Big Data Infrastructure

Session 2: From Business Intelligence to Data Science

Jimmy Lin
University of Maryland
Monday, February 2, 2015

Why big data?

Science
Engineering
Commerce

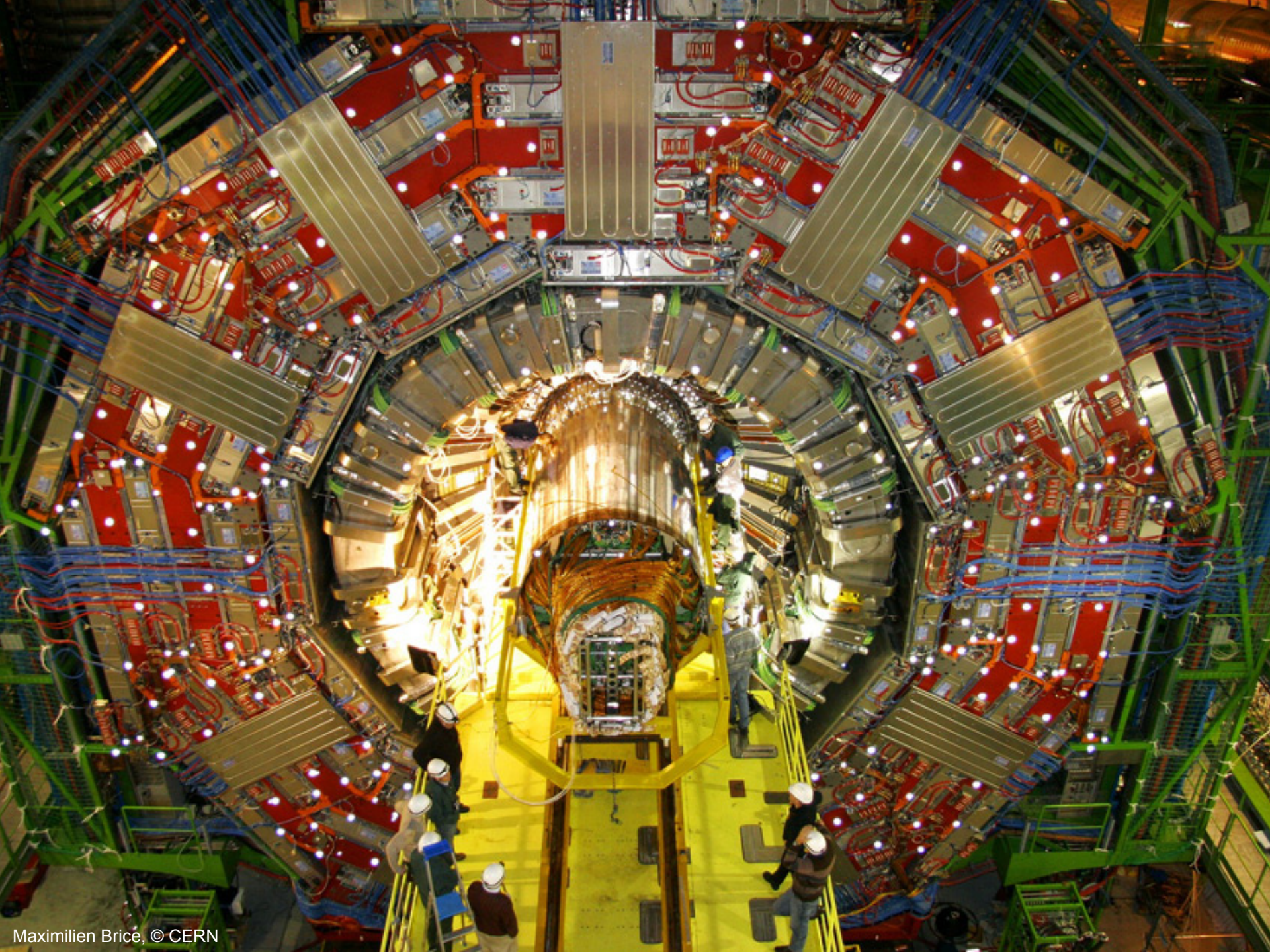
Source: Wikipedia (Everest)

Science

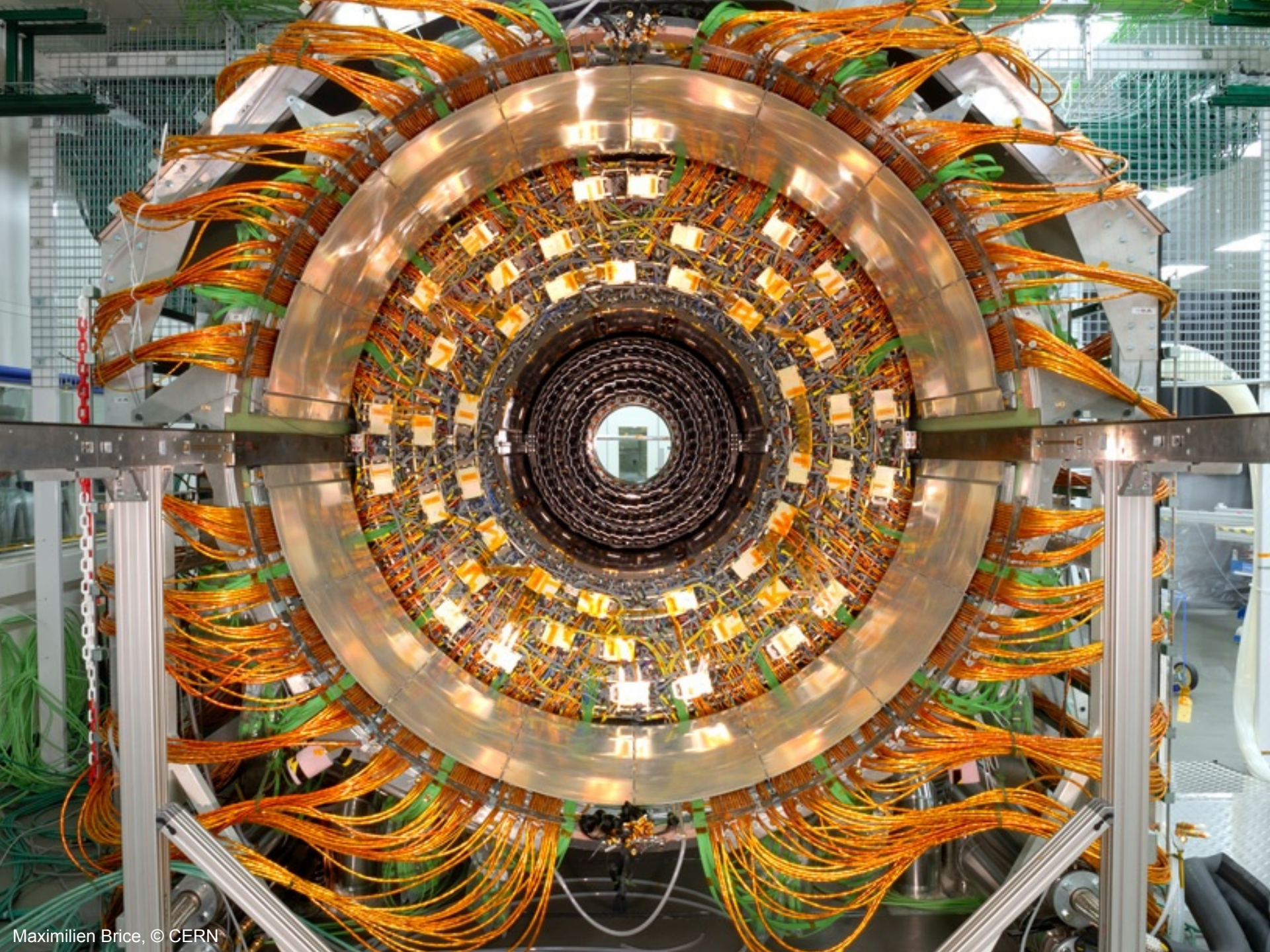
Emergence of the 4th Paradigm

Data-intensive e-Science

Maximilien Brice, © CERN



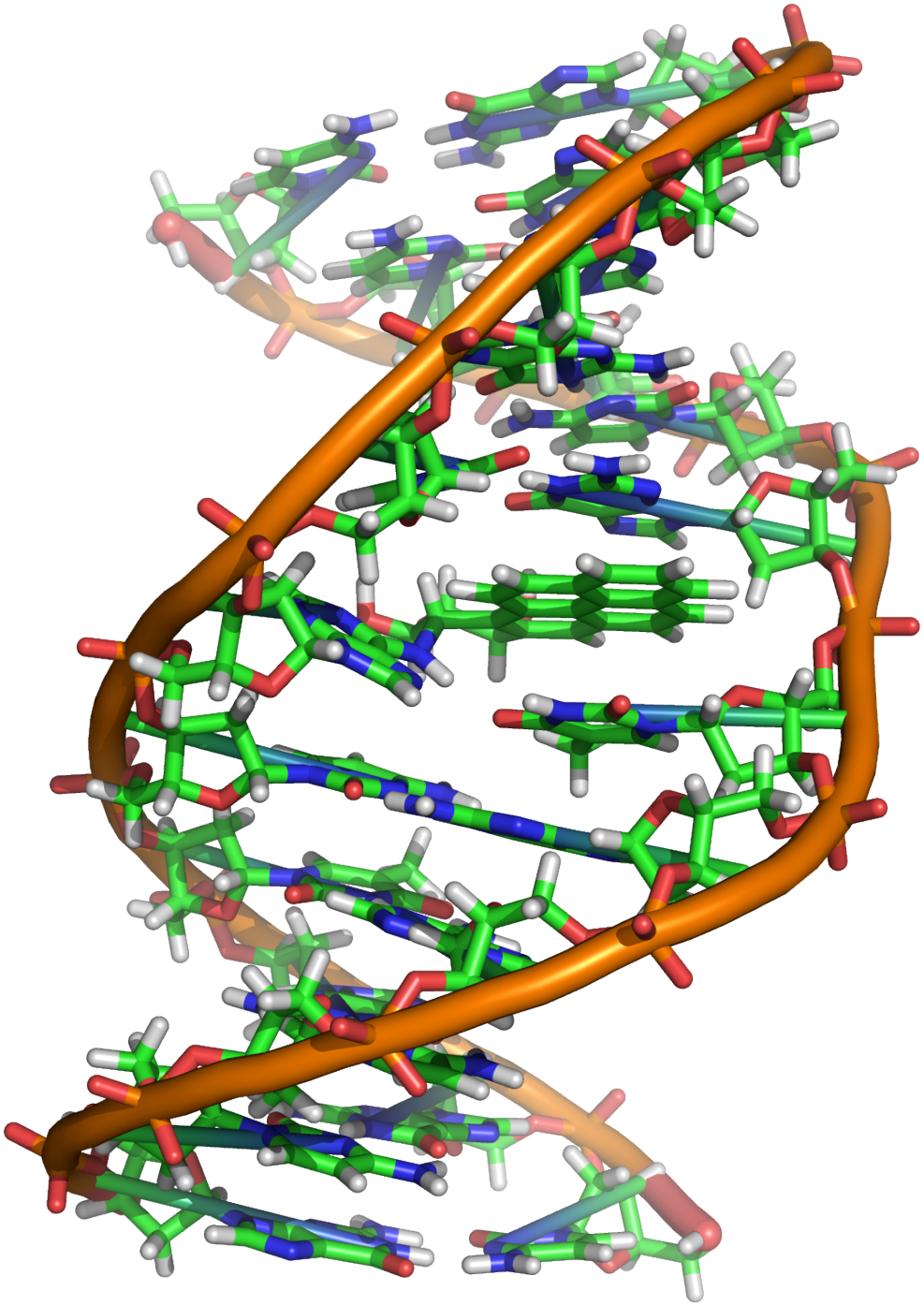
Maximilien Brice, © CERN



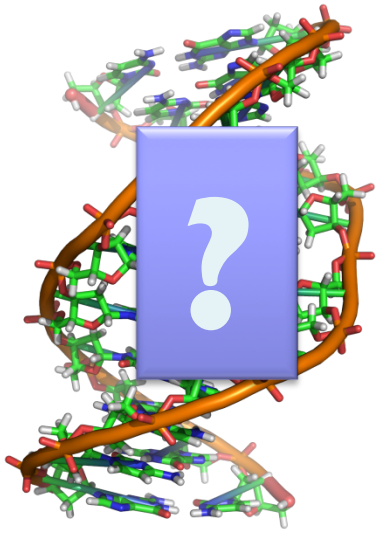
Maximilien Brice, © CERN

Source: Wikipedia (Hurricane)

