Pig

6.033 Quiz 1 Review

Qian Long

Pig Latin

- high level language on top of MapReduce
- combines SQL-like syntax with procedural programming
- logical plan
- compilation to MapReduce jobs

Nested Data Model

- **atom** - single atomic value
  - ‘alice’, 3
- **tuple** - sequence of fields of any type
  - (‘alice’, ‘lakers’), (‘field1’, { (‘blah1’), (‘blah2’)}, ‘field3’)
- **bag** - collection of tuples, allows duplicates
  - { (‘alice’, ‘lakers’), (‘alice’, (‘ipod’, ‘apple’)) }
- **map** - key/value pairs
  - key must be atom, value can be any type (can be different)
  - [‘key1’ -> { (‘field1’, ‘field2’)}, ‘key2’ -> 20]

Common Operations

- LOAD
  - read data
- FOREACH … GENERATE
- FILTER
- (CO) GROUP
  - group by matching field
- FLATTEN
- JOIN
  - (CO)GROUP followed by FLATTEN
- STORE
  - write computed output

More Operations

- UNION
- CROSS
- ORDER
- DISTINCT

FOREACH...GENERATE, FLATTEN

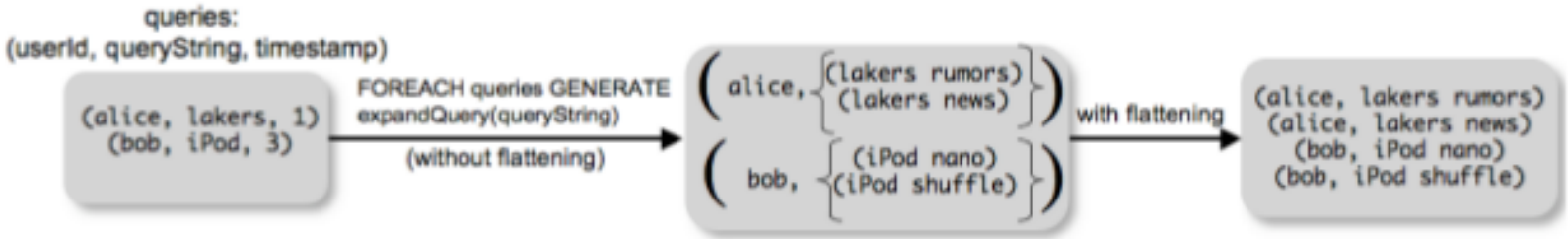

Figure 1: Example of *flattening* in FOREACH.

