Infrastructures for big data

Rasmus Pagh

Today's lecture

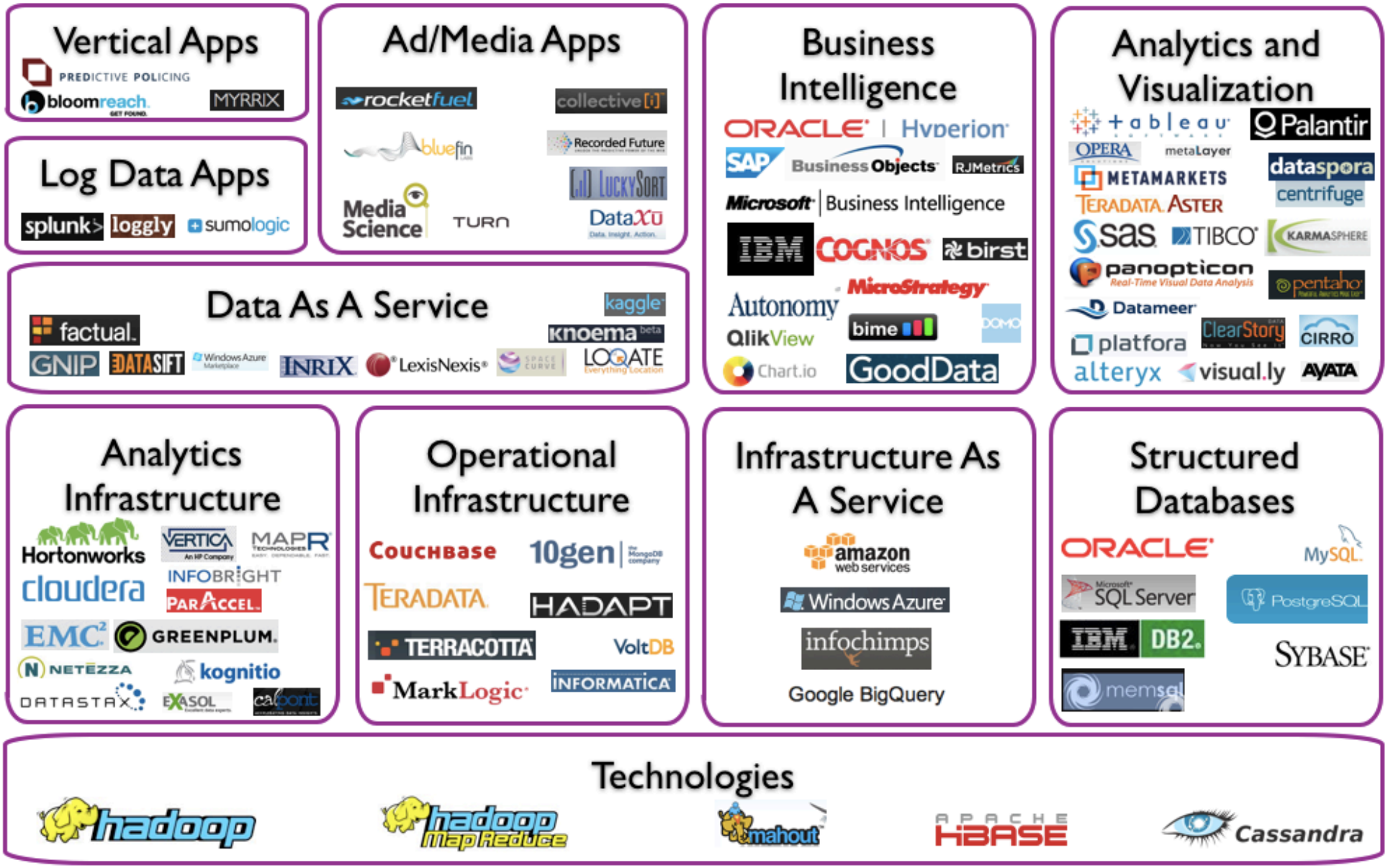
Three technologies for handling big data:

- MapReduce (Hadoop)
- BigTable (and descendants)
- Data stream algorithms

Alternatives to (some uses of) DBMSs.

Part of larger trend → **N**ot **O**nly **SQL**.

