CSE 344 Section 7

1. For each of the following statements, identify whether it is true or false and explain why.

   a. The main reason NoSQL database was introduced was because relational databases did not scale up to a very large number of servers.

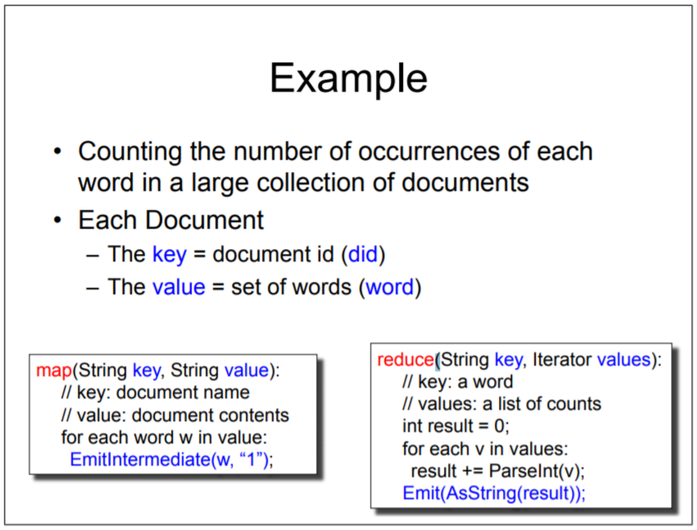
   b. A Resilient Distributed Dataset (RDD) is another term for a distributed file on disks.

   c. Hadoop MapReduce is generally faster than Spark at evaluating queries.

   d. If you can compute a query in parallel on 1000 servers in 15 minutes, then you can always compute it in parallel on 500 servers in at most 30 minutes.

   e. If you can compute a query in parallel on 500 servers in 30 minutes, then you can always compute it in parallel on 1000 servers in 15 minutes.

The following slide gives an example of the map and reduce functions to group by and aggregate on count:



Example

- Counting the number of occurrences of each word in a large collection of documents
- Each Document
  - The key = document id (did)
  - The value = set of words (word)

```
map(String key, String value):
  // key: document name
  // value: document contents
  for each word w in value:
    EmitIntermediate(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(AsString(result));
```

Use pseudo-code to write the map and reduce functions.

2. Convert the following SQL queries to MapReduce:

   a.

```
Students(name, major, credits)

select s.name, max(s.credits)
from Students s
where s.major == “CSE”
group by s.name;

map(Tuple s):
     // write implementation here




// key is name, credits is list of values of credit from tuples which
// have the given name
reduce(String key, List<int> credits):
     // write implementation here
```

b.

```
Devices(model, company, height, width)

select model, avg(height * width)
from Devices
where company == “Microsoft”
and height <> width;

map(Tuple p):
   // write implementation here
```

```
// key is the model, values is tuples containing (height * width, 1)
reduce(String key, List<double, int> values):
   // write implementation here
```

c. (final, autumn-17)

```
Product(category, price, quantity)

select p.category, count(*)
from Product p
where p.price > 100
group by p.category
having sum(p.quantity) > 200;


map(Tuple p):
      // write implementation here
```

```
// key is the category, values is the list of quantities having that
// category
reduce(String key, List<int> quantities):
      // write implementation here
```