, just HDFS with its own compute engine. The asterisk is because MR is very general-purpose, but actually somewhat difficult, inflexible, and inefficient for many algorithms.

By design, Hadoop is
great for batch mode
data crunching.

Not so great for event-
stream processing.

By design, Hadoop is
great for batch mode
data crunching.

Not so great for
transactions.

MapReduce is very
course-grained.

1-Map and
1-Reduce phase...

For *Hadoop* in
particularly,
the *Java API* is
hard to use.

There are great higher-level APIs that relieve this burden. We can't discuss them today, but see Cascading, Scalding, Cascalog, for examples.



Use Cases

Copyright © 2011-2013, Dean Wampler, All Rights Reserved

Thursday, June 6, 13

Let's look at some use cases to understand the strengths and weaknesses of Hadoop-oriented solutions vs. NoSQL-oriented solutions.

Photo: Transamerica Building

TL;DR

- *Hadoop*
- Very flexible compute model
- “Table” scans
- Batch mode
- *NoSQL*
- Focused on a particular data model
- Transactional
- Event driven

TL;DR

But *Hadoop*
MapReduce is often
used with a *NoSQL*
store.

Just to be clear that there isn't a clear divide between these technologies, of course. For example, it's not uncommon to integrate MR jobs with Cassandra, HBase, MongoDB, and even relational tables.

Search



Copyright © 2011-2013, Dean Wampler, All Rights Reserved

Thursday, June 6, 13

Improving information search and retrieval, usually through some means of indexing. See the two search talks today by Ryan Tabora and Drew Raines.

Lucene with Solr or ElasticSearch

*A specific solution
for search.*

For very large data sets...

NetApp project:

Use *Solr* to store indices
to data in *HDFS* and
HBase.

Low-cost Data Warehouse

Thursday, June 6, 13

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

Copyright © 2011-2013, Dean Wampler, All Rights Reserved

Problem:

Your current *data warehouse* can only store *6-months* of data without a *\$1M upgrade*.

Traditional DW

- *Pros*
- Mature
- Rich SQL, analytics
- Mid-size Data
- *Cons*
- Expensive - \$/TB
- Scalability limits

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?

Replace the
data warehouse
with *NoSQL*?

*SQL is very important
for data warehouse
applications.*

NoSQL does give you the more cost-effective storage, but SQL is very important for most DW applications, 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?

Replace the
data warehouse
with *Hadoop*?

- *Traditional DW*

- + Mature

- + Rich SQL, analytics

- Scalability

- \$\$/TB

- *Hadoop*

- Less mature

- + Improving SQL

- + Scalable!

- + Low \$/TB

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.

Hadoop has become a popular *data warehouse* supplement/replacement.

Many of my projects have offloaded an overburdened or expensive traditional data warehouse to Hadoop. Sometimes a wholesale replacement, but more often a supplemental strategy, at least for a transitional period of some duration.

SQL on Hadoop

Copyright © 2011-2013, Dean Wampler, All Rights Reserved

Thursday, June 6, 13

Let's discuss the SQL options now, as I consider them very important. Otherwise, the majority of Data Analysts and casual SQL users would not find Hadoop very useful.

Use SQL when you can!

- Hive: SQL on top of MapReduce.
- Shark: Hive ported to Spark.
- Impala: HiveQL with new, faster back end.
- Lingual: SQL on Cascading.

Word Count in Hive SQL!

```
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;
```

Works for Hive, Shark, and Impala

Hive

- SQL dialect.
- Uses MapReduce back end.
 - So annoying latency.
- First SQL on Hadoop.
- Developed by Facebook.

Shark

- HiveQL front end.
- Spark back end.
- Provides better performance.
- Developed by Berkeley AMP.

Impala

- HiveQL front end.
- C++ and Java back end.
- Provides up to 100x performance improvement!
- Developed by Cloudera.

Lingual

- ANSI SQL front end.
- Cascading back end.
- Same strengths/weaknesses for runtime performance as Hive.

Event Stream Processing



Copyright © 2011-2013, Dean Wampler, All Rights Reserved

Thursday, June 6, 13

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

Recall, Hadoop is *great*
for *batch mode* data
crunching.

