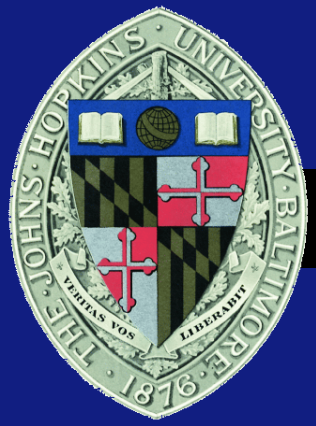
Other M/R Interfaces: PIG II

EN 600.420

Instructor: Randal Burns

17 April 2017



Department of Computer Science, *Johns Hopkins University*

PIG Language Constructs

- **FOREACH:** allows parallel processing (in a mapper) for all inputs in a data set
- **FILTER:** discard unwanted data (in either mapper or reducer)
- **GROUP/CO-GROUP:** put related data together using the shuffle process.

- **These constructs allow for database-style query optimization.**

PIG Data Model

- Atom: simple value
- Tuple: sequence of values
- Bag: multiset with duplicates
  - flexible schema for elements

$$\left\{ \begin{array}{c} \text{('alice', 'lakers')} \\ \text{('alice', ('iPod', 'apple'))} \end{array} \right\}$$

- Map: key/value data structure
  - Keys must be atoms for efficiency

$$\left[\begin{array}{c} \text{'fan of'} \rightarrow \left\{ \begin{array}{c} \text{('lakers')} \\ \text{('iPod')} \end{array} \right\} \\ \text{'age'} \rightarrow 20 \end{array} \right]$$

PIG Expressions

- Simple set that have to be parallelizable

$$t = \left(\text{'alice'}, \left\{ \begin{array}{c} (\text{'lakers'}, 1) \\ (\text{'iPod'}, 2) \end{array} \right\}, [\text{'age'} \rightarrow 20]\right)$$

Let fields of tuple `t` be called `f1, f2, f3`

Expression Type	Example	Value for t
Constant	'bob'	Independent of t
Field by position	$0	'alice'
Field by name	f3	['age' → 20]
Projection	f2.$0	{('lakers') ('iPod')}
Map Lookup	f3#'age'	20
Function Evaluation	SUM(f2.$1)	1 + 2 = 3
Conditional Expression	f3#'age'>18? 'adult':'minor'	'adult'
Flattening	FLATTEN(f2)	'lakers', 1 'iPod', 2

Table 1: Expressions in Pig Latin.

Bags and Co-Groupings

- Pig programming uses the patten of co-grouping data, applying aggregates, and then flattening the results
  - Allows SQL like functionality in sequenced programming
  - It's not super-intuitive (see the paper)

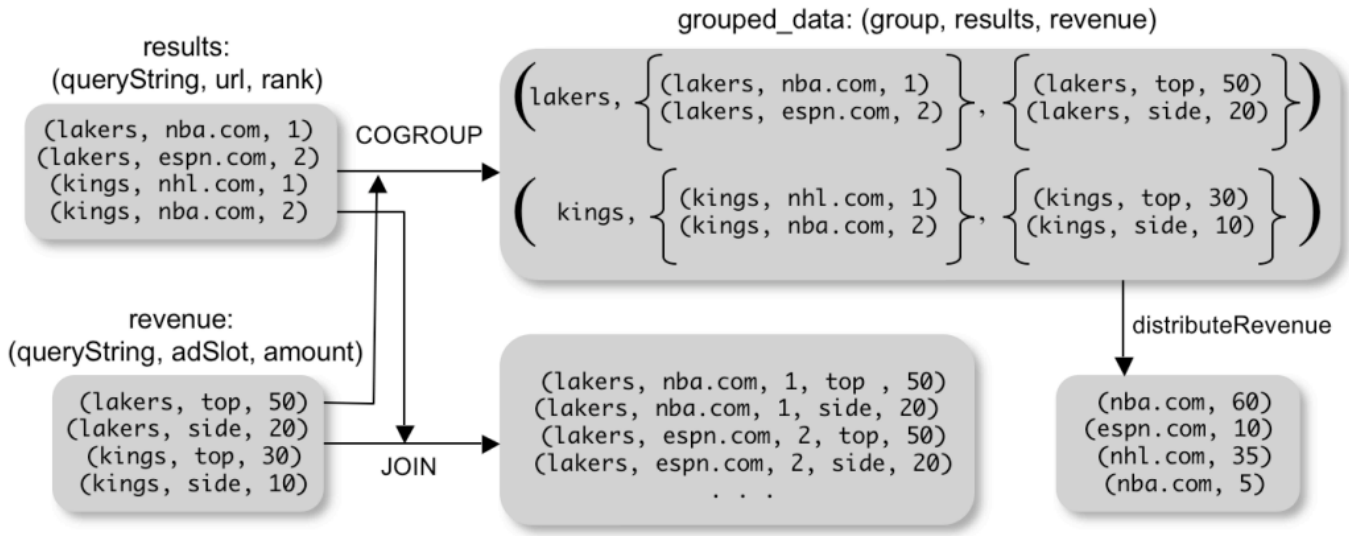


Figure 2: COGROUP versus JOIN.

