Other M/R Interfaces: PIG II

EN 600.420

Instructor: Randal Burns

16 April 2018

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

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

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

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

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)

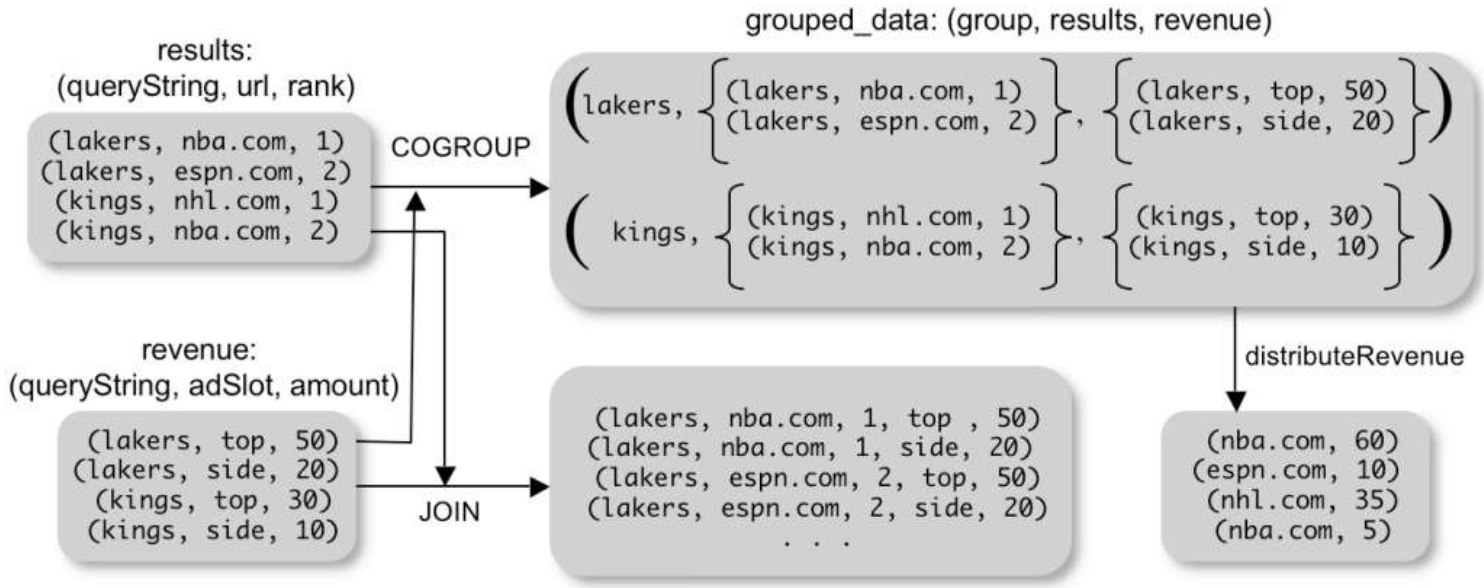


Figure 2: COGROUP versus JOIN.