Not so great for *event-*
stream processing.



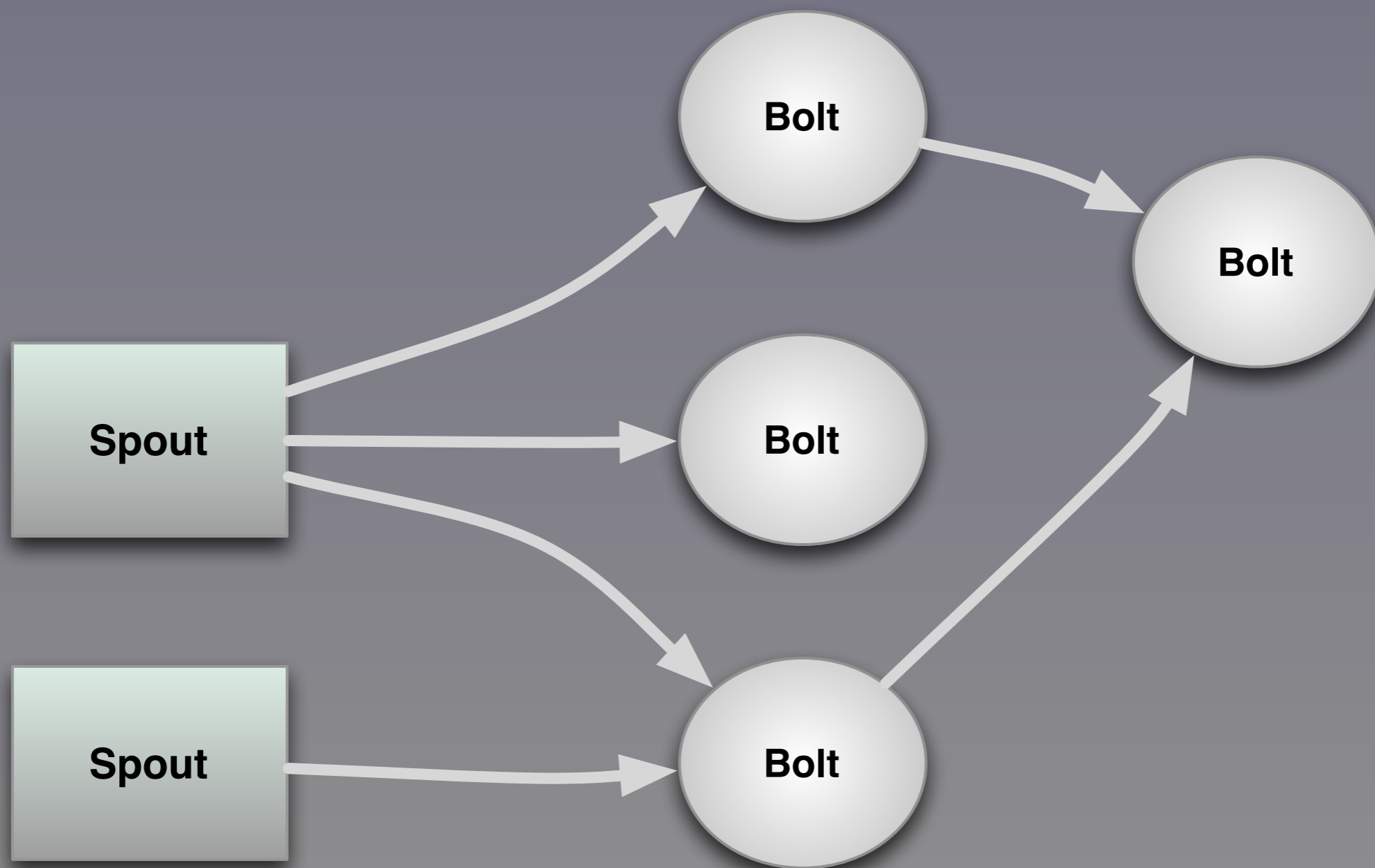
Copyright © 2011-2013, Dean Wampler, All Rights Reserved

Thursday, June 6, 13

Nathan Marz's Storm system for scalable, distributed event stream processing. A complement to Hadoop, as he describes in detail in his book "Big Data" (Manning).