Big Data Landscape



dave@vcdave.com

blogs.forbes.com/davefeinleib

NO-SQL

- Silly to indentify technologies with what they are not.
- Better: **N**ot **O**nly **SQL**.

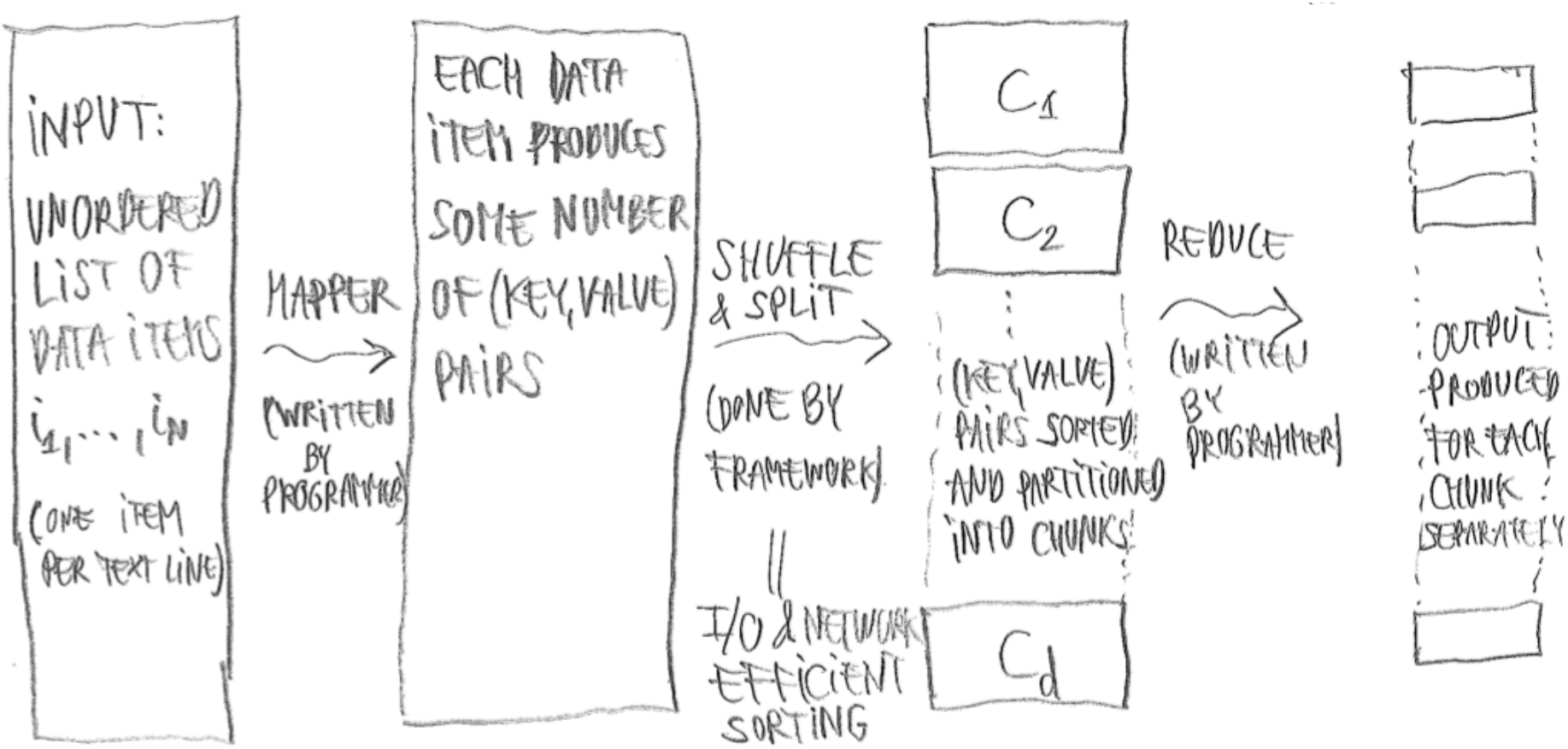
But what is it?

- Lemire: Programmer's revolt against database administrators.
- Common reason: Independence from very expensive large DBMSs.