(CO)GROUP vs JOIN

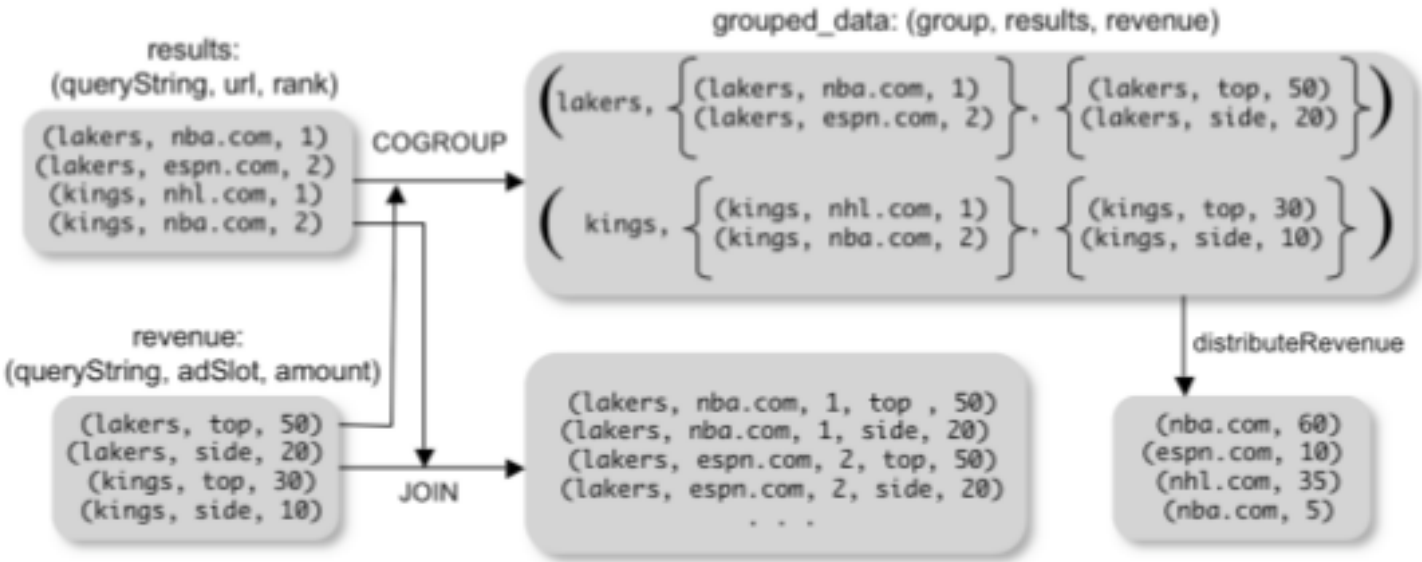

Figure 2: COGROUP versus JOIN.

Building Logical Plan

- Pig Interpreter
  - parses command, verifies inputs
  - constructs logical plan
- lazy execution
  - processing only happens on STORE command
  - allows re-ordering for optimization
  - in-memory pipelining

Compile to MapReduce Jobs

- converts each (CO)GROUP/JOIN into a map/reduce job
  - map/reduce boundary
  - map job assigns intermediate keys based on what you are grouping by
  - reduce job does the actual grouping based on the intermediate keys
- have 2 options for commands between subsequent (CO)GROUPs
  - (a) computed locally in reduce task of first (CO)GROUP
  - (b) computed in map task of second (CO)GROUP
  - Pig does (a) to reduce intermediate data between map/reduce jobs

Compilation Example

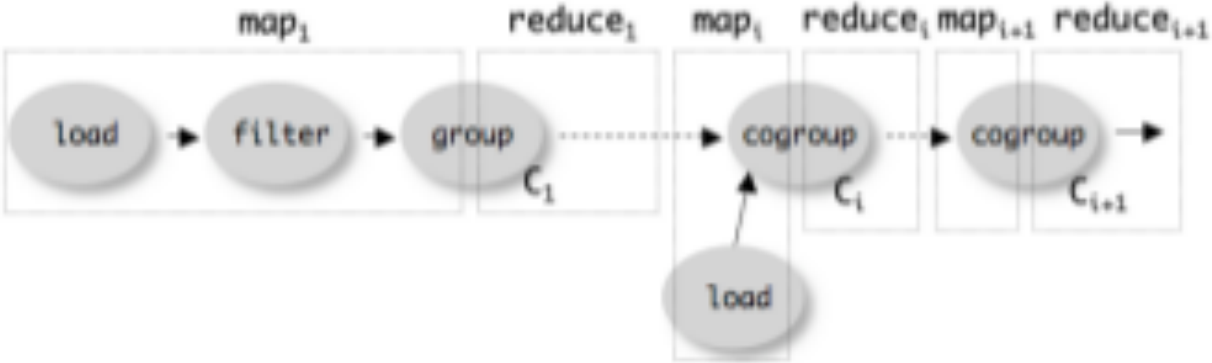

Figure 3: Map-reduce compilation of Pig Latin.

Sample Question

Suppose you have three Pig tables, papers, authors, and paperauths, defined as follows:

```
papers = load 'papers.dat' using PigStorage('\t') as (pid:int, pname:chararray );
authors = load 'authors.dat' using PigStorage('\t') as (aid:int, aname:chararray);
paperauths = load 'paperauths.dat' using PigStorage('\t') as (pa_pid:int, pa_aid:int);
```

Now, consider the following Pig program (note that this program doesn't actually use the author's table).

```
x = group paperauths by pa_pid;
x2 = foreach x generate flatten(paperauths), COUNT(paperauths) as count;
x3 = cogroup x2 by pa_pid, papers by pid;
x4 = foreach x3 generate flatten(papers), flatten(x2);
x5 = foreach x4 generate pname, count;

dump x5;
```

Here dump is a Pig command to print a result set

Sample Question

Q1: How many map/reduce jobs would this translate into?

Sample Question

A1: 2

Sample Question

Q2: Describe briefly what the first map job does.

Sample Question

A2: It reads the paperauths file, iterating through the records and emitting tuples of the form pid, (paper-auths tuple) (i.e., where the key is pid and the value is the tuple.

Sample Question

Q3: Describe briefly what the first reduce job does.

Sample Question

A3: For each tuple t in each
{ paperauth tuple } set S passed into the reduce job, emit t.pa_pid, COUNT(S).
We also accepted answers that did not do the COUNT in the reduce
(presumably deferring it to the next map job.)

Sample Questions

http://db.csail.mit.edu/6.830/assignments/q2-2012.pdf

http://db.csail.mit.edu/6.830/assignments/q2-solns-2012.pdf