Compiliing to MR

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

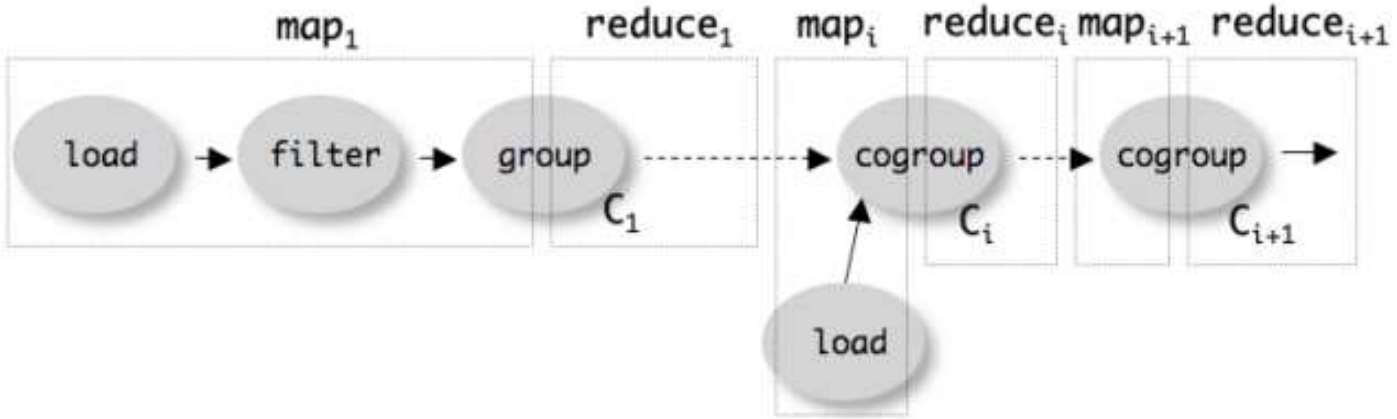
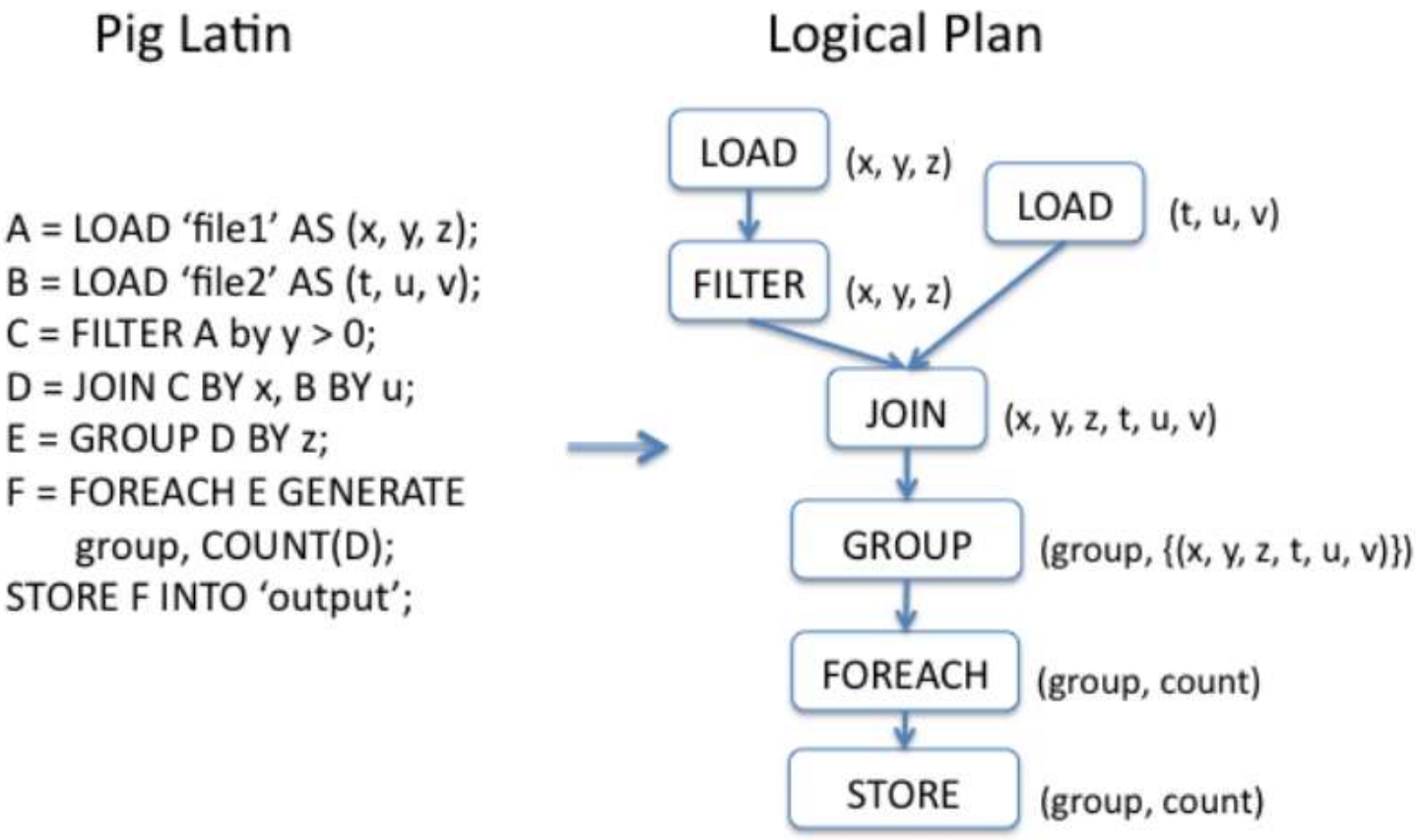


Figure 3: Map-reduce compilation of Pig Latin.