Photo: Top of the AON Building in Chicago after a Storm passed through.

Storm implements
reliable, distributed
event processing.



In Storm terminology, Spouts are data sources and bolts are the event processors. There are facilities to support reliable message handling, various sources encapsulated in Sprouts and various targets of output. Distributed processing is baked in from the start.

- *Hadoop*

- + Cheap

- + Scalable

- + Commercial Support

- Batch mode

- *Storm*

- Less mature

- + Robust

- Commercial Support

- + “Real time”

If you need to respond to event streams, Hadoop simply isn't suitable. Storm is one possible solution. It was designed for this problem, but it is less mature and commercial support is just starting to appear.

Databases to the Rescue?



Copyright © 2011-2013, Dean Wampler, All Rights Reserved

Thursday, June 6, 13

Photo: Actually, this is in Chicago, looking out my condo window.

SQL or NoSQL Databases?

Databases are designed for fast, transactional updates.

So, consider a database for event processing.

Machine Learning



Copyright © 2011-2013, Dean Wampler, All Rights Reserved

Thursday, June 6, 13

ML includes recommendation engines (e.g., the way Netflix recommends movies to you or Amazon recommends products), classification (e.g., SPAM classifiers, character and image recognition), and clustering. Other specialized examples include text mining and other forms of natural language processing (NLP).

- *Recommendations*: Netflix movies, Amazon products, ...
- *Classification*: SPAM filters, character recognition, ...
- *Clustering*: Find groups in social networks, ...

Hadoop has a *general-purpose compute model.*

Arbitrary *ML algorithms*
can be implemented
using *MapReduce.*

However, recall that
MapReduce is very
course-grained...

Pattern

A new toolkit for writing
models in SAS, etc.,
then run them on
Cascading.

Spark

An alternative to
MapReduce with more
flexible and composable
abstractions.

Spark

Originally designed for
Machine Learning.

However you *train* the
models...

Train with Hadoop, etc.

*Serve requests with
NoSQL store.*

Social Network Data



Copyright © 2011-2013, Dean Wampler, All Rights Reserved

Thursday, June 6, 13

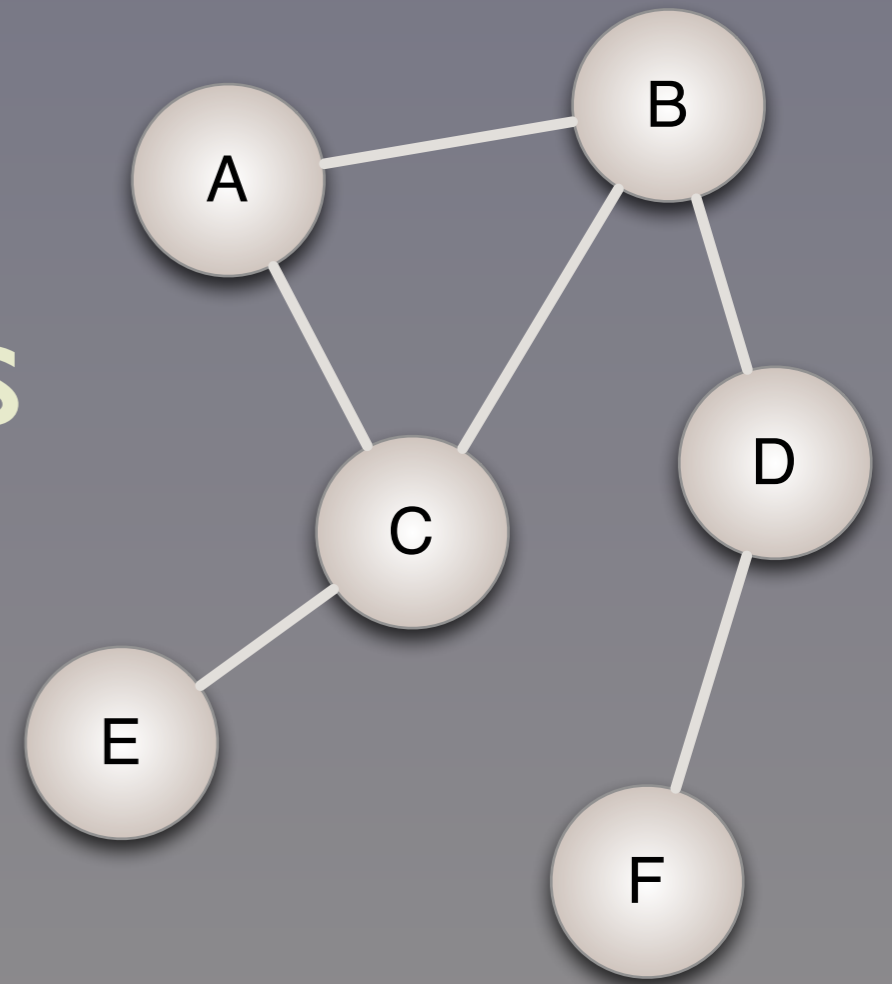
How should you represent associations in social networks, e.g., friends on Facebook, followers on Twitter, ...