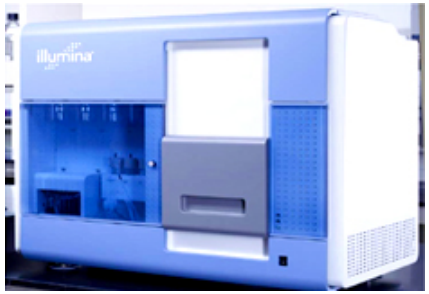
Source: Wikipedia (Galaxy)



Source: Wikipedia (DNA)



Subject genome



Sequencer

GATGCTTACTATGCGGGCCCC
CGGTCTAATGCTTACTATGC
GCTTACTATGCGGGCCCCTT
AATGCTTACTATGCGGGCCCCTT
TAATGCTTACTATGC
AATGCTTAGCTATGCGGGC
AATGCTTACTATGCGGGCCCCTT
AATGCTTACTATGCGGGCCCCTT
CGGTCTAGATGCTTACTATGC
AATGCTTACTATGCGGGCCCCTT
CGGTCTAATGCTTAGCTATGC
ATGCTTACTATGCGGGCCCCTT

Reads

Human genome: 3 gbp
A few billion short reads
(~100 GB compressed data)

Engineering

The unreasonable effectiveness of data

Count and normalize!

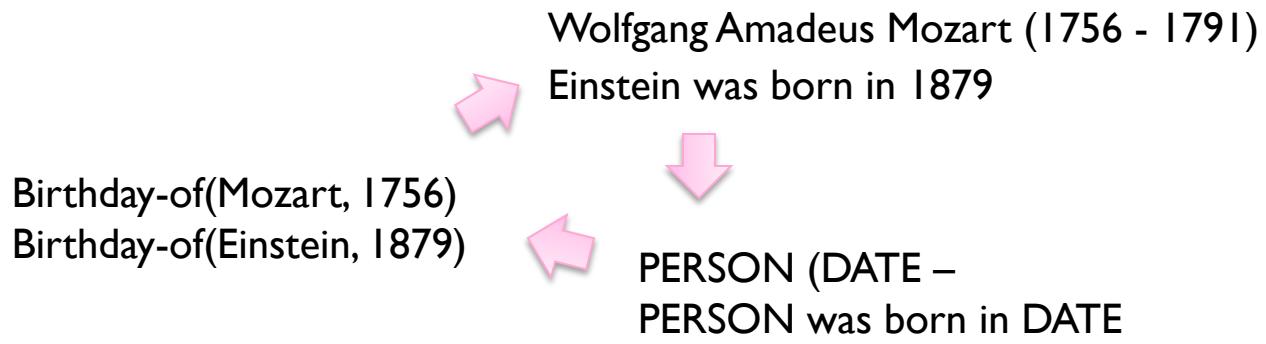
Source: Wikipedia (Three Gorges Dam)

What to do with more data?

- Answering factoid questions
  - Pattern matching on the Web
  - Works amazingly well

Who shot Abraham Lincoln? → X shot Abraham Lincoln

- Learning relations
  - Start with seed instances
  - Search for patterns on the Web
  - Using patterns to find more instances



(Brill et al., TREC 2001; Lin, ACM TOIS 2007)
(Agichtein and Gravano, DL 2000; Ravichandran and Hovy, ACL 2002; …)

Translate

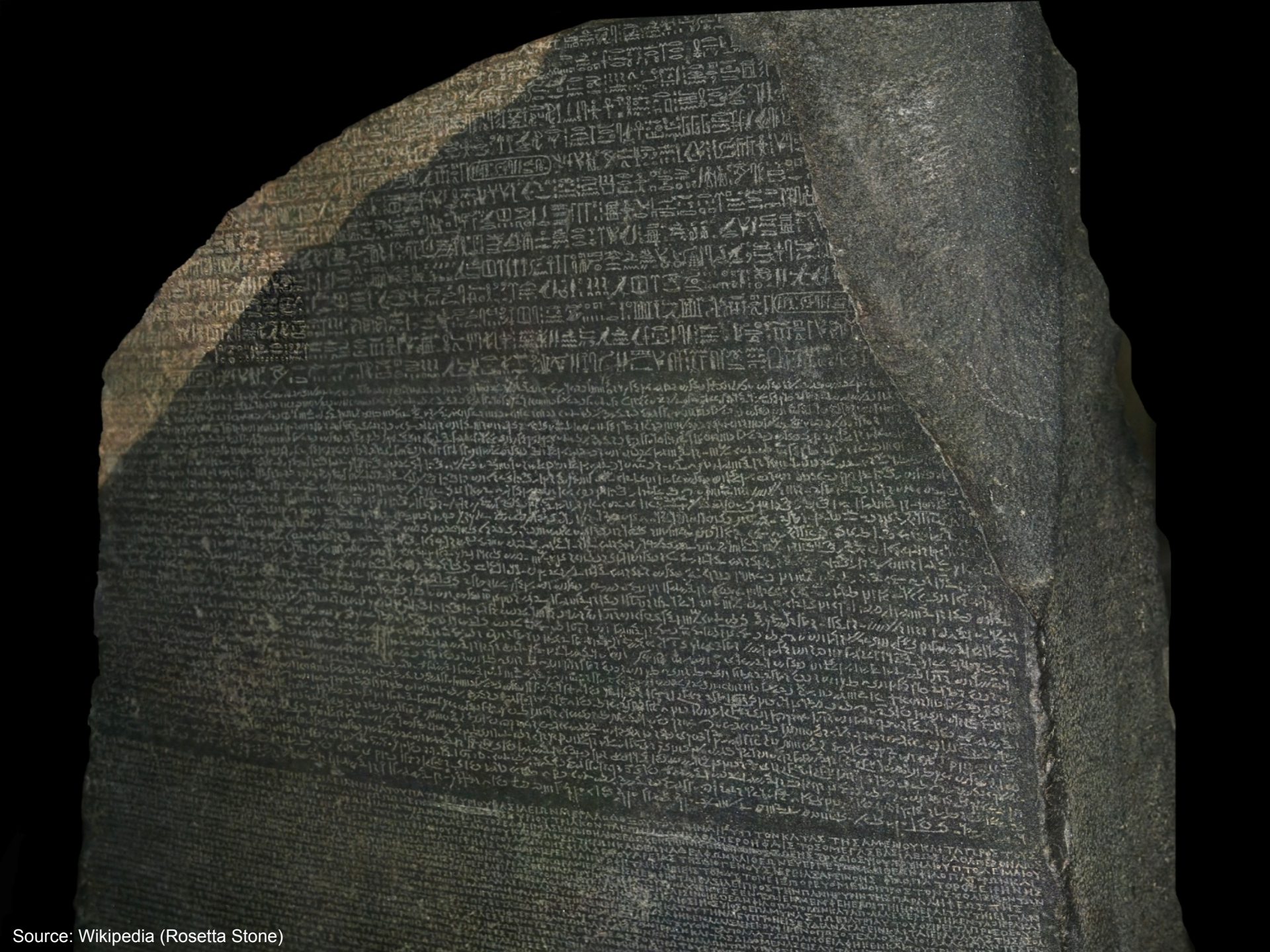
English | Spanish | French | English - detected

How does Google's translation system work?

如何谷歌的翻译系统的工作?

Wrong?

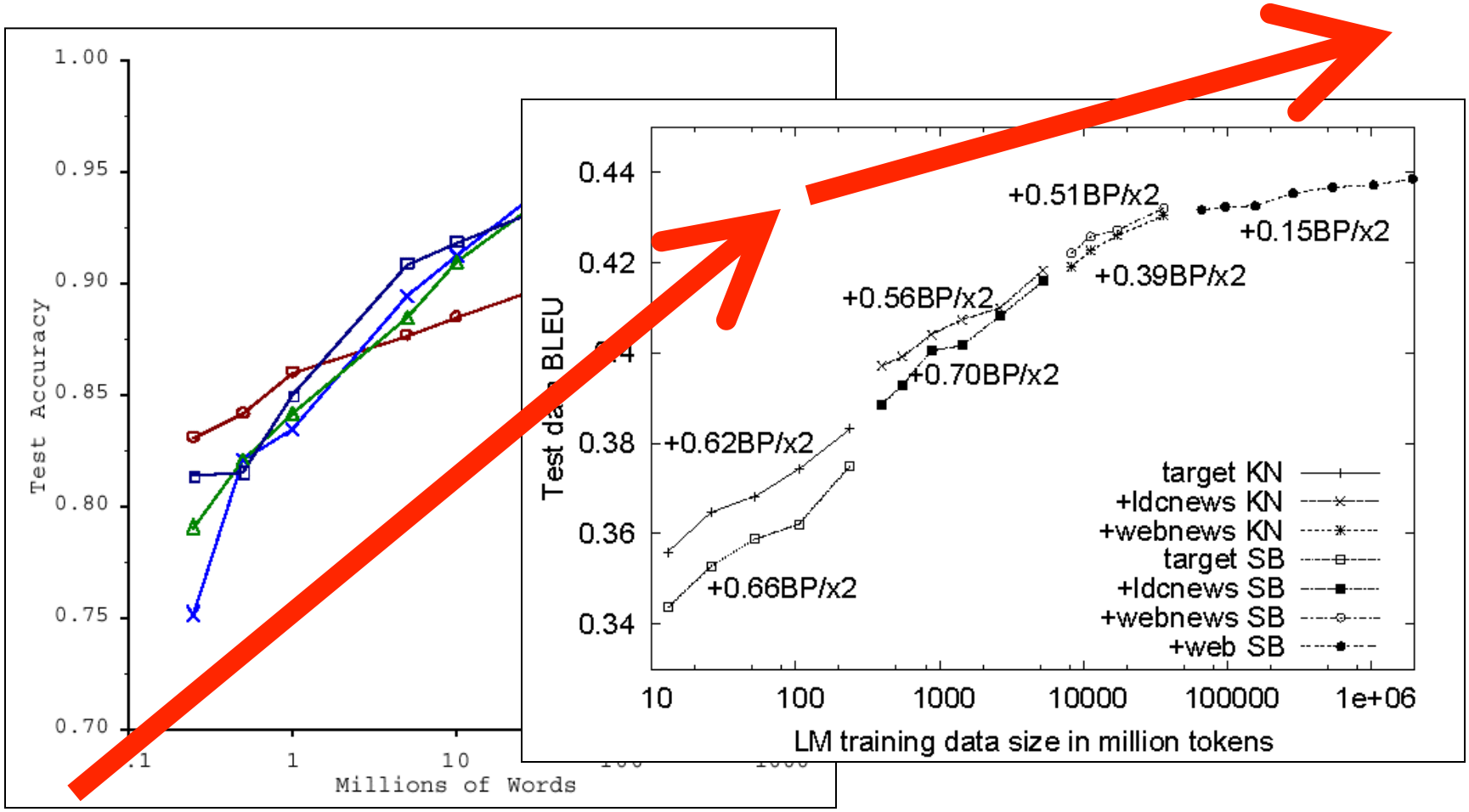
Rúhé gǔgē de fānyì xìtǒng de gōngzuò?