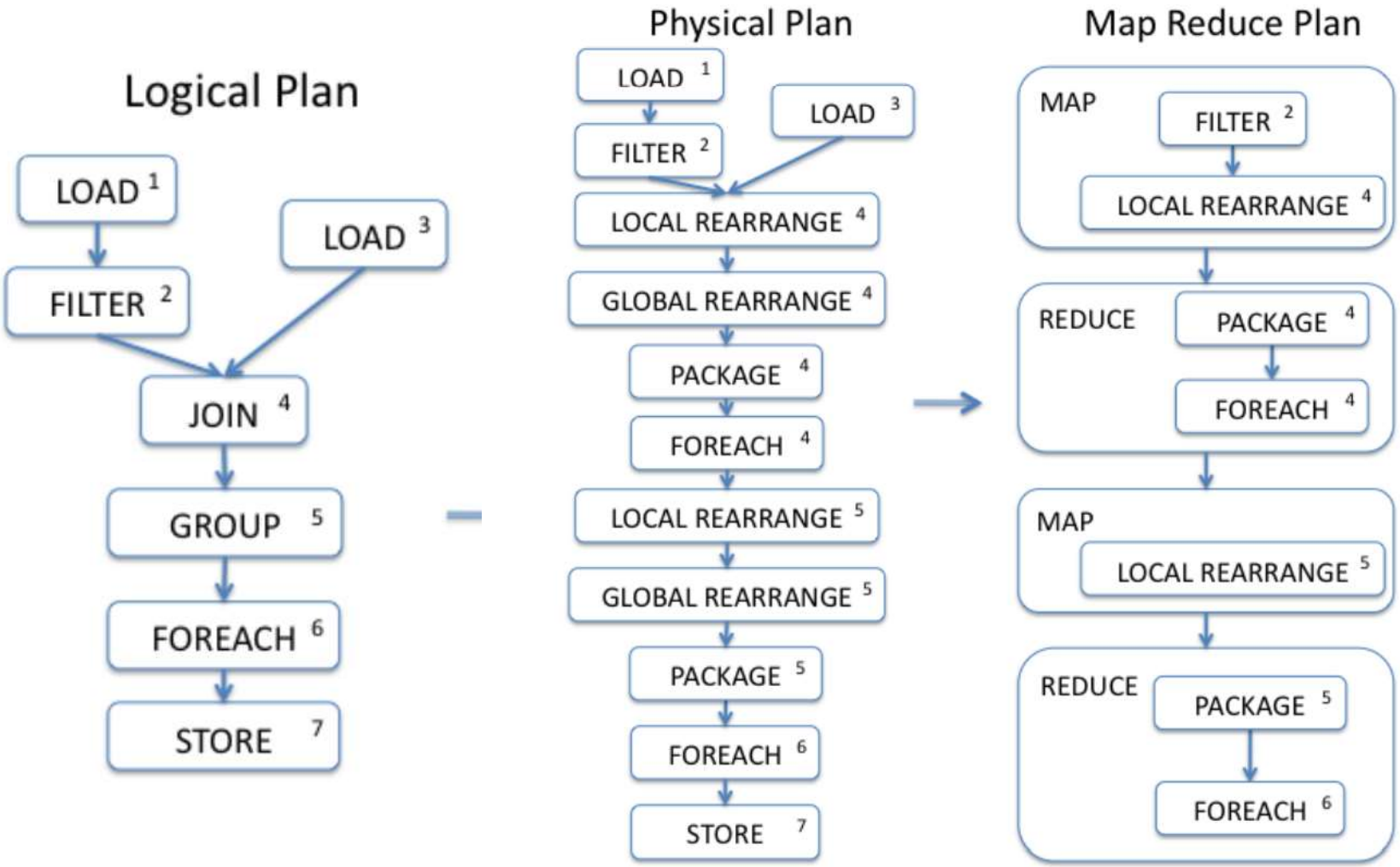
Compiling to MR (ii)

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



Compiling to MR (ii)

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



Hive

- Hive: data model and system for data warehousing in map/reduce systems.
- HiveQL: SQL programming for Map/Reduce
  - Not SQL 92 complete
  - No transactions, no materialized views, limited subquery support
- The definitive Hive paper
  - Thusoo *et al.* Hive - A Warehousing Solution Over a Map-Reduce Framework. PVLDB, 2009.