Google's Page Rank

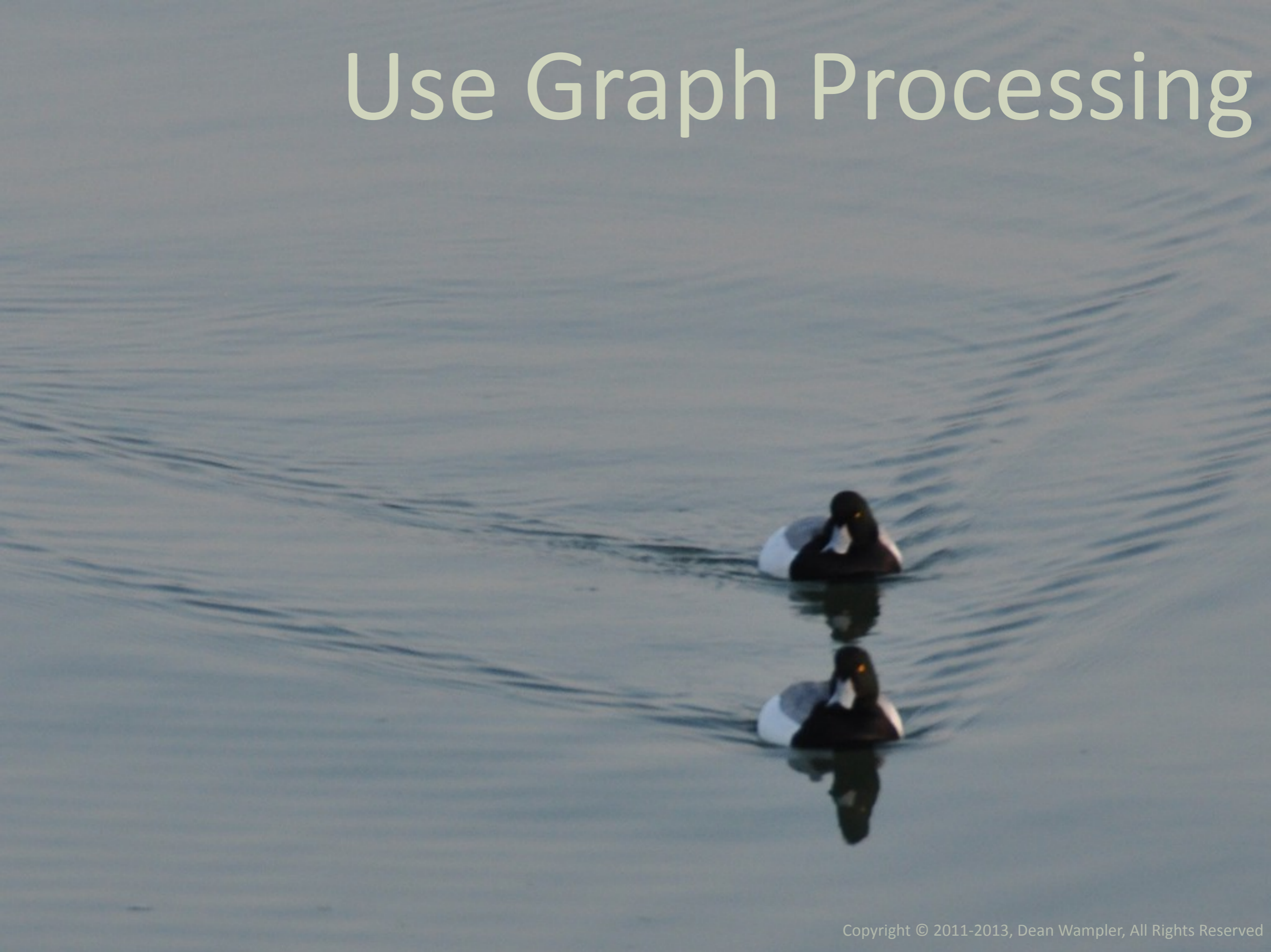
Google invented MapReduce,
... but MapReduce is not ideal for
Page Rank and other graph
algorithms.