Source: Wikipedia (Rosetta Stone)

No data like more data!

s/knowledge/data/g;



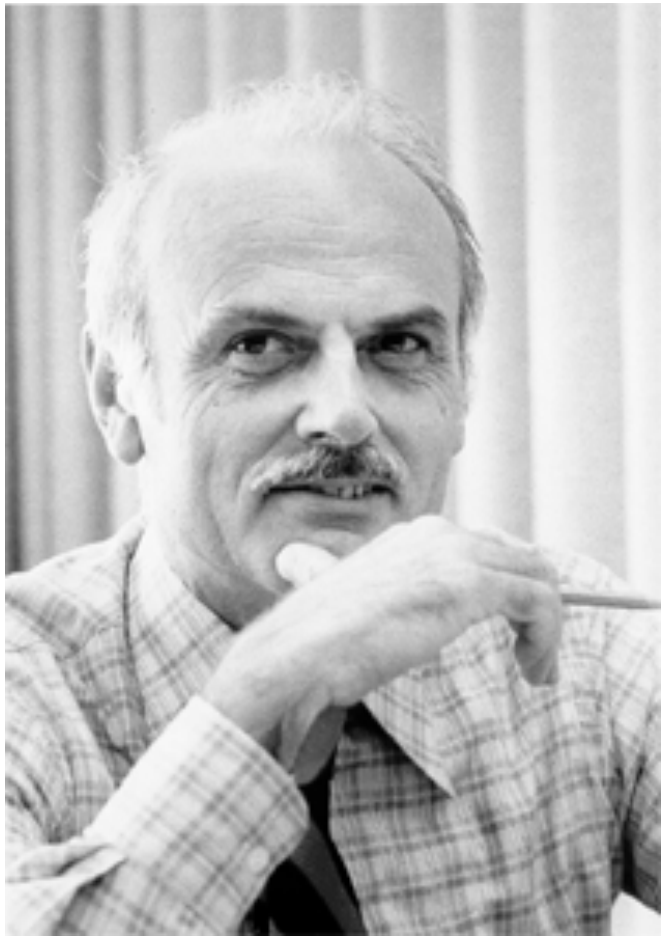
(Banko and Brill, ACL 2001)
(Brants et al., EMNLP 2007)


Know thy customers
Data → Insights → Competitive advantages
Commerce
EPSON
PEARL BAR
Source: Wikiedia (Shinjuku, Tokyo)

Business Intelligence

An organization should retain data that result from carrying out its mission and exploit those data to generate insights that benefit the organization, for example, market analysis, strategic planning, decision making, etc.

Duh!?



Source: Wikipedia

Database Workloads

- OLTP (online transaction processing)
  - Typical applications: e-commerce, banking, airline reservations
  - User facing: real-time, low latency, highly-concurrent
  - Tasks: relatively small set of "standard" transactional queries
  - Data access pattern: random reads, updates, writes (involving relatively small amounts of data)
- OLAP (online analytical processing)
  - Typical applications: business intelligence, data mining
  - Back-end processing: batch workloads, less concurrency
  - Tasks: complex analytical queries, often ad hoc
  - Data access pattern: table scans, large amounts of data per query

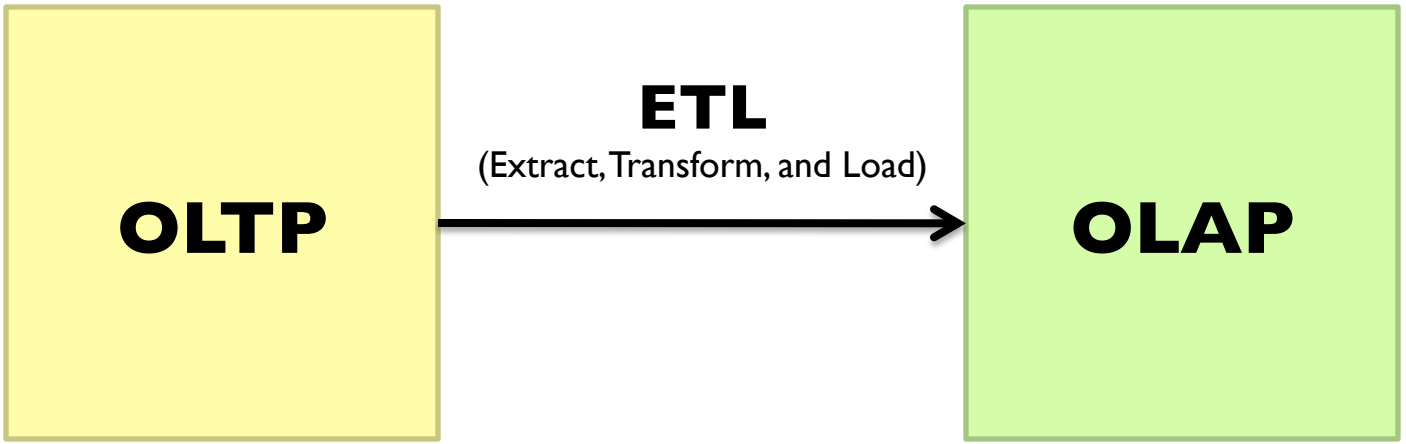
One Database or Two?

- Downsides of co-existing OLTP and OLAP workloads
  - Poor memory management
  - Conflicting data access patterns
  - Variable latency
- Solution: separate databases
  - User-facing OLTP database for high-volume transactions
  - Data warehouse for OLAP workloads
  - How do we connect the two?

Data Warehousing

Source: Wikipedia (Warehouse)

OLTP/OLAP Architecture



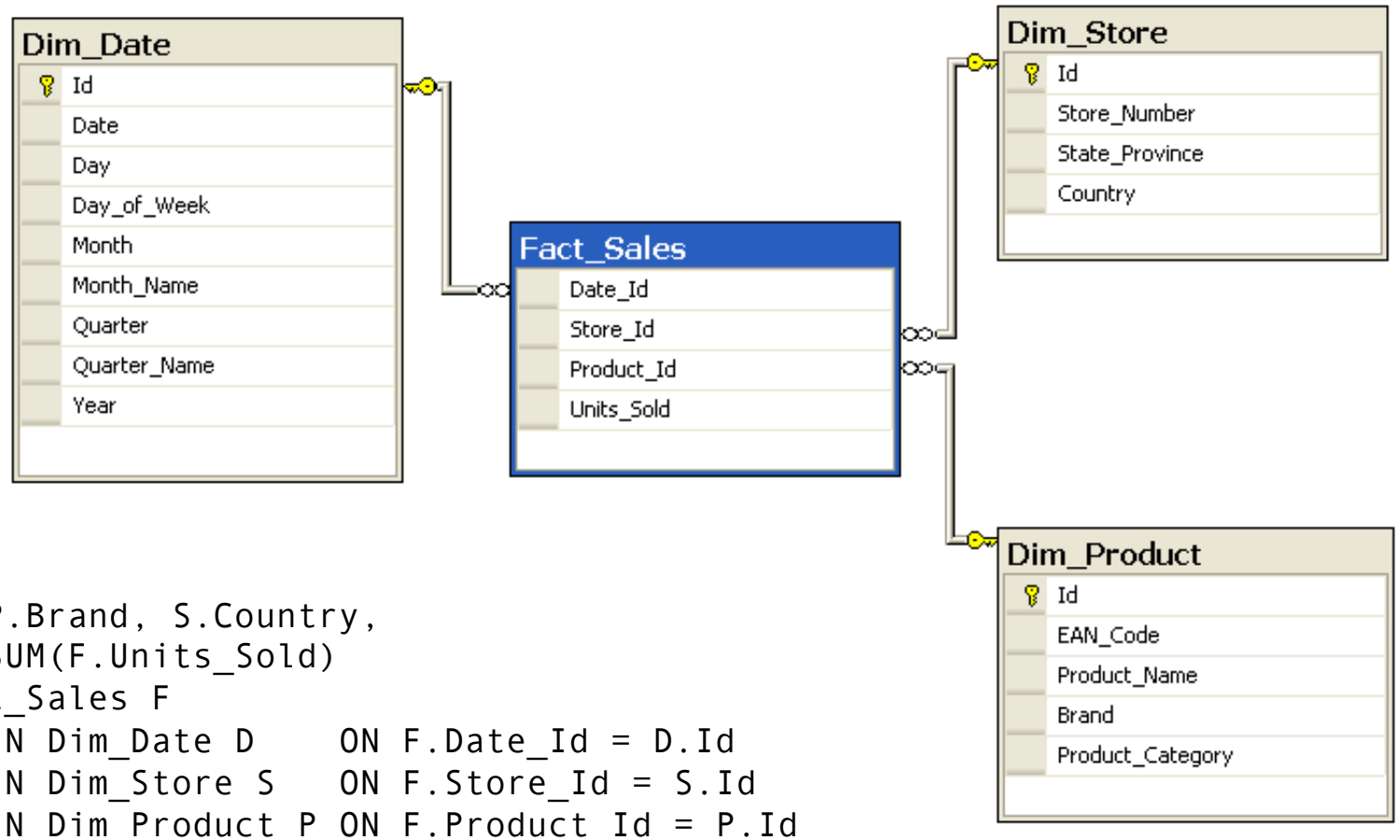
OLTP/OLAP Integration

- OLTP database for user-facing transactions
- Extract-Transform-Load (ETL)
  - Extract records from source
  - Transform: clean data, check integrity, aggregate, etc.
  - Load into OLAP database
- OLAP database for data warehousing

What do you actually do?