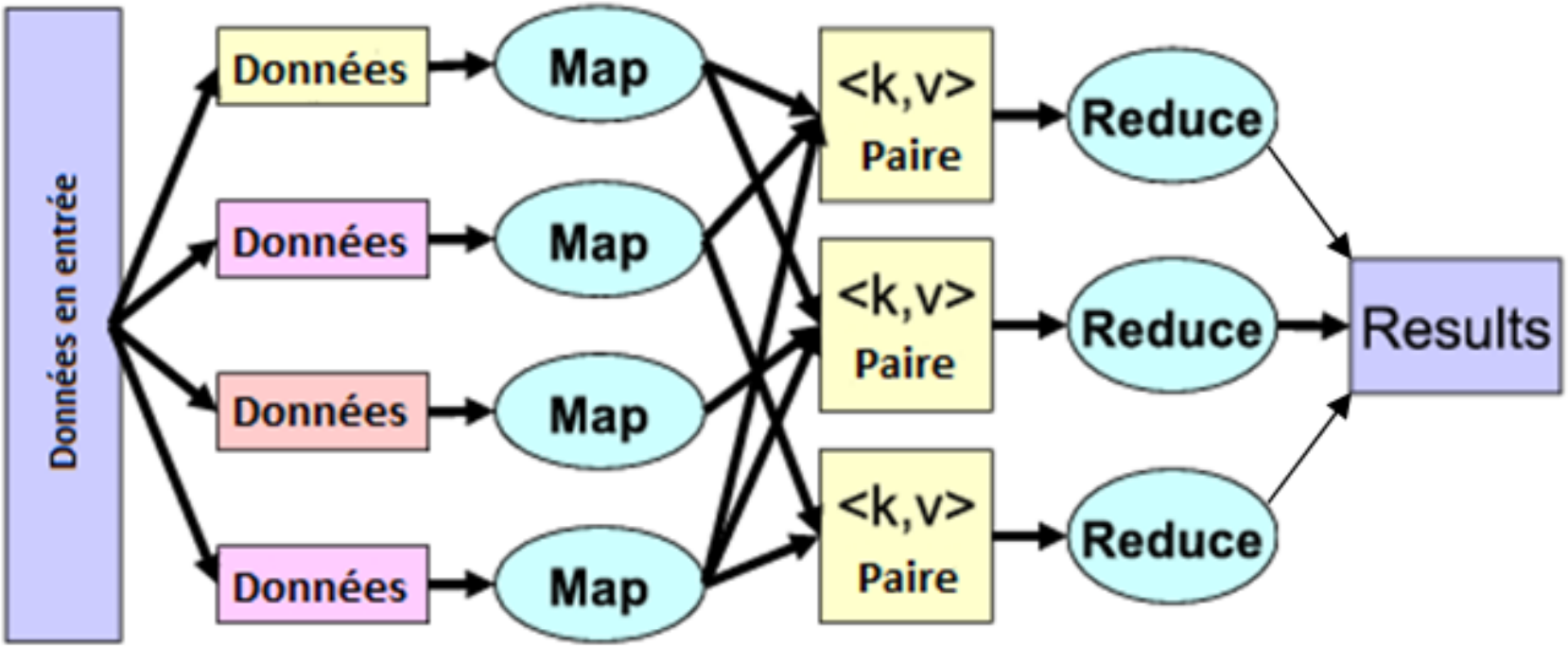
MapReduce

- Google system for distributed queries on line-based data.
- Runs on a **cluster** of networked machines (can be 1000s).
- Open source version: Hadoop
- Builds on distributed file system.
- Does not deal with transactions.

