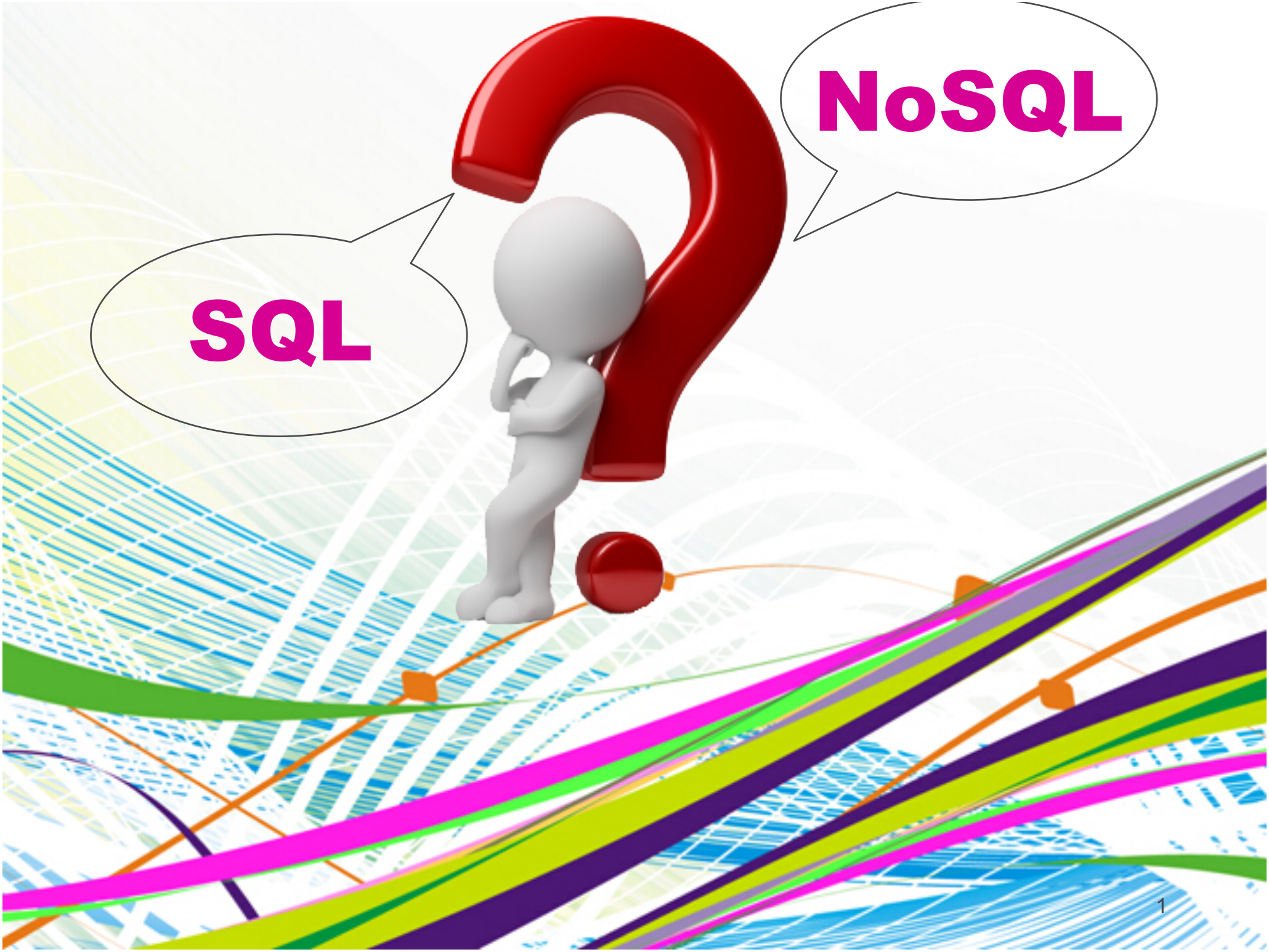

SQL
NoSQL

NoSQL: The Name

- "SQL" = Traditional relational DBMS
- Recognition over past decade or so:
  Not every data management/analysis problem is best solved using a traditional relational DBMS
- "NoSQL" = "No SQL" =
  Not using traditional relational DBMS
- "No SQL" ≠ Don't use SQL language

NoSQL: The Name

- "SQL" = Traditional relational DBMS
- Recognition over past decade or so:
  Not every data management/analysis problem is best solved using a traditional relational DBMS
- "NoSQL" = "No SQL" =
  Not using traditional relational DBMS
- "No SQL" ≠ Don't use SQL language

✳ "NoSQL" = "Not Only SQL"

Not Only SQL

Not every data management/analysis problem is best solved using a traditional DBMS

Database Management System (DBMS) provides….