- Dashboards, report generation
- *Ad hoc* analyses

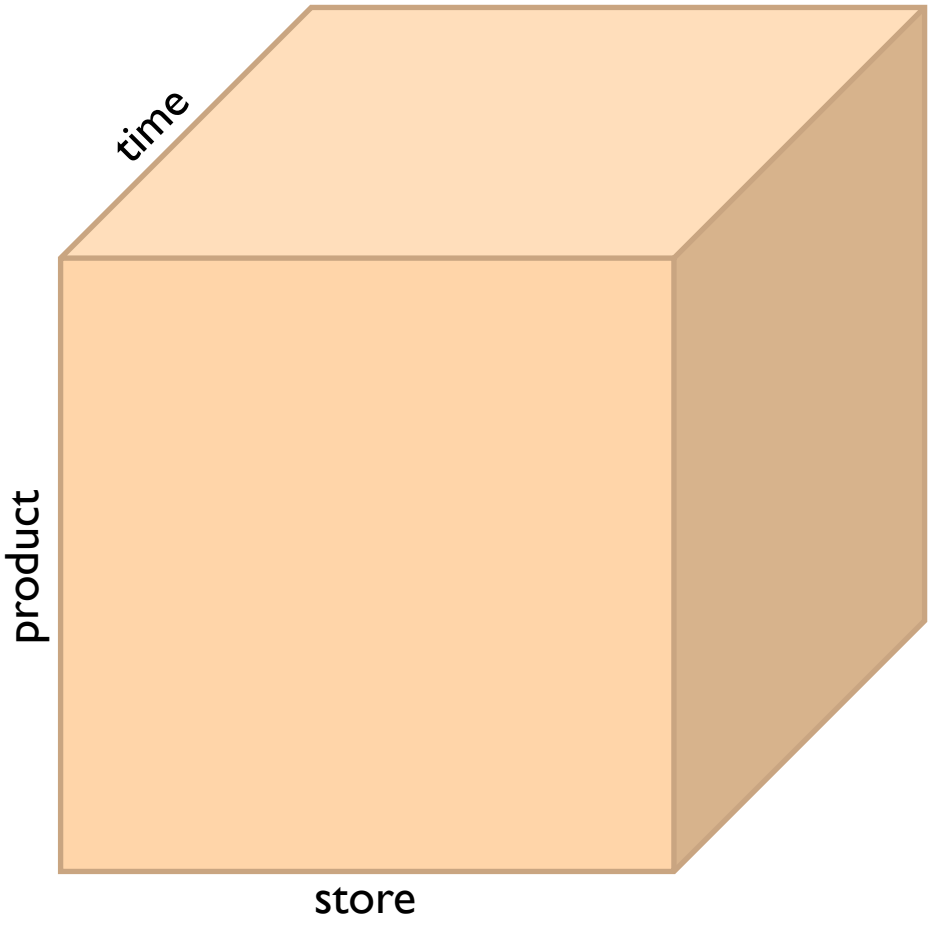
Structure of Data Warehouses




```
SELECT  P.Brand, S.Country,
        SUM(F.Units_Sold)
FROM Fact_Sales F
INNER JOIN Dim_Date D    ON F.Date_Id = D.Id
INNER JOIN Dim_Store S   ON F.Store_Id = S.Id
INNER JOIN Dim_Product P ON F.Product_Id = P.Id
WHERE D.YEAR = 1997 AND P.Product_Category = 'tv'
GROUP BY P.Brand, S.Country;
```

Source: Wikipedia (Star Schema)

OLAP Cubes



Common operations

slice and dice

roll up/drill down

pivot

OLAP Cubes: Research Challenges

- Fundamentally, lots of group-bys and aggregations
  - How to take advantage of schema structure to avoid repeated work?
- Cube materialization
  - Realistic to materialize the entire cube?
  - If not, how/when/what to materialize?

Fast forward…

facebook®

Jeff Hammerbacher, Information Platforms and the Rise of the Data Scientist.
In, *Beautiful Data*, O'Reilly, 2009.

> "On the first day of logging the Facebook clickstream, more than 400 gigabytes of data was collected. The load, index, and aggregation processes for this data set really taxed the Oracle data warehouse. Even after significant tuning, we were unable to aggregate a day of clickstream data in less than 24 hours."

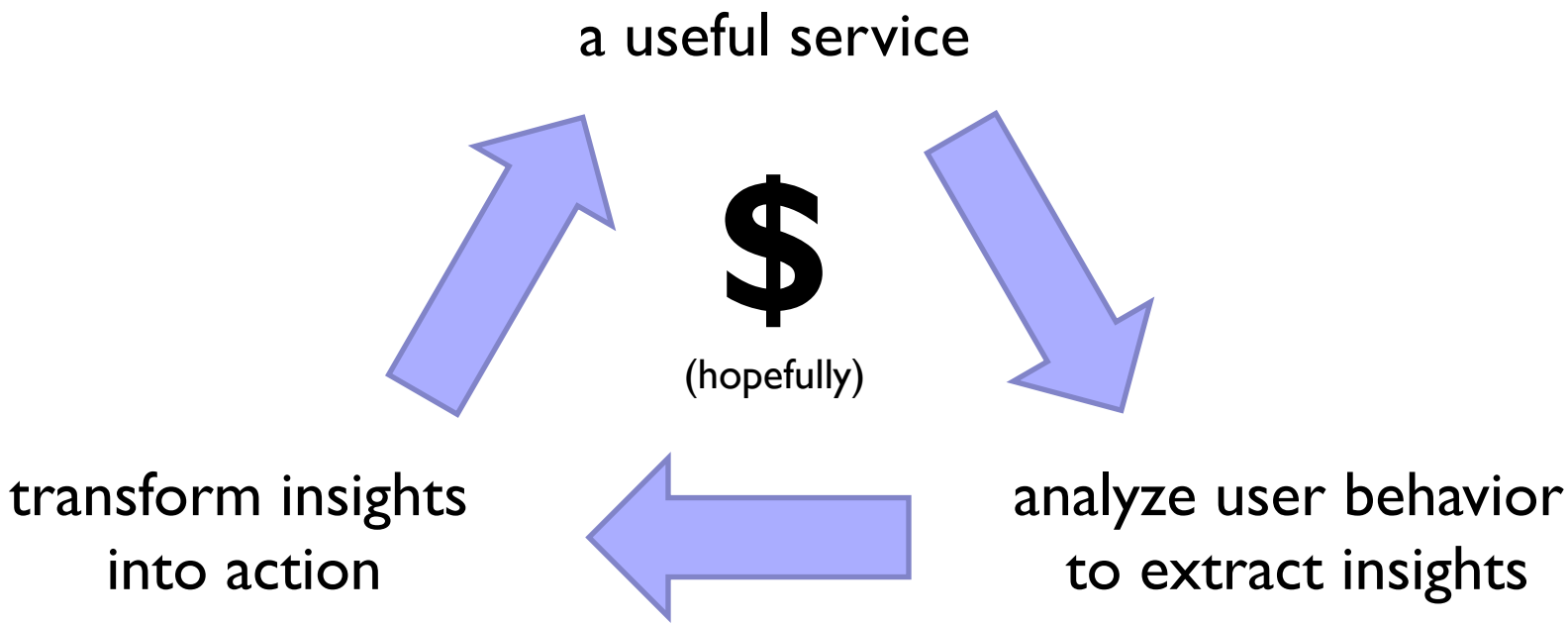
ETL Bottleneck

- ETL is typically a nightly task:
  - What happens if processing 24 hours of data takes longer than 24 hours?
- Hadoop is perfect:
  - Ingest is limited by speed of HDFS
  - Scales out with more nodes
  - Massively parallel
  - Ability to use any processing tool
  - Cheaper than parallel databases
  - ETL is a batch process anyway!

What's changed?

- Dropping cost of disks
  - Cheaper to store everything than to figure out what to throw away
- Types of data collected
  - From data that's *obviously* valuable to data whose value is less apparent
- Rise of social media and user-generated content
  - Large increase in data volume
- Growing maturity of data mining techniques
  - Demonstrates value of data analytics