Hive Example: Status Meme

- Table schema:

```
status_updates(userid int,status string,ds string)
```

- Load log files daily:

```
LOAD DATA LOCAL INPATH '/logs/status_updates'
INTO TABLE status_updates PARTITION (ds='2009-03-20')
```

Daily Statistics

- Join logs with profiles and figure out the number of tweets from men/women and by school

```
FROM (SELECT a.status, b.school, b.gender
      FROM status_updates a JOIN profiles b
           ON (a.userid = b.userid and
               a.ds='2009-03-20' )
      ) subq1
INSERT OVERWRITE TABLE gender_summary
                       PARTITION(ds='2009-03-20')
SELECT subq1.gender, COUNT(1) GROUP BY subq1.gender
INSERT OVERWRITE TABLE school_summary
                       PARTITION(ds='2009-03-20')
SELECT subq1.school, COUNT(1) GROUP BY subq1.school
```

How do it go?

- Hive puts tables on HDFS as files and runs queries as Hadoop! jobs

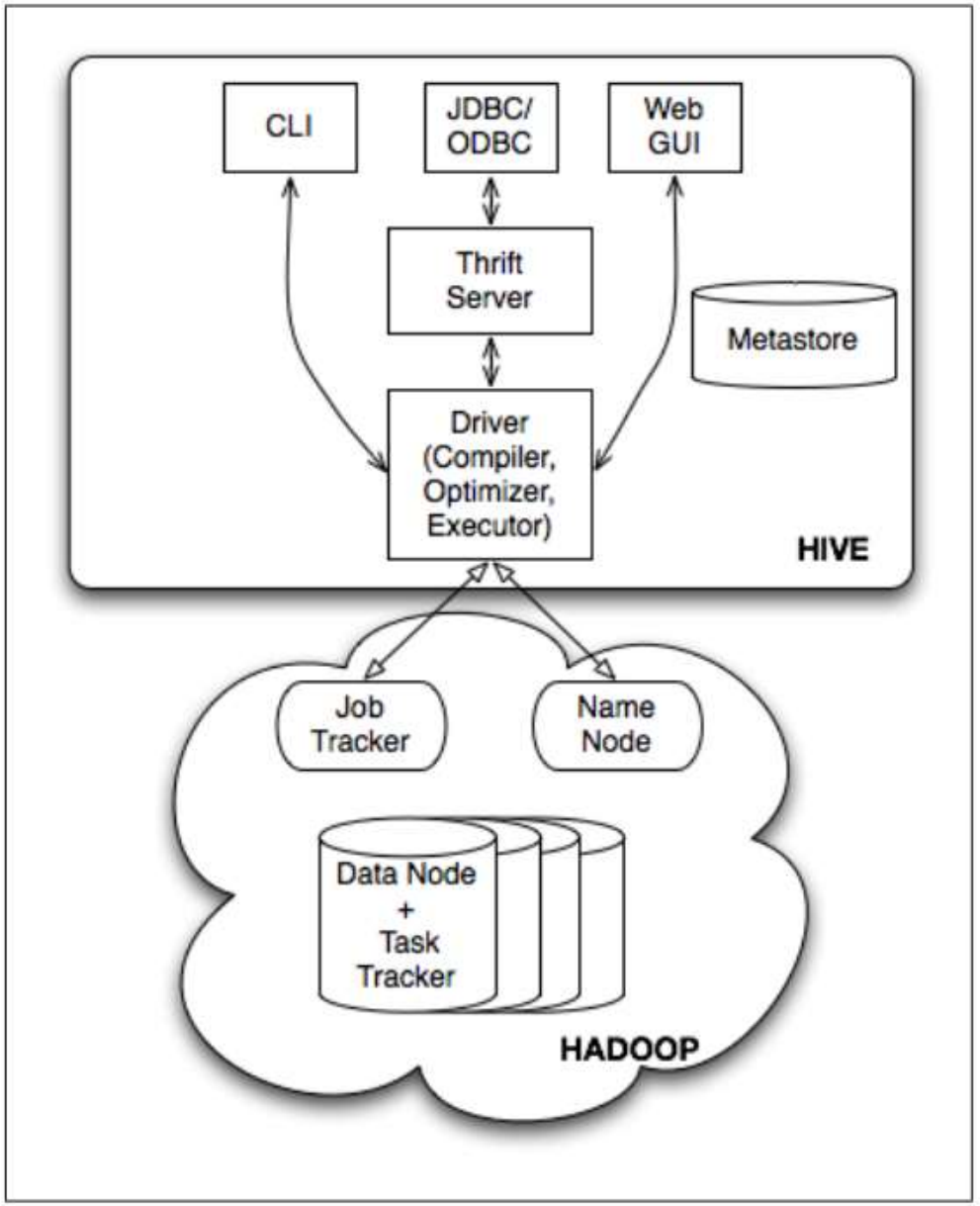