… efficient, reliable, convenient, and safe multi-user storage of and access to massive amounts of persistent data.

NoSQL Systems

Alternative to traditional relational DBMS

+ Flexible schema
+ Quicker/cheaper to set up
+ Massive scalability
+ Relaxed consistency → higher performance & availability

– No declarative query language → more programming
– Relaxed consistency → fewer guarantees

Example #1: Web log analysis

Each record: UserID, URL, timestamp, additional-info

Task: Find all pairs of UserIDs accessing same URL

Example #1: Web log analysis

Each record: UserID, URL, timestamp, additional-info

Separate records: UserID, name, age, gender, …

Task: Find average age of user accessing given URL

Example #2: Social-network graph

Each record: UserID_1, UserID_2

Separate records: UserID, name, age, gender, …

Task: Find all friends of given user

Example #2: Social-network graph

Each record: UserID_1, UserID_2

Separate records: UserID, name, age, gender, …

Task: Find all friends of friends given user

Example #2: Social-network graph

Each record: UserID_1, UserID_2

Separate records: UserID, name, age, gender, …

Task: Find all women friends of men friends of given user

Example #2: Social-network graph

Each record: $UserID_1$, $UserID_2$

Separate records: UserID, name, age, gender, …

Task: Find all friends of friends of friends of … friends of given user

Example #3: Wikipedia pages

Large collection of documents

Combination of structured and unstructured data

Task: Retrieve area of all counties in square meter

NoSQL Systems

Alternative to traditional relational DBMS

+ Flexible schema
+ Quicker/cheaper to set up
+ Massive scalability
+ Relaxed consistency → higher performance & availability

– No declarative query language → more programming
– Relaxed consistency → fewer guarantees

NoSQL Systems

Several realization

NoSQL Systems

Several realization

- MapReduce framework
- Key-value stores
- Document stores
- Graph database systems

MapReduce Framework

Originally from Google, open source Hadoop

- No data model, data stored in files
- User provides specific functions

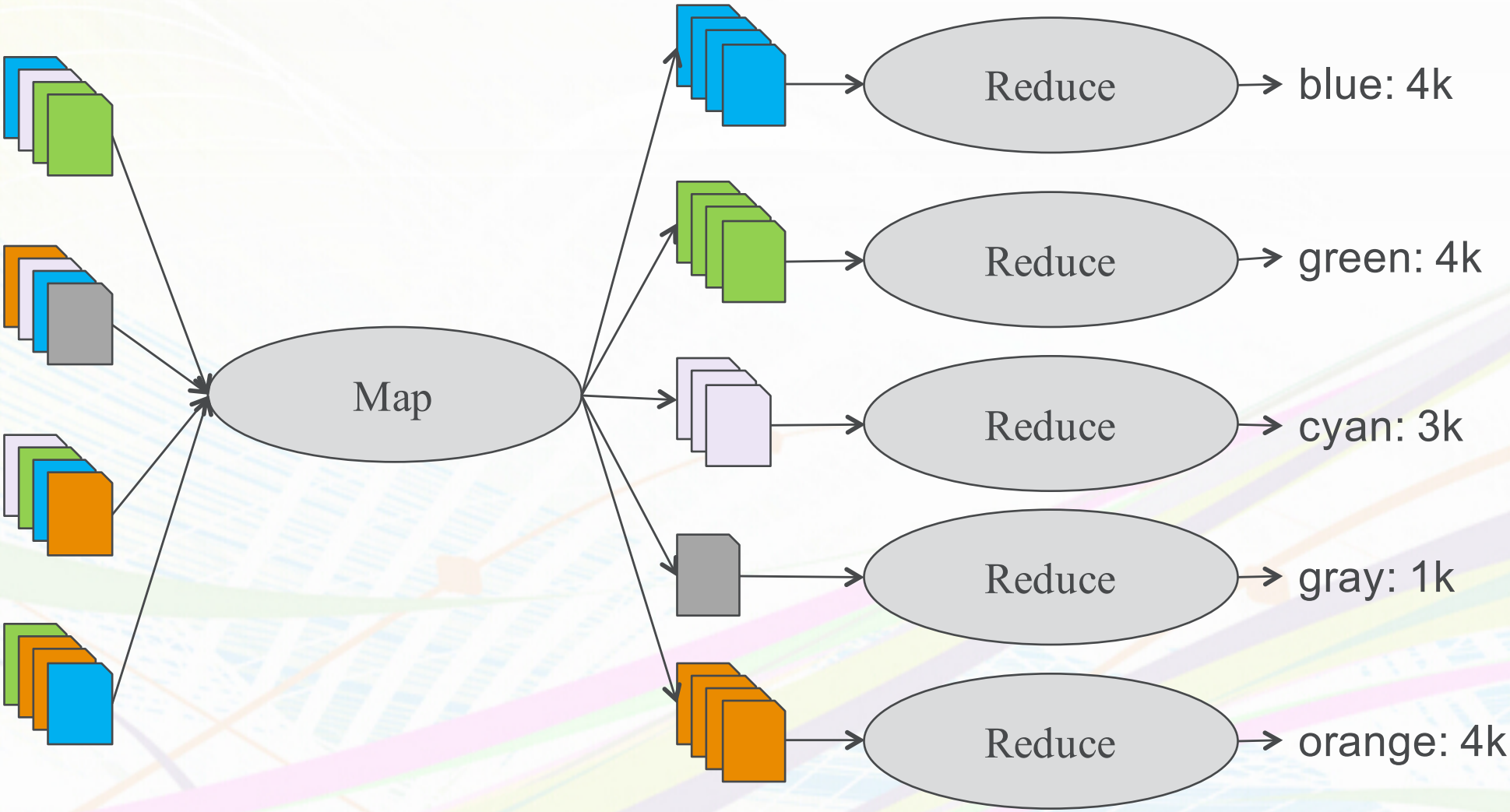
Map and Reduce Functions

Map: Divide problem into subproblems

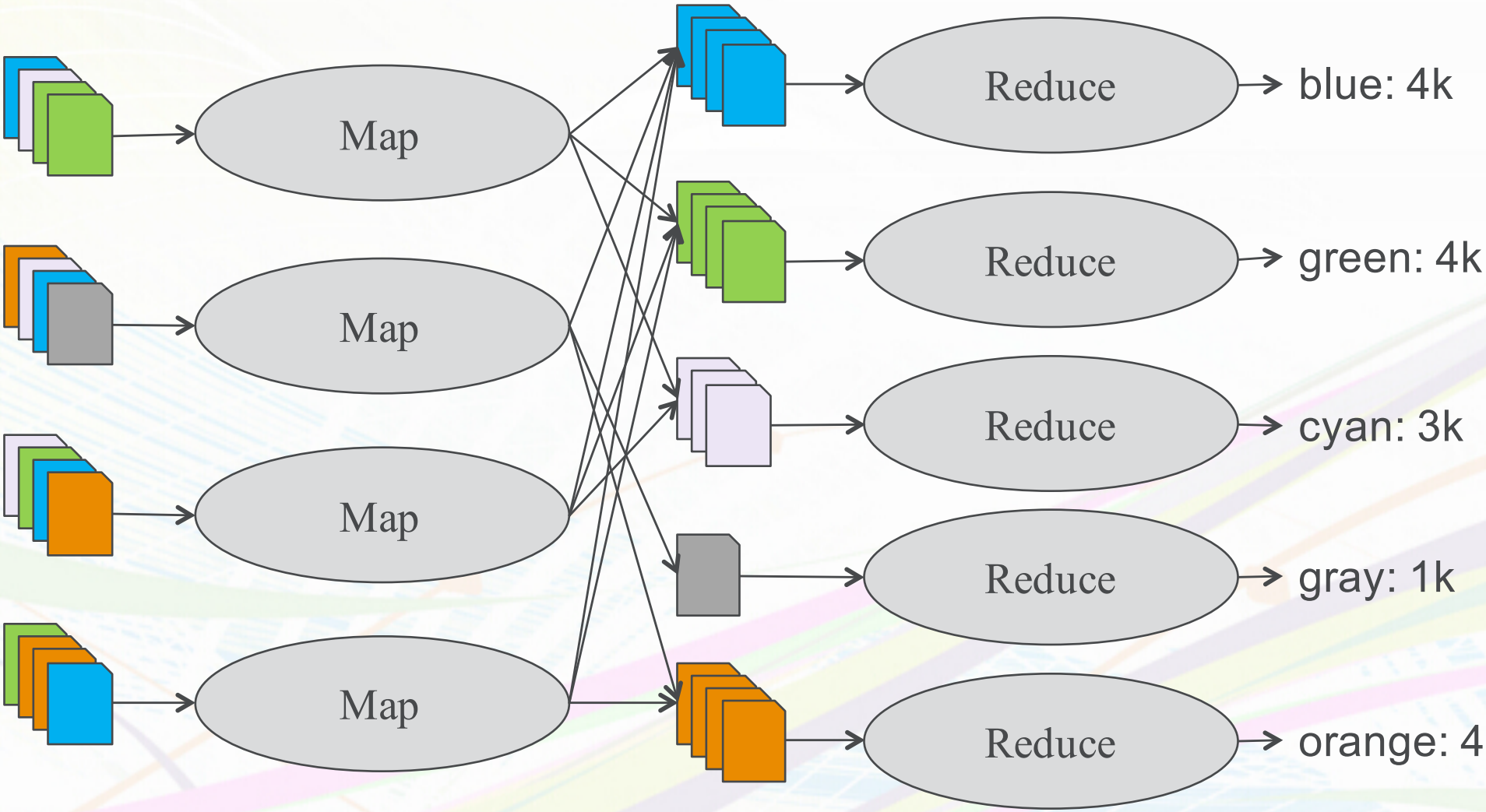
Reduce: Do work on subproblems, combine results

- map: takes (item_key, value), produces one or more (key, value') pairs
- reduce: takes (key, {set of value'}), produces one or more output results
  - typically (key, agg_value)

MapReduce


Map
Reduce
Reduce
Reduce
Reduce
Reduce
blue: 4k
green: 4k
cyan: 3k
gray: 1k
orange: 4k

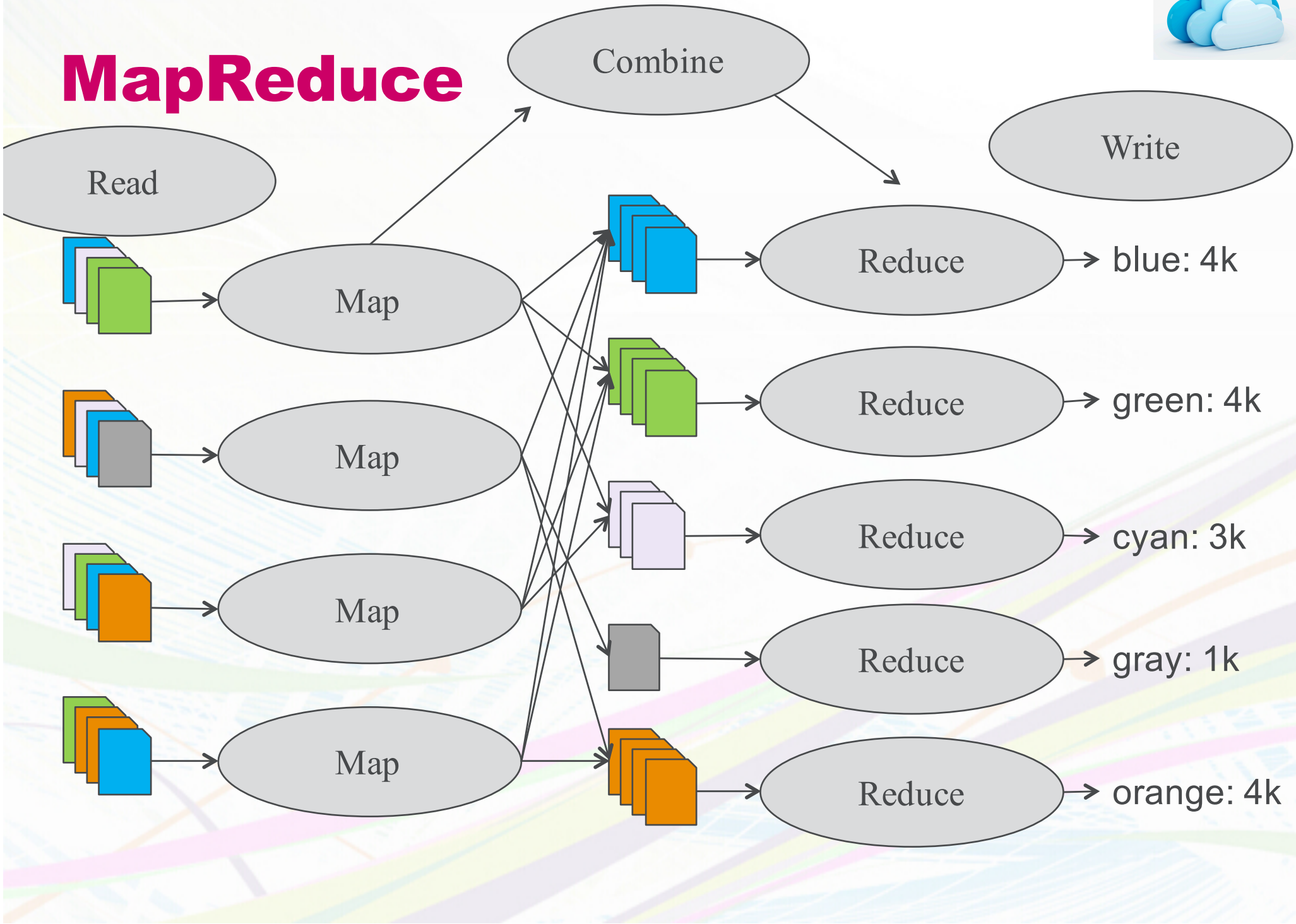
MapReduce


Map
Map
Map
Map
Reduce
Reduce
Reduce
Reduce
Reduce
blue: 4k
green: 4k
cyan: 3k
gray: 1k
orange: 4k

MapReduce

Combine
Map
Map
Map
Map
Reduce
Reduce
Reduce
Reduce
Reduce
blue: 4k
green: 4k
cyan: 3k
gray: 1k
orange: 4k

MapReduce


Read
Combine
Write
Map
Map
Map
Map
Reduce
Reduce
Reduce
Reduce
Reduce
blue: 4k
green: 4k
cyan: 3k
gray: 1k
orange: 4k

Simple example: Word count

```
map(String key, String value) {
  // key: document name, line no
  // value: contents of line
  for each word w in value:
    emit(w, "1")
}
```

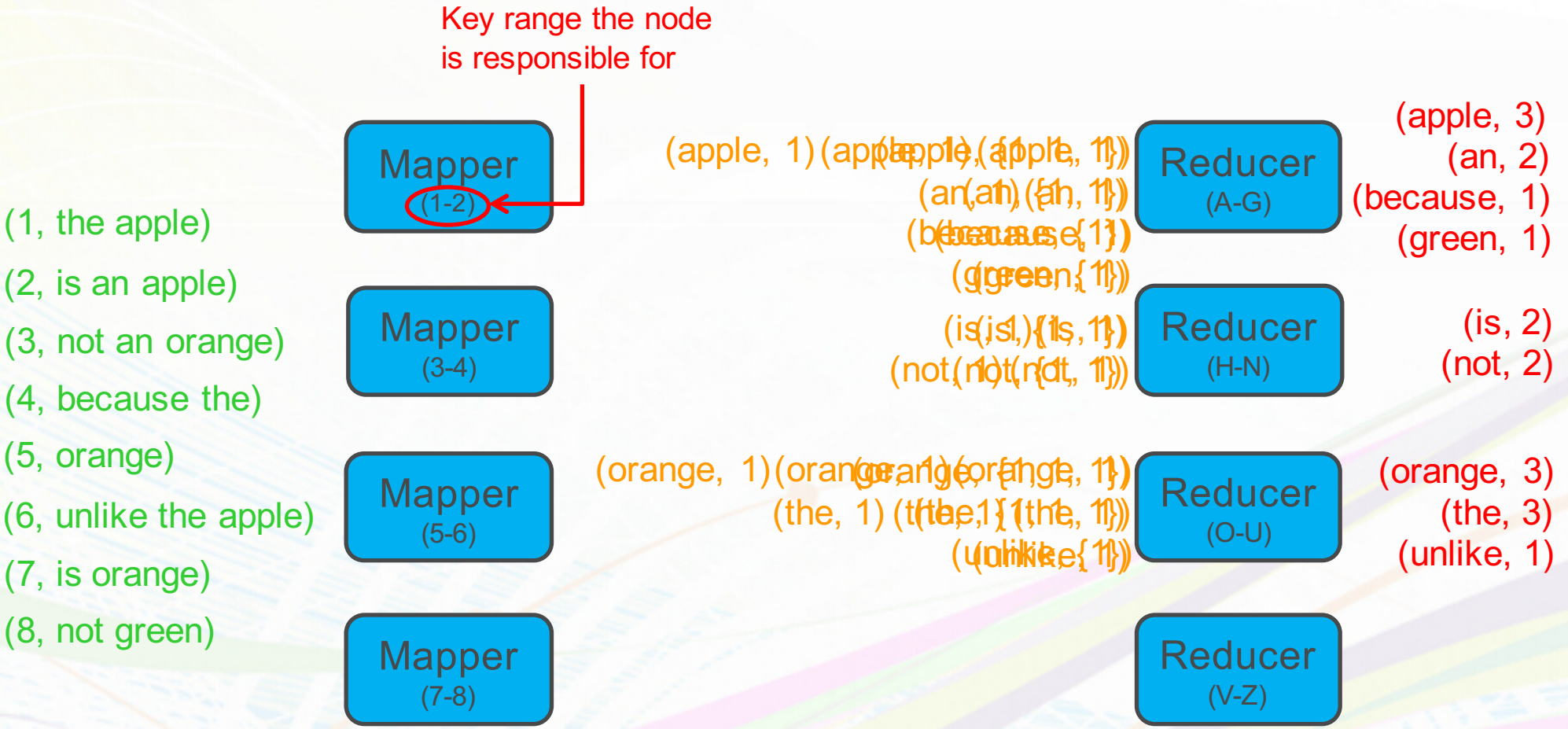
```
reduce(String key, Iterator values) {
  // key: a word
  // values: a list of counts
  int result = 0;
  for each v in values:
    result += ParseInt(v);
  emit(key, result)
}
```

Goal: Given a set of documents, count how often each word occurs

Input: Key-value pairs (document:lineNumber, text)

Output: Key-value pairs (word, #occurrences)

Simple example: Word count



MapReduce Framework

- No data model, data stored in files
- User provides specific functions
- System provides data processing “glue”, fault tolerance, scalability

MapReduce Framework

Schemas and declarative queries are missed

Hive – schemas, SQL-like query language

Pig – more imperative but with relational operators

- Both compile to "workflow" of Hadoop (MapReduce) jobs

Dryad allows user to specify workflow

- Also DryadLINQ language

Key-Value Stores

Extremely simple interface

- Data model: (key, value) pairs
- Operations: Insert(key,value), Fetch(key), Update(key), Delete(key)

Implementation: efficiency, scalability, fault-tolerance

- Records distributed to nodes based on key
- Replication
- Single-record transactions, “eventual consistency”

Key-Value Stores

Extremely simple interface

- Data model: (key, value) pairs
- Operations: Insert(key,value), Fetch(key), Update(key), Delete(key)

Example systems

- Google BigTable, Amazon Dynamo, Cassandra, HBase, …

Document Stores

Like Key-Value Stores except value is document

- Data model: (key, document) pairs
- Document: JSON, XML, other semistructured formats
- Basic operations: Insert(key,document), Fetch(key), Update(key), Delete(key)
- Also Fetch based on document contents

Example systems

- MongoDB, CouchDB, SimpleDB, …

Graph Database Systems

- Data model: nodes and edges
- Nodes may have properties (including ID)
- Edges may have labels or roles

Graph Database Systems

- Interfaces and query languages vary
- Single-step versus “path expressions” versus full recursion
- Example systems

  Neo4j, Pregel, …
- RDF “triple stores” can map to graph databases

NoSQL Systems

- "NoSQL" = "Not Only SQL"

  Not every data management/analysis problem
  is best solved *exclusively* using a traditional DBMS

- Current incarnations
  - MapReduce framework
  - Key-value stores
  - Document stores
  - Graph database systems

Not Only SQL