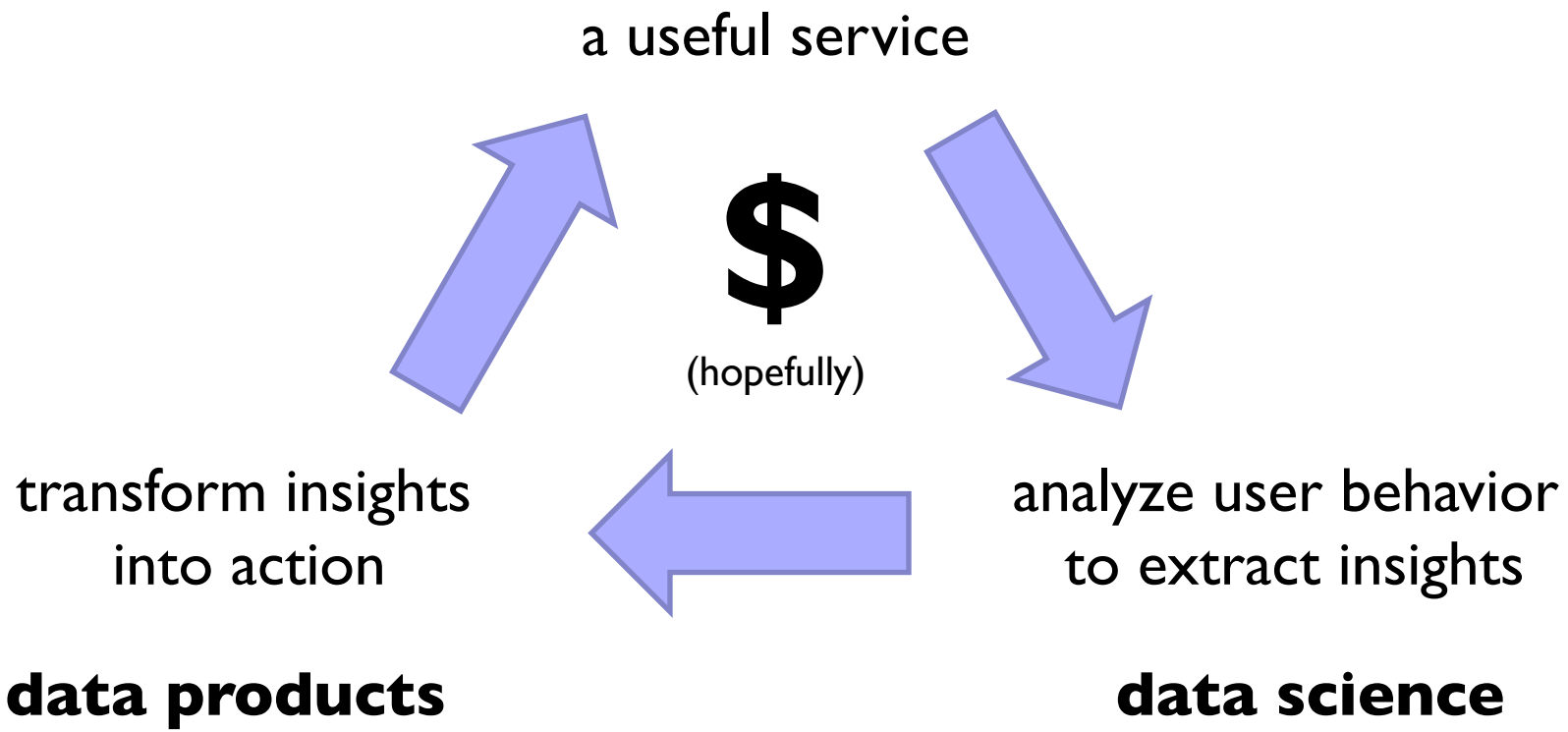
Virtuous Product Cycle

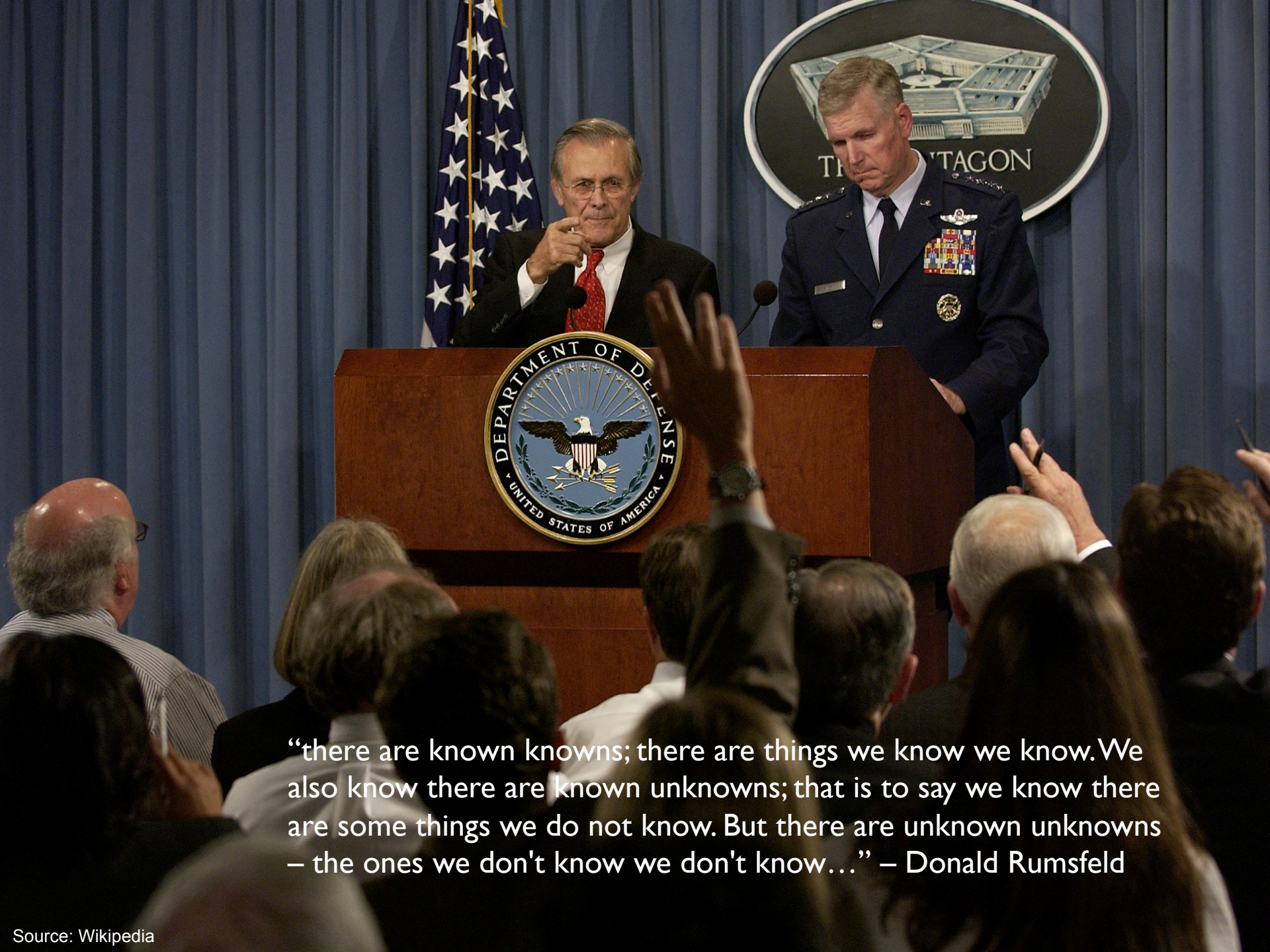


What do you actually do?

- Dashboards, report generation
- *Ad hoc* analyses
  - "Descriptive"
  - "Predictive"
- Data products

Virtuous Product Cycle





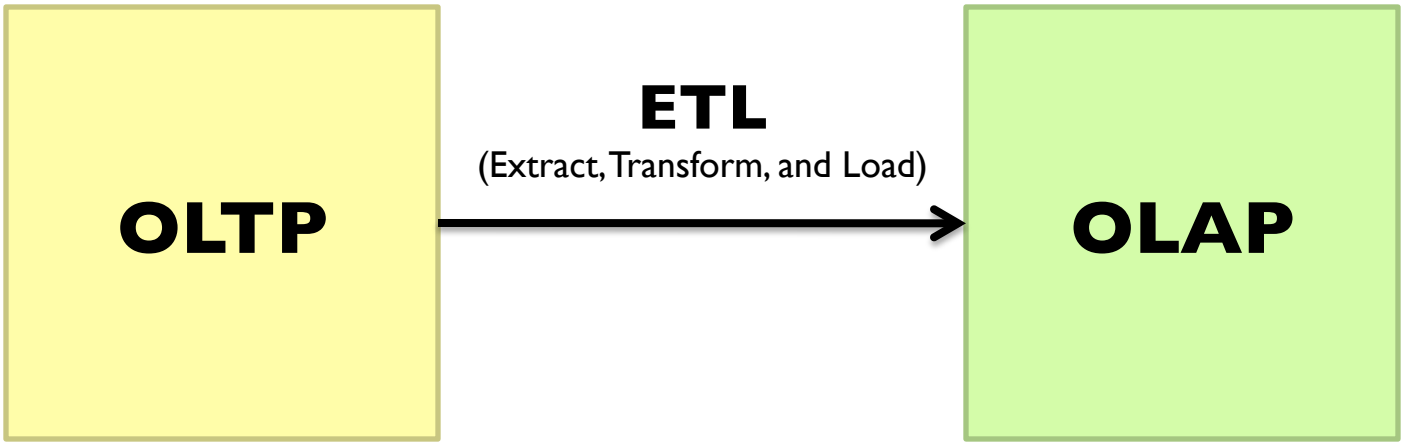
"there are known knowns; there are things we know we know. We also know there are known unknowns; that is to say we know there are some things we do not know. But there are unknown unknowns – the ones we don't know we don't know…" – Donald Rumsfeld

Source: Wikipedia

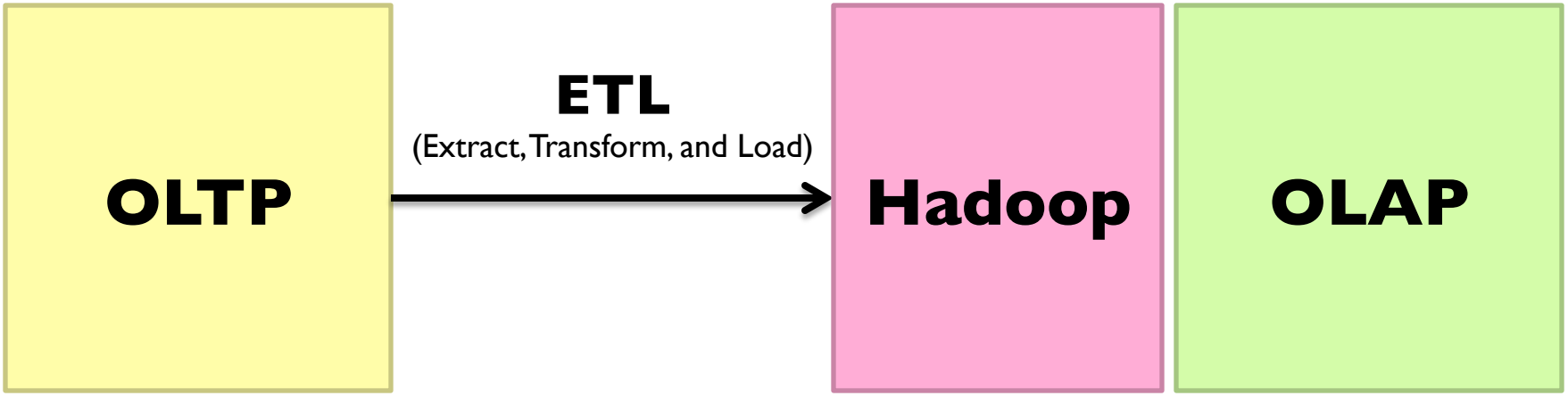
Known and Unknown Unknowns

- Databases are great if you know what questions to ask
  - "Known unknowns"
- What if you don't know what you're looking for?
  - "Unknown unknowns"

OLTP/OLAP Architecture

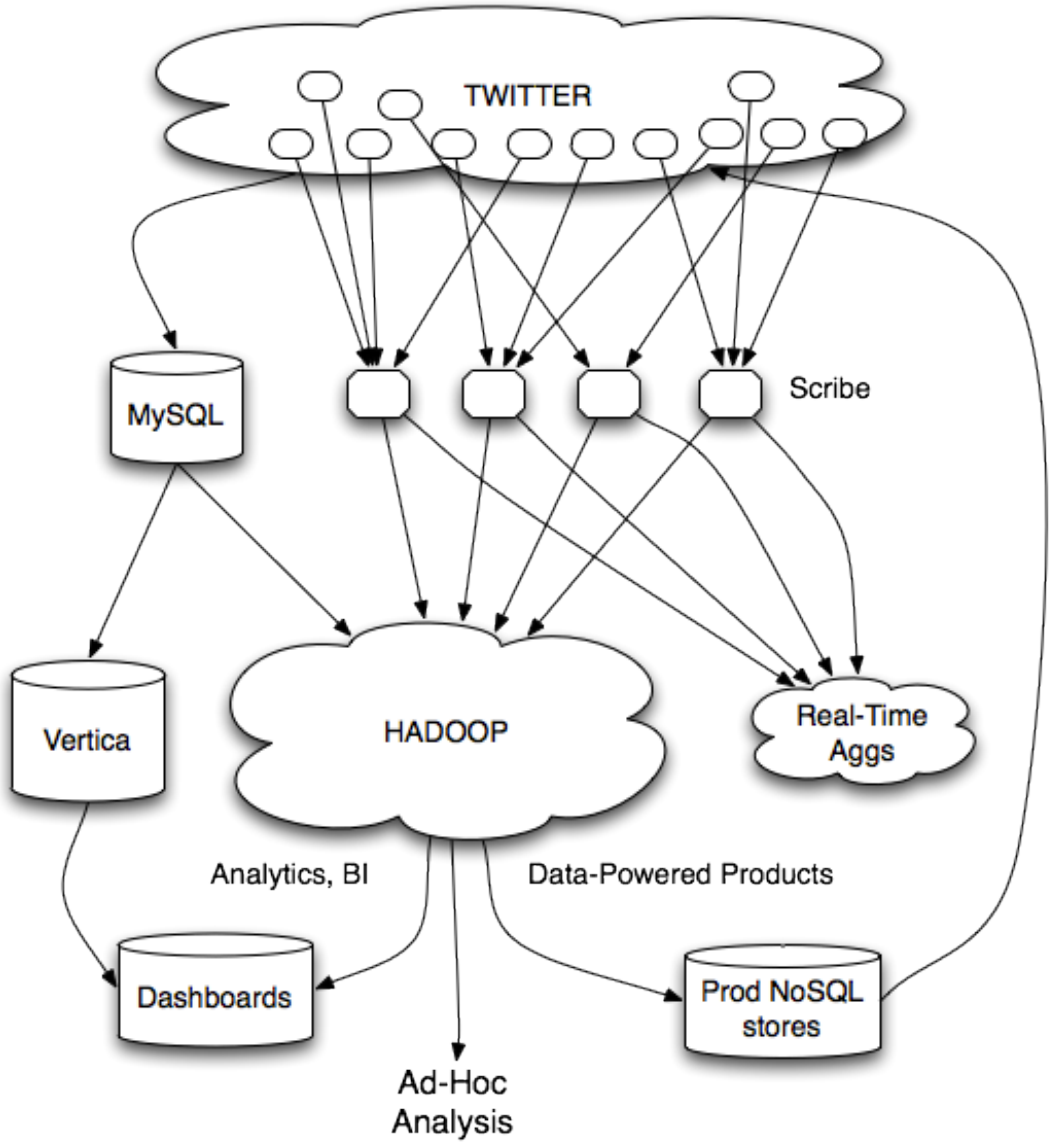


OLTP/OLAP/Hadoop Architecture



ETL: Redux

- Often, with noisy datasets, ETL *is* the analysis!
- Note that ETL necessarily involves brute force data scans
- L, then E and T?



Twitter's data warehousing architecture

circa ~2010

~150 people total

~60 Hadoop nodes

~6 people use analytics stack daily

circa ~2012

~1400 people total

10s of Ks of Hadoop nodes, multiple DCs

10s of PBs total Hadoop DW capacity

~100 TB ingest daily

dozens of teams use Hadoop daily

10s of Ks of Hadoop jobs daily

Why big data?

Science
Engineering
Commerce

Source: Wikipedia (Everest)

Source: Wikipedia (The Scream)

Source: Wikipedia (Japanese rock garden)

How will I actually learn Hadoop?

- This class session
- Hadoop: The Definitive Guide
- RTFM
- RTFC(!)

Materials in the course

- The course homepage
- Hadoop: The Definitive Guide
- Data-Intensive Text Processing with MapReduce
- Cloud[9]

Take advantage of GitHub!
clone, branch, send pull request

Source: Wikipedia (Mahout)

Basic Hadoop API*

- Mapper
  - void setup(Mapper.Context context)
    Called once at the beginning of the task
  - void map(K key, V value, Mapper.Context context)
    Called once for each key/value pair in the input split
  - void cleanup(Mapper.Context context)
    Called once at the end of the task
- Reducer/Combiner
  - void setup(Reducer.Context context)
    Called once at the start of the task
  - void reduce(K key, Iterable<V> values, Reducer.Context context)
    Called once for each key
  - void cleanup(Reducer.Context context)
    Called once at the end of the task

*Note that there are two versions of the API!

Basic Hadoop API*

- Partitioner
  - int getPartition(K key, V value, int numPartitions)
    Get the partition number given total number of partitions
- Job
  - Represents a packaged Hadoop job for submission to cluster
  - Need to specify input and output paths
  - Need to specify input and output formats
  - Need to specify mapper, reducer, combiner, partitioner classes
  - Need to specify intermediate/final key/value classes
  - Need to specify number of reducers (but not mappers, why?)
  - Don't depend of defaults!

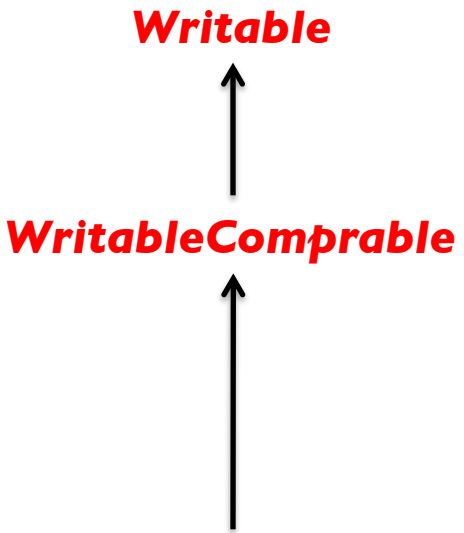
*Note that there are two versions of the API!

A tale of two packages...

org.apache.hadoop.mapreduce
org.apache.hadoop.mapred

Source: Wikipedia (Budapest)

Data Types in Hadoop: Keys and Values



Defines a de/serialization protocol. Every data type in Hadoop is a Writable.

Defines a sort order. All keys must be of this type (but not values).

Concrete classes for different data types.

SequenceFiles

Binary encoded of a sequence of key/value pairs

“Hello World”: Word Count

```
Map(String docid, String text):
    for each word w in text:
        Emit(w, 1);

Reduce(String term, Iterator<Int> values):
    int sum = 0;
    for each v in values:
        sum += v;
        Emit(term, value);
```

“Hello World”: Word Count

```
private static class MyMapper
    extends Mapper<LongWritable, Text, Text, IntWritable> {

  private final static IntWritable ONE = new IntWritable(1);
  private final static Text WORD = new Text();

  @Override
  public void map(LongWritable key, Text value, Context context)
      throws IOException, InterruptedException {
    String line = ((Text) value).toString();
    StringTokenizer itr = new StringTokenizer(line);
    while (itr.hasMoreTokens()) {
      WORD.set(itr.nextToken());
      context.write(WORD, ONE);
    }
  }
}
```

“Hello World”: Word Count

```
private static class MyReducer
    extends Reducer<Text, IntWritable, Text, IntWritable> {

  private final static IntWritable SUM = new IntWritable();

  @Override
  public void reduce(Text key, Iterable<IntWritable> values,
      Context context) throws IOException, InterruptedException {
    Iterator<IntWritable> iter = values.iterator();
    int sum = 0;
    while (iter.hasNext()) {
      sum += iter.next().get();
    }
    SUM.set(sum);
    context.write(key, SUM);
  }
}
```

Three Gotchas

- Avoid object creation if possible
  - Reuse Writable objects, change the payload
- Execution framework reuses value object in reducer
- Passing parameters via class statics

Getting Data to Mappers and Reducers

- Configuration parameters
  - Directly in the Job object for parameters
- "Side data"
  - DistributedCache
  - Mappers/reducers read from HDFS in setup method

Complex Data Types in Hadoop

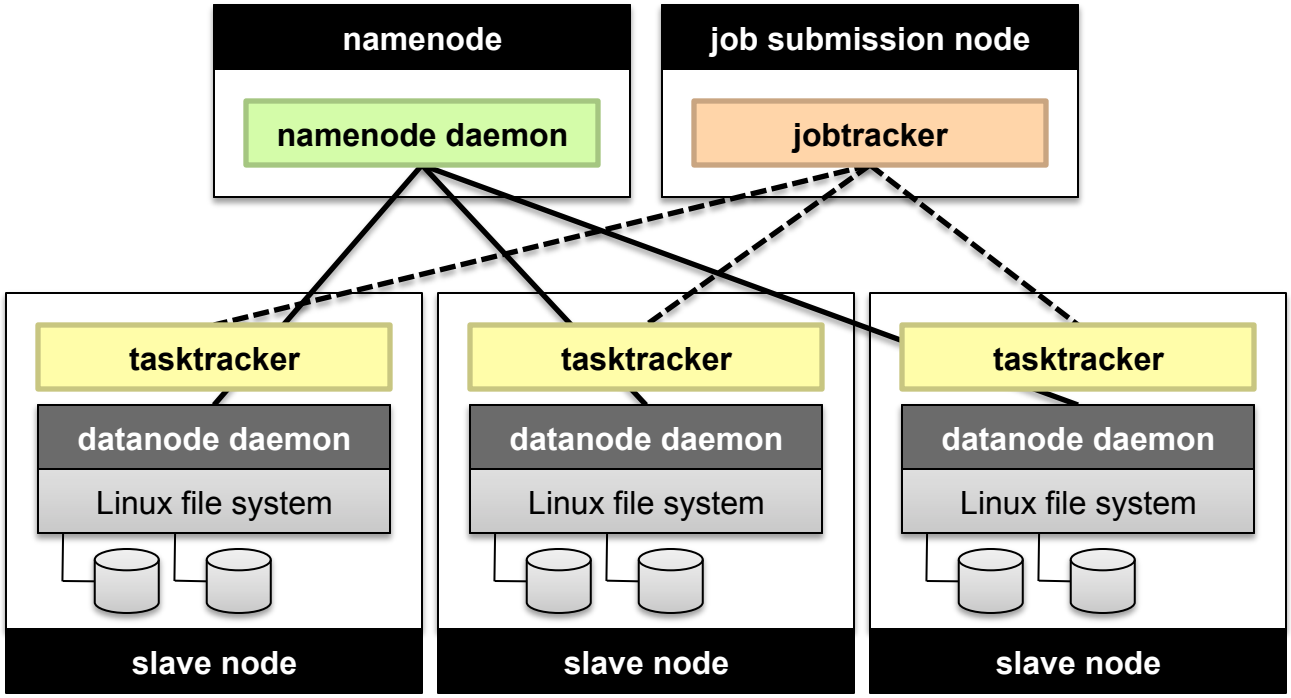
- How do you implement complex data types?
- The easiest way:
  - Encoded it as Text, e.g., (a, b) = "a:b"
  - Use regular expressions to parse and extract data
  - Works, but pretty hack-ish
- The hard way:
  - Define a custom implementation of Writable(Comprable)
  - Must implement: readFields, write, (compareTo)
  - Computationally efficient, but slow for rapid prototyping
  - Implement WritableComparator hook for performance
- Somewhere in the middle:
  - Cloud[9] (via lin.tl) offers JSON support and lots of useful Hadoop types
  - Quick tour…

Basic Cluster Components*

- One of each:
  - Namenode (NN): master node for HDFS
  - Jobtracker (JT): master node for job submission
- Set of each per slave machine:
  - Tasktracker (TT): contains multiple task slots
  - Datanode (DN): serves HDFS data blocks

* Not quite… leaving aside YARN for now

Putting everything together…

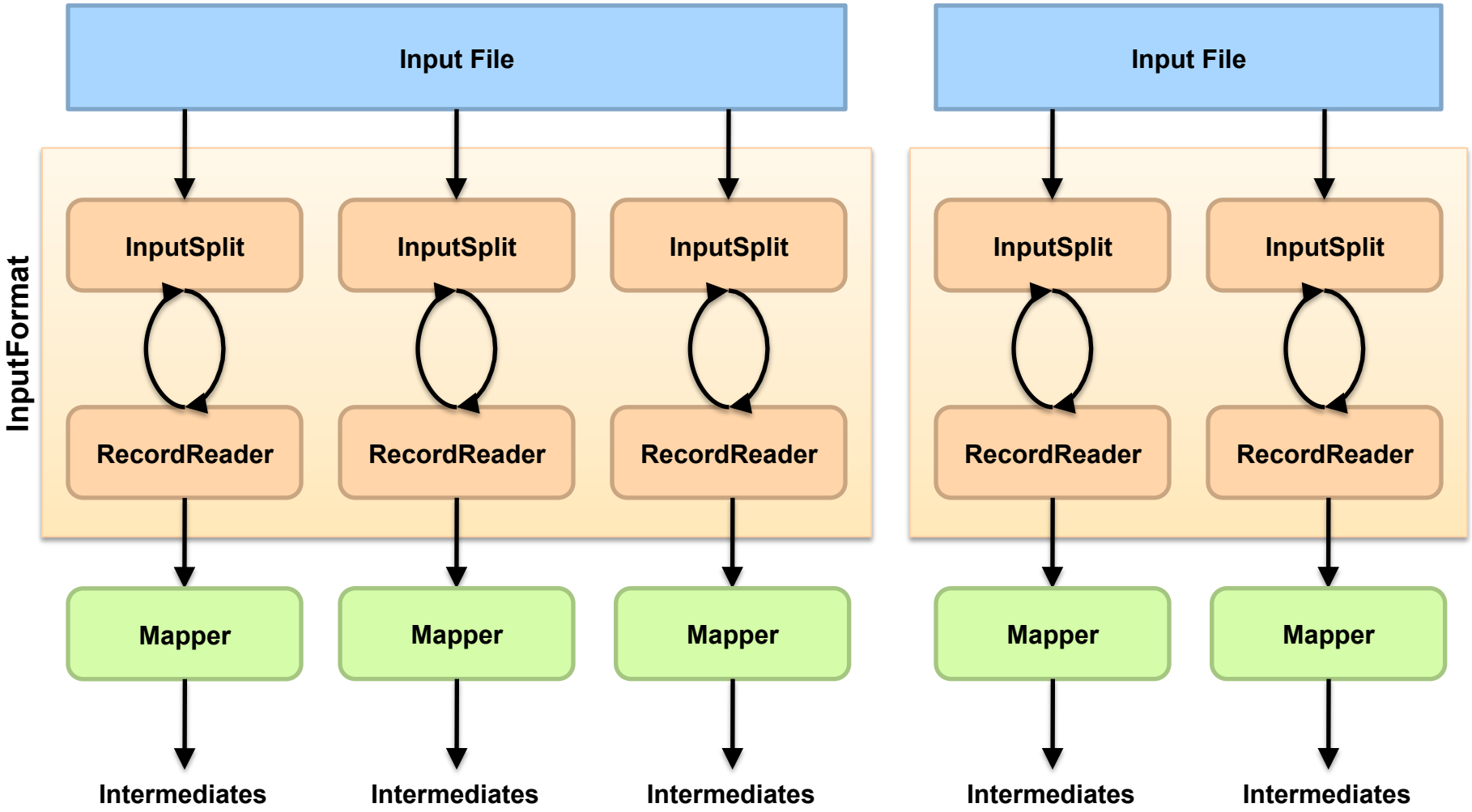


Anatomy of a Job

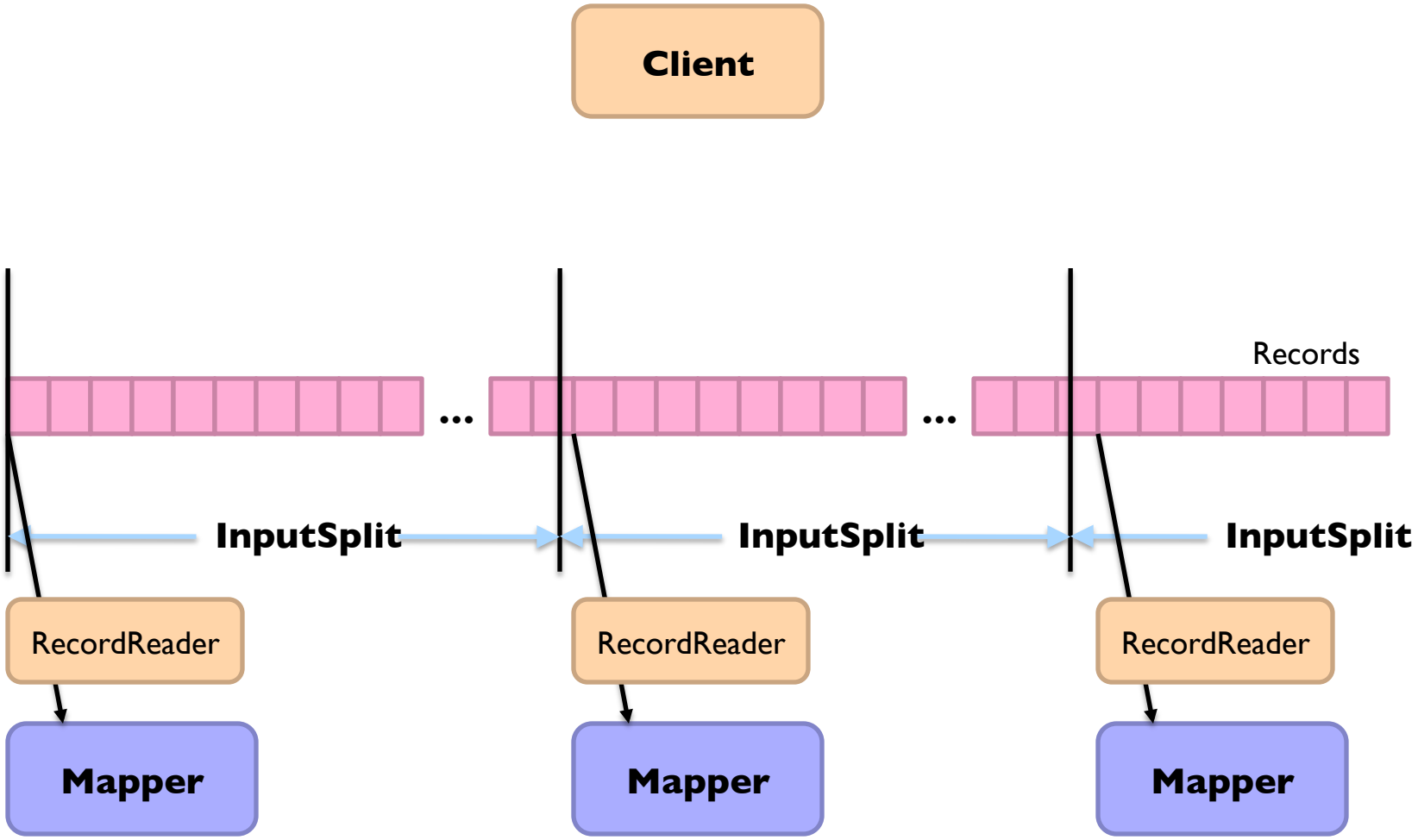
- MapReduce program in Hadoop = Hadoop job
  - Jobs are divided into map and reduce tasks
  - An instance of running a task is called a task attempt (occupies a slot)
  - Multiple jobs can be composed into a workflow
- Job submission:
  - Client (i.e., driver program) creates a job, configures it, and submits it to jobtracker
  - That's it! The Hadoop cluster takes over…

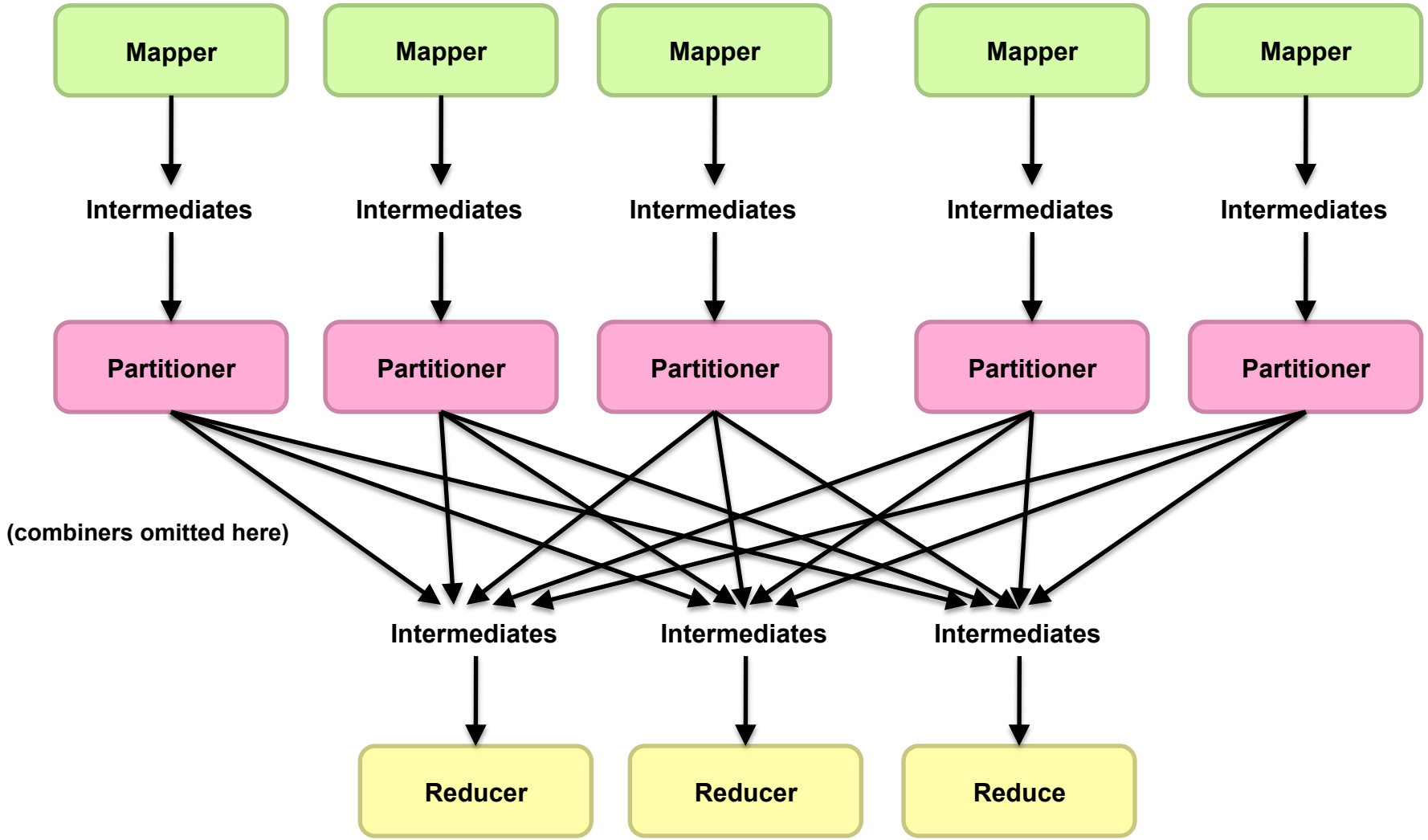
Anatomy of a Job

- Behind the scenes:
  - Input splits are computed (on client end)
  - Job data (jar, configuration XML) are sent to JobTracker
  - JobTracker puts job data in shared location, enqueues tasks
  - TaskTrackers poll for tasks
  - Off to the races…

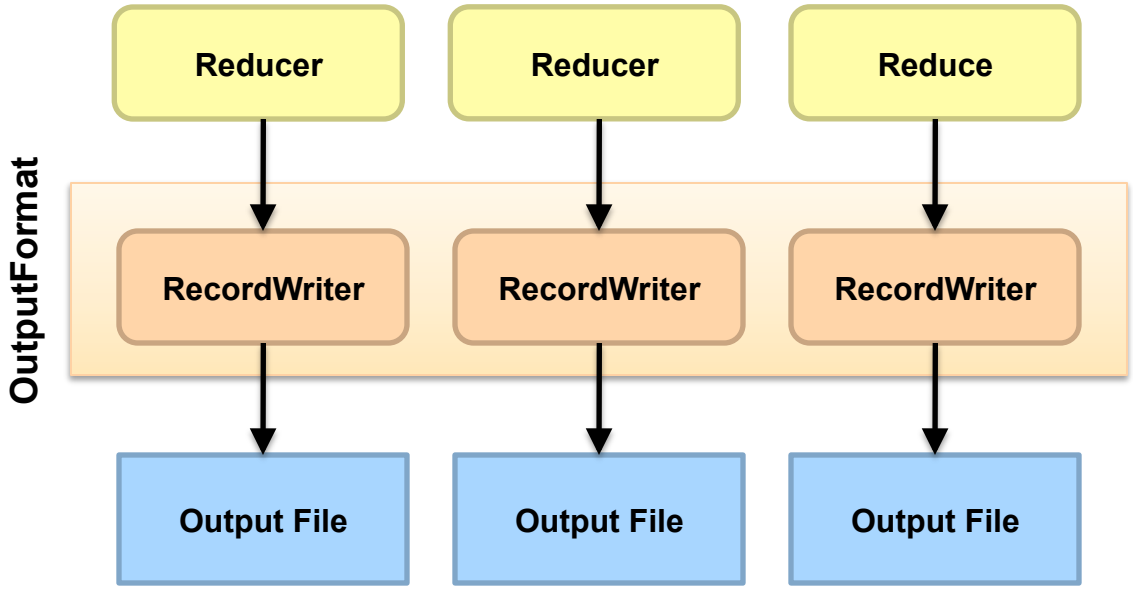


Source: redrawn from a slide by Cloduera, cc-licensed


Client
Records
InputSplit
InputSplit
InputSplit
RecordReader
RecordReader
RecordReader
Mapper
Mapper
Mapper



Source: redrawn from a slide by Cloduera, cc-licensed



Source: redrawn from a slide by Cloduera, cc-licensed

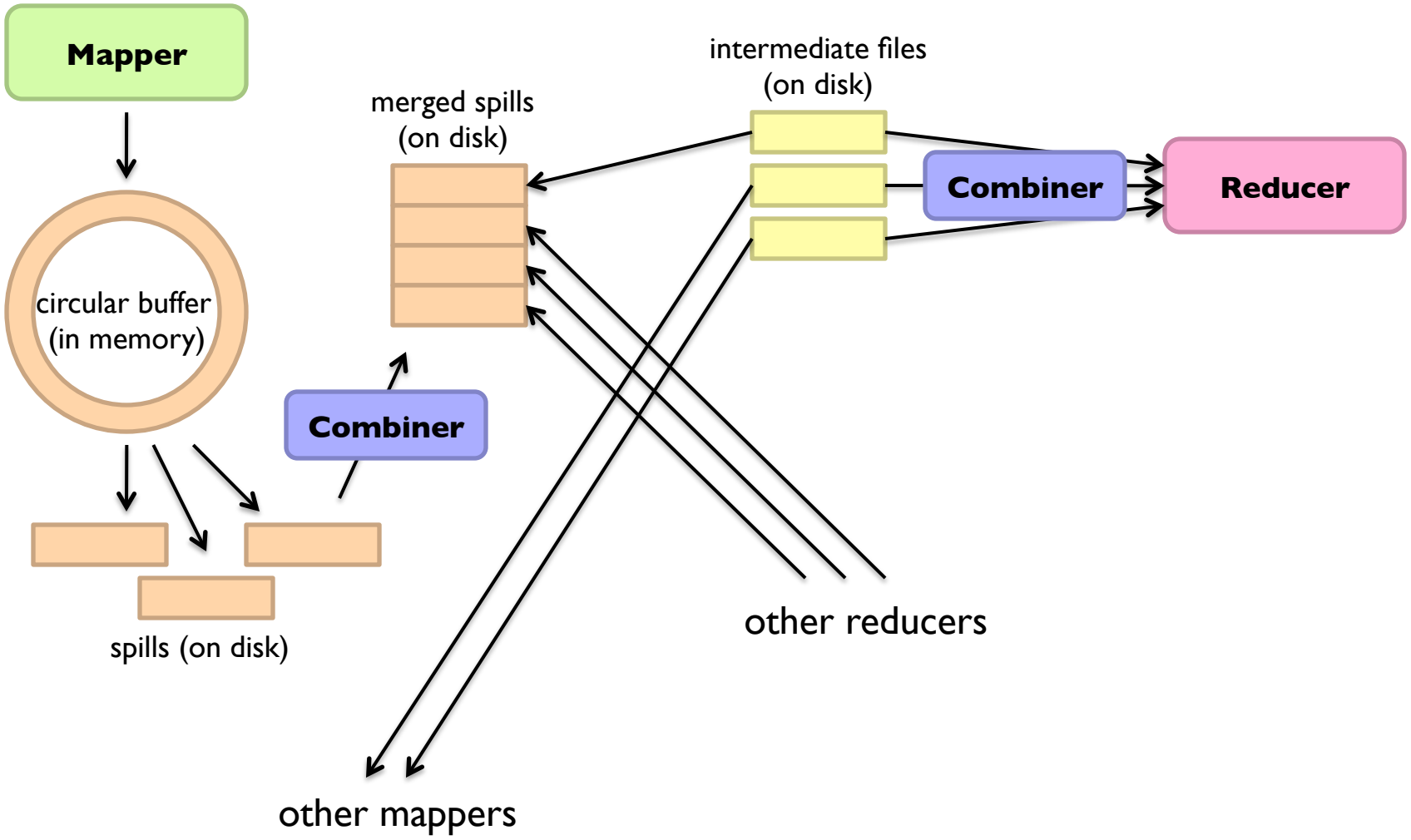
Input and Output

- InputFormat:
  - TextInputFormat
  - KeyValueTextInputFormat
  - SequenceFileInputFormat
  - …
- OutputFormat:
  - TextOutputFormat
  - SequenceFileOutputFormat
  - …

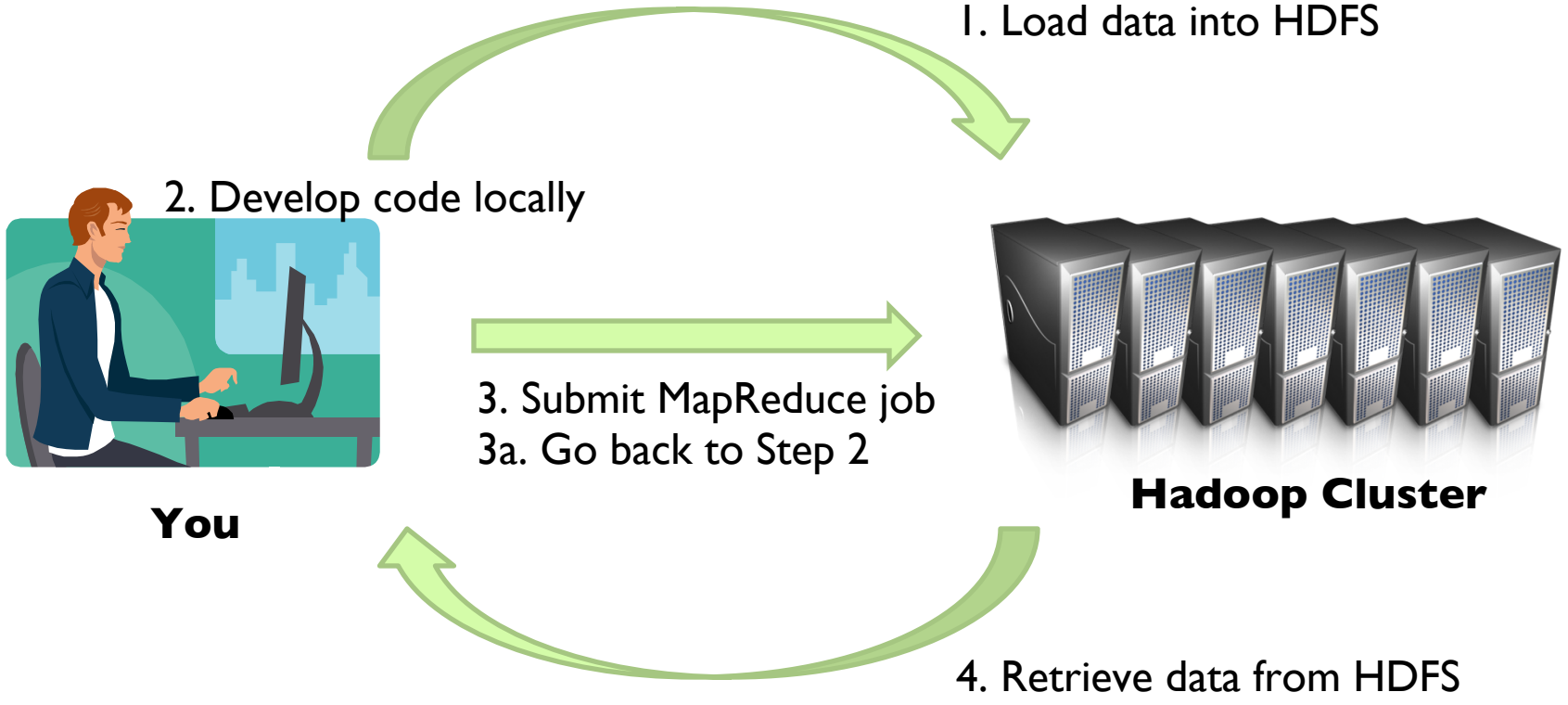
Shuffle and Sort in Hadoop

- Probably the most complex aspect of MapReduce
- Map side
  - Map outputs are buffered in memory in a circular buffer
  - When buffer reaches threshold, contents are "spilled" to disk
  - Spills merged in a single, partitioned file (sorted within each partition): combiner runs during the merges
- Reduce side
  - First, map outputs are copied over to reducer machine
  - "Sort" is a multi-pass merge of map outputs (happens in memory and on disk): combiner runs during the merges
  - Final merge pass goes directly into reducer

Shuffle and Sort



Hadoop Workflow



Recommended Workflow

- Here's how I work:
  - Develop code in Eclipse on host machine
  - Build distribution on host machine
  - Check out copy of code on VM
  - Copy (i.e., scp) jars over to VM (in same directory structure)
  - Run job on VM
  - Iterate
  - …
  - Commit code on host machine and push
  - Pull from inside VM, verify
- Avoid using the UI of the VM
  - Directly ssh into the VM

Debugging Hadoop

- First, take a deep breath
- Start small, start locally
- Build incrementally

Code Execution Environments

- Different ways to run code:
  - Plain Java
  - Local (standalone) mode
  - Pseudo-distributed mode
  - Fully-distributed mode
- Learn what's good for what

Hadoop Debugging Strategies

- Good ol' System.out.println
  - Learn to use the webapp to access logs
  - Logging preferred over System.out.println
  - Be careful how much you log!
- Fail on success
  - Throw RuntimeExceptions and capture state
- Programming is still programming
  - Use Hadoop as the "glue"
  - Implement core functionality outside mappers and reducers
  - Independently test (e.g., unit testing)
  - Compose (tested) components in mappers and reducers

Source: Wikipedia (Japanese rock garden)