Why not MapReduce?

- 1 MR job for each iteration that updates all n nodes/edges.
- Graph saved to disk after each iteration.
- ...



Use Graph Processing



Copyright © 2011-2013, Dean Wampler, All Rights Reserved

Thursday, June 6, 13

A good summary presentation: <http://www.slideshare.net/shatteredNirvana/pregel-a-system-for-largescale-graph-processing>

Photo: San Francisco Bay

Google's Pregel

- Pregel: New graph framework for Page Rank.
- Bulk, Synchronous Parallel (BSP).
 - Graphs are first-class citizens.
 - Efficiently processes updates...

Open-source Alternatives

- Apache Giraph.
- Apache Hama.
- Aurelius Titan.

All are
somewhat
immature.

Open-source Alternatives

- Neo4J.
- Mature, single machine graphical networks.

Neo4J

- Not the same kind of distributed system like Pregel, but
 - More mature.
 - Commercially supported.
 - Maybe you don't need distributed?

Batch Mode + Event-Stream Processing?

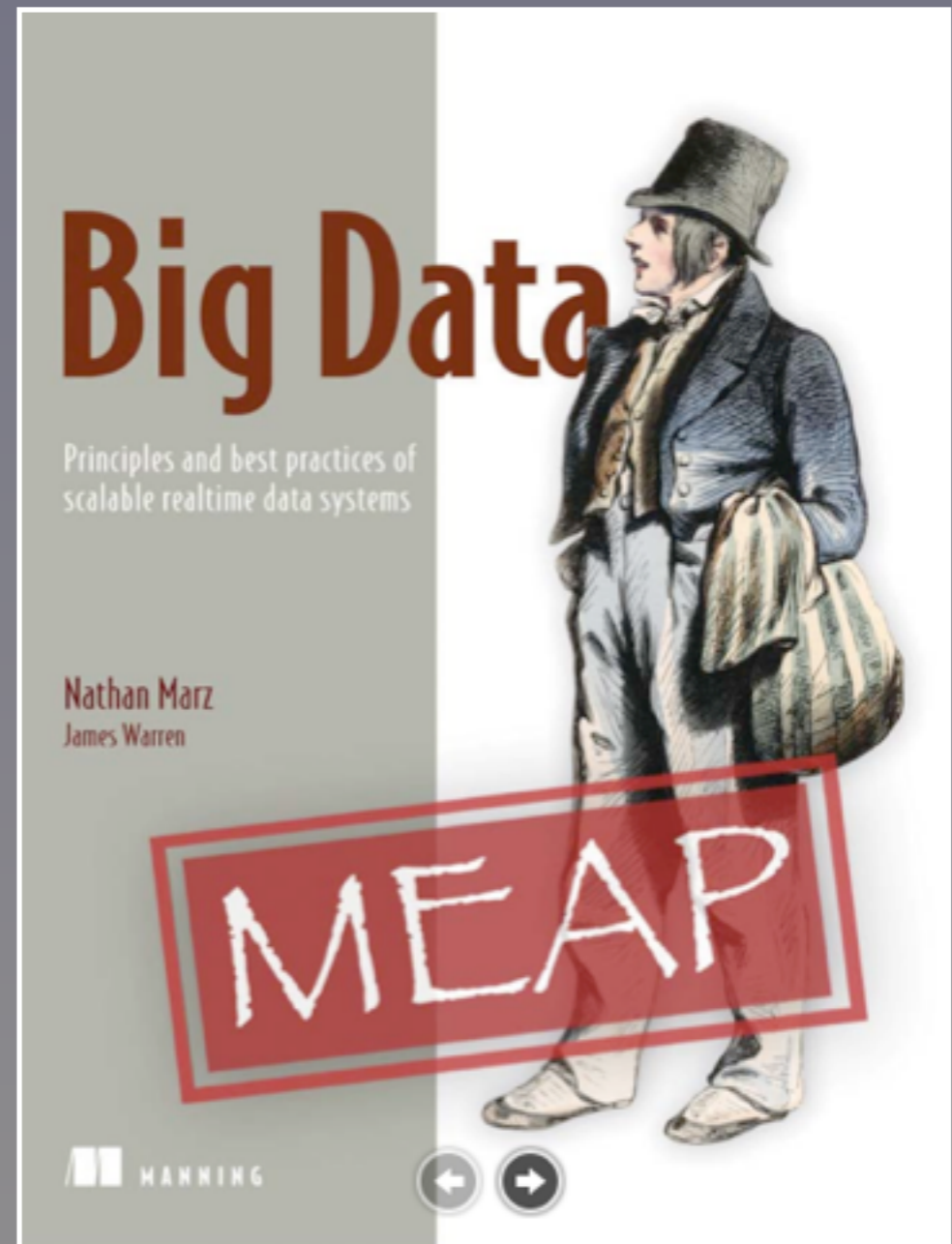
Copyright © 2011-2013, Dean Wampler, All Rights Reserved