Mapreduce programming model



MapReduce execution



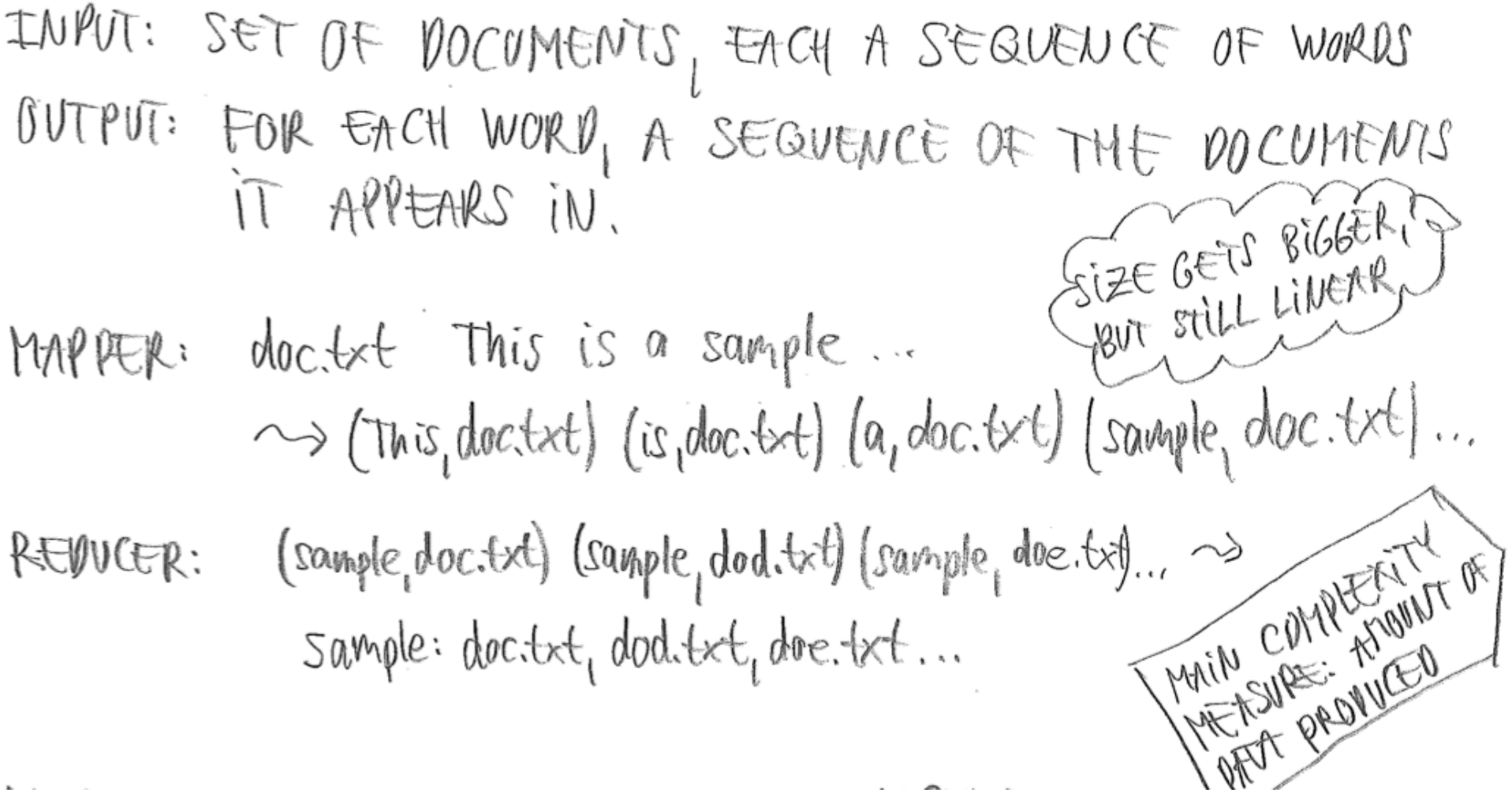
MapReduce examples

1. **Word count**
   Mapper: Transform text lines into pairs (w,1).
   Reducer: Add the occurrences of each word.