Compiliing to MR

- Each PIG program compiles to several MR programs and is run in Hadoop!

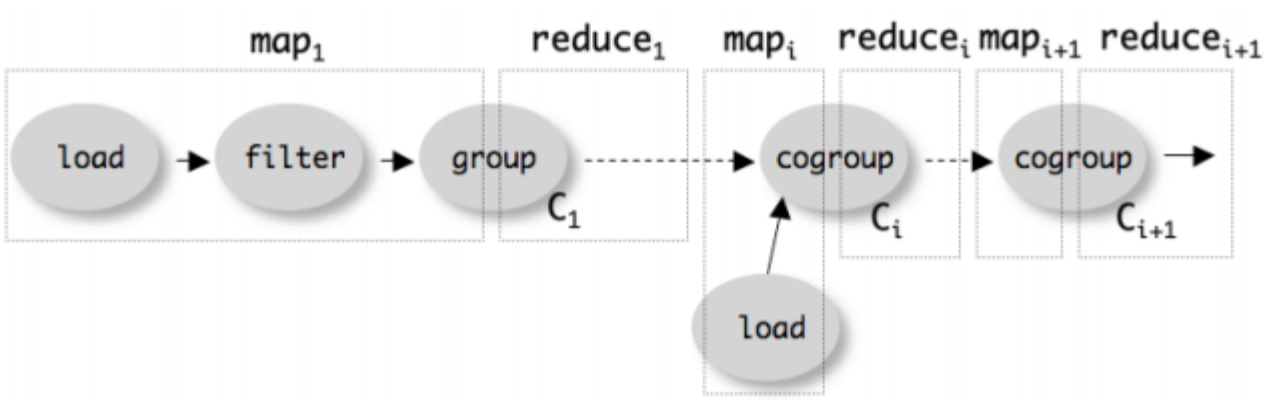


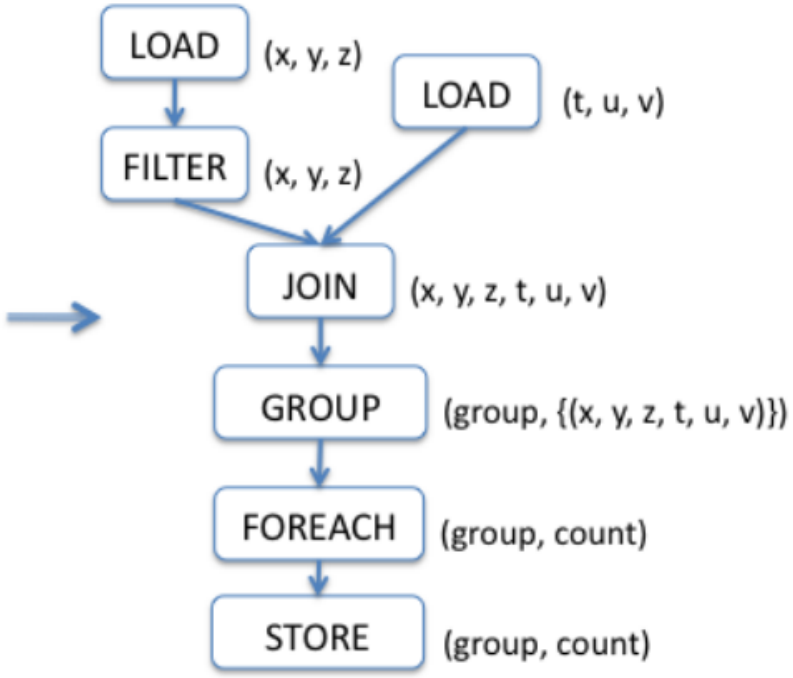
Figure 3: Map-reduce compilation of Pig Latin.

Compiling to MR (ii)

- From: Gates *et al. Building a High-Level Dataflow System on top of Map-Reduce: The Pig Experience, VLDB 2009.*

Pig Latin

```
A = LOAD 'file1' AS (x, y, z);
B = LOAD 'file2' AS (t, u, v);
C = FILTER A by y > 0;
D = JOIN C BY x, B BY u;
E = GROUP D BY z;
F = FOREACH E GENERATE
      group, COUNT(D);
STORE F INTO 'output';
```



Compiling to MR (ii)

- From: Gates *et al. Building a High-Level Dataflow System on top of Map-Reduce: The Pig Experience, VLDB 2009.*