Thursday, June 6, 13

Now that we have cataloged some issues and solutions, let's recap and look forward.
Photo: Grebe and distorted post reflection.

“Big Data”

Nathan Marz’s
vision for data
systems...



- Use Hadoop for batch-mode processing of very large data sets.
- Use Storm for event handling, e.g., increment updates.
- Use NoSQL for flexible, scalable storage.
- Stir...

So, where are we??



Copyright © 2011-2013, Dean Wampler, All Rights Reserved

Thursday, June 6, 13

Some thoughts about where the industry is and where it's headed.

Hadoop
meets lots of
needs now,
but...



Copyright © 2011-2013, Dean Wampler, All Rights Reserved

Thursday, June 6, 13



Hadoop MapReduce is the Enterprise Java Beans of our time.

Copyright © 2011-2013, Dean Wampler, All Rights Reserved

Thursday, June 6, 13

I worked with EJBs a decade ago. The framework was completely invasive into your business logic. There were too many configuration options in XML files. The framework “paradigm” was a poor fit for most problems (like soft real time systems and most algorithms beyond Word Count). Internally, EJB implementations were inefficient and hard to optimize, because they relied on poorly considered object boundaries that muddled more natural boundaries. (I’ve argued in other presentations and my “FP for Java Devs” book that OOP is a poor modularity tool...)

The fact is, Hadoop reminds me of EJBs in almost every way. It’s a 1st generation solution that mostly works okay and people do get work done with it, but just as the Spring Framework brought an essential rethinking to Enterprise Java, I think there is an essential rethink that needs to happen in Big Data, specifically around Hadoop. The functional programming community, is well positioned to create it...



I see Hadoop
as
transitional.

Copyright © 2011-2013, Dean Wampler, All Rights Reserved

Thursday, June 6, 13

At least MapReduce. HDFS has longer legs, although it has its own flaws, like a single-point of failure NameNode that was only recently fixed (with replicated NNs), but is still immature, and it has upper bounds on storage. The MapR File System is superior in all these ways, but proprietary. Anyway, HDFS will last longer, but eventually be replaced by “next generation” distributed file systems, but MapReduce is already in decline and being replaced by more performant engines such as Impala, Spark, and Storm (for different purposes...).

What about NoSQL??



Copyright © 2011-2013, Dean Wampler, All Rights Reserved

Thursday, June 6, 13

Does it feel “dated” like Hadoop?

The “good” NoSQL systems have all the right qualities, like minimal, focused abstractions.

Purpose-built Tools



Thursday, June 6, 13

I see that a trend where completely generic tooling is giving way to more “purpose-built” tooling...

ElasticSearch and Solr for Search.

New Hadoop file formats, optimize access (e.g., Parquet).

In the quest for ever better performance over massive datasets, the generic file formats in Hadoop and MapReduce are hitting a performance wall (although not everyone agrees). Parquet is column oriented & contains the data schema, like Thrift, Avro, and Protobuf. It will be exploited to optimize queries over massive data sets, much faster than the older file formats. Similarly, Impala is purpose built optimized query engine (that relies on Parquet).

New compute engines in Hadoop that replace MapReduce.

Thursday, June 6, 13

In the quest for ever better performance over massive datasets, the generic file formats in Hadoop and MapReduce are hitting a performance wall (although not everyone agrees). Parquet is column oriented & contains the data schema, like Thrift, Avro, and Protobuf. It will be exploited to optimize queries over massive data sets, much faster than the older file formats. Similarly, Impala is purpose built optimized query engine (that relies on Parquet).

Machine Learning and
other advanced analytics
are hard to write
and perform poorly
on Hadoop...

... but alternatives are
emerging, including
Spark, Storm, and
proprietary solutions.

A Final Emerging Trend...



Thursday, June 6, 13

Both are examples of technologies focused on particular problems.
photo: Trump hotel and residences on the Chicago River.

Probabilistic Programming

- Languages for Probabilistic Graphical Models??
- Bayesian Networks.
- Markov Chains.



Final Thoughts



Copyright © 2011-2013, Dean Wampler, All Rights Reserved

Thursday, June 6, 13

Photo: Gull on a pier near Fort Mason, SF

Consider Small-batch, Artisanal Data...

Thursday, June 6, 13

If you have lots of data, it can be valuable, but you also have lots of work to do to manage and exploit it. Finding the right amount of “curated” data to use and a matching architecture still have valuable benefits, as always. Unfortunately, I see people jump on the Big Data bandwagon who think there’s gold in their few GBs of data... That said, the “medium data” market is not well served at the moment. Hadoop can be overkill, yet traditional tools can be too expensive or not handle the load...

SQL Strikes Back!



Copyright © 2011-2013, Dean Wampler, All Rights Reserved

Thursday, June 6, 13

NoSQL solves meets lots of requirements better than traditional RDBMSs, but people loves them some SQL!!

Don't overlook SQL

- It's entrenched.
- Organizations want a SQL solution if they can have it.

Hadoop owes a lot of
its popularity to Hive!

Some “NoSQL”
databases have added
query languages (e.g.,
Cassandra, MongoDB).

“NewSQL” databases
are bringing NoSQL
performance to the
relational model.

Examples

- Google Spanner and F1.
- NuoDB.
- VoltDB.

Spanner is the successor to BigTable. It is a globally-distributed database (consistency is maintained using the Paxos algo. and hardware synchronized clocks through GPS and atomic clocks!) Each table requires a primary key. F1 is an RDBMS built on top of it.

NuoDB is a cloud based RDBMS.

VoltDB is an example “in-memory” database, which are ideal for lots of small transactions that leverage indexing and rarely require full table scans.

What does this mean for NoSQL?

- *People love SQL...*
- *but NoSQL meets other requirements.*

Questions?

NoSQL Search Roadshow
San Francisco, June 6, 2013
dean@concurrentthought.com
[@deanwampler](https://twitter.com/deanwampler)
polyglotprogramming.com/talks



SOME RIGHTS RESERVED

Copyright © 2011-2013, Dean Wampler, All Rights Reserved

Thursday, June 6, 13

All pictures Copyright © Dean Wampler, 2011-2013, All Rights Reserved. All other content is free to use, but attribution is requested.

Photo: Same sunrise, in Burlingame.