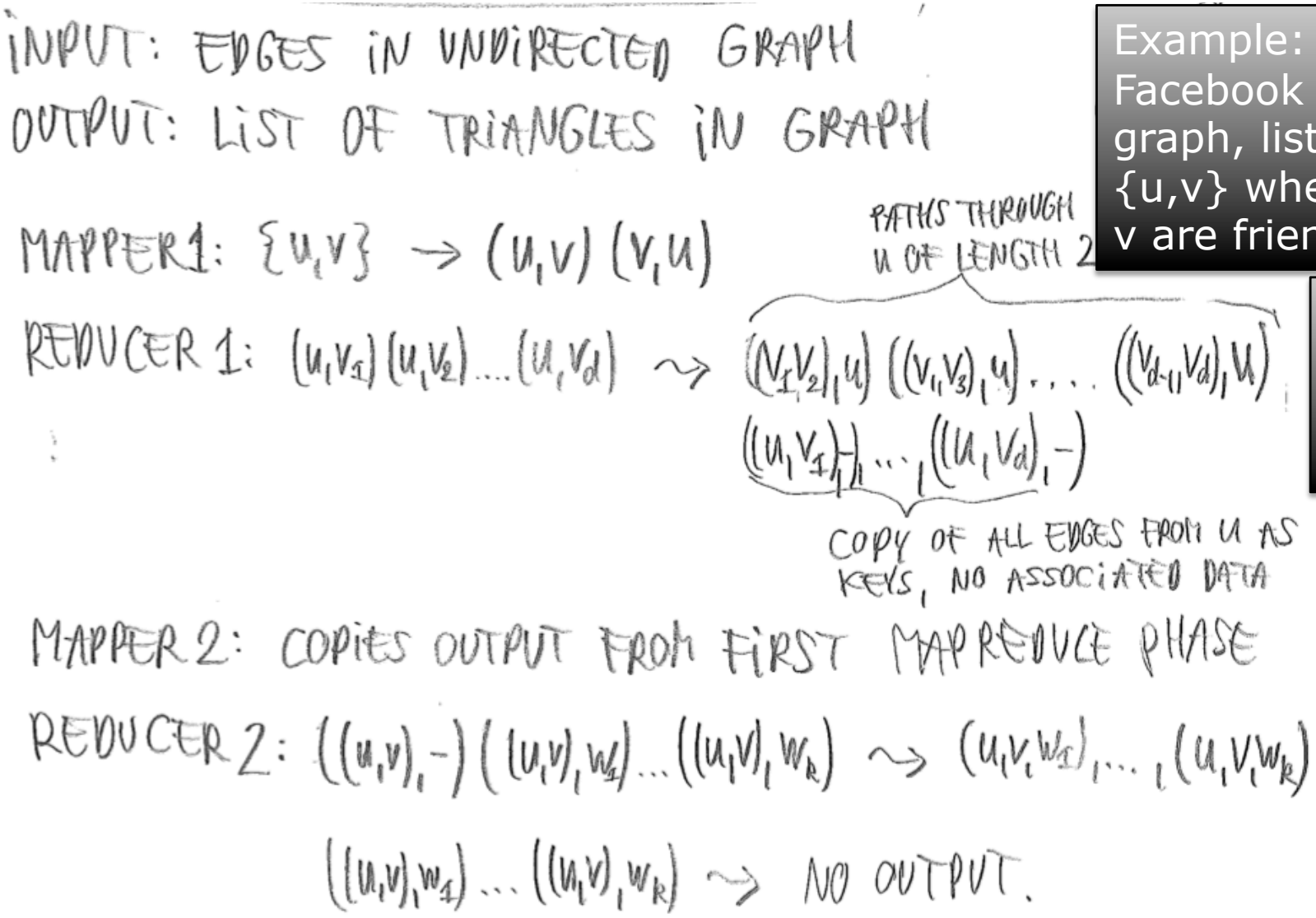
2. **SQL aggregation**
   Can implement queries of the form:

```
SELECT myReducer
FROM myMapper(R)
GROUP BY key
```

Example: Inverted lists



Example: Listing triangles



Example: Facebook friend graph, lists all pairs {u,v} where u and v are friends.

Triangle: Three mutual friends

Problem session

Suppose you want to compute a join of $R_1(a,b)$ and $R_2(b,c)$, on attribute b.

1. How could the input be represented to fit the MapReduce framework?

2. How can the join be computed?
   - Specify a mapper
   - Specify a reducer

BigTable

- Google system for storing and accessing data persistently in a distributed system.
- Highly scalable on clusters of cheap machines - add machines to scale up.
- Highly fault tolerant (replication).
- Like other distributed storage systems offers relaxed consistency compared to a DBMS ("eventual consistency").

Bigtable in a nutshell

- Small subset of DBMS functionality (”meet 7 out of 8 demands”).
- Data model generalizes relational one:
  `(column:string,rowId:string,time:int)` → `string`
  Stored sorted lexicographically by key.
- Only simple queries and transactions:
  - Lookup string using rowId and column.
  - Transaction: Modify a single row.

More BigTable

- Many similar systems have followed (distributed hash tables, Cassandra, DynamoDB, Hbase…).

- The course page links to a nice presentation by Jeff Dean, one the the system's main engineers.
  - The first 17 minutes give a good overview.
  - The rest is technical details outside the scope of this class.

Data streams

- In some applications (networks, sensors) data is produces so fast that normal techniques cannot keep up.
- Data model: Stream of data items
  - e.g. tuples, numbers, graph edges,…
- Processing model:
  - One pass over data (cannot go back).
  - Memory only large enough to store tiny part of data.
  - Instead, we store a “sketch” or “summary” that encodes enough information.

Things easy on a stream

- Count the number of data items.
- Compute aggregates of numbers
  - Sum
  - Average
  - Maximum, minimum, top-k
  - Variance
- Select tuples satisfying a condition.
- Select sample with 1% of data items.
- Split into several streams.

- More?

Primitive: Heavy hitters

- Find frequency of each data item with error at most f.
  - Can say "frequency zero" if frequency < f.
  - Space usage is around 1/f.
  - Extension to allow a "weight" for each item.

- Example:
  - Stream consists of (country,amount) pairs.
  - Want to report the countries that account for a fraction f of the total amount.

Primitive: Distinct elements

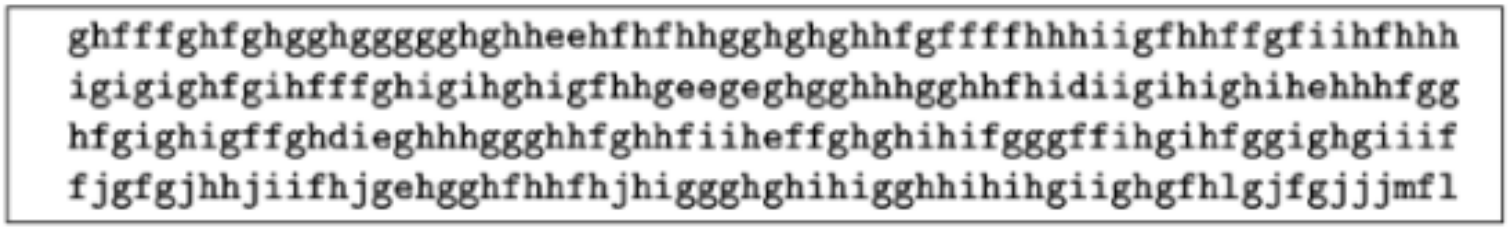
- Number of distinct elements in stream?
  - Answer should be correct within k% error.
  - Amazing: Space usage does not depend on the length of the stream!
- Examples:
  - Estimating result size in a DBMS.
  - 

FIGURE 1. The LOGLOG Algorithm with $m = 256$ condenses the whole of Shakespeare's works to a table of 256 "small bytes" of 4 bits each. The estimate of the number of distinct words in this run is $n^\circ = 30897$ (the true answer is $n = 28239$), which represents a relative error of $+9.4\%$.

Primitive: Euclidian distance

- Given two data streams of numbers, how similar are they?
  - View each stream as a point in n-dimensional space.
  - Want to know the distance between points, again with k% error allowed.

- Possible in space that depends only on the precision k and log(n)
  - For large n, log n is a constant in practice.

Data stream differences

- Many data stream algorithms lets you look at the **difference** of two streams.

- Examples:
  - Which items have significantly higher frequency in s1 than in s2?
  - If each item occurs only once in a stream: How many items were in s1 but not in s2?

- Application: Anomaly detection.

Putting primitives together

- Can build data stream algorithms by combining or pipelining primitives.

- Examples:
  - Number of distinct customers per shop (split stream, distinct elements).
  - Average number of times an item occurs (count total length, distinct elements).
  - Countries where the sales have increased by > 1 million compared to last month (store last month's sketch, take diff).

Where are the stream systems?

- Prototype systems such as Stanford Stream Data Manager (CQL)
- Gigascope: System used at AT&T.
- "Real-time" databases may work on data streams, but are typically update-optimized DBMSs.

Course goals

- After the course the students should be able to:
  - …
  - formulate an analytics task as a sequence of operations on a stream, or in the MapReduce framework.

Next steps

- This afternoon: Last exercise session
  - Exam-type exercise.
- Midnight: Deadline for hand-in 4, and re-submission of hand-in 3.

- Next week:
  - I encourage you to sit down and do problems 1, 3, 4, and 5 of the exam from January 2012. This should take (at most) about 3 hours.
  - At the lecture I will go through the exam and how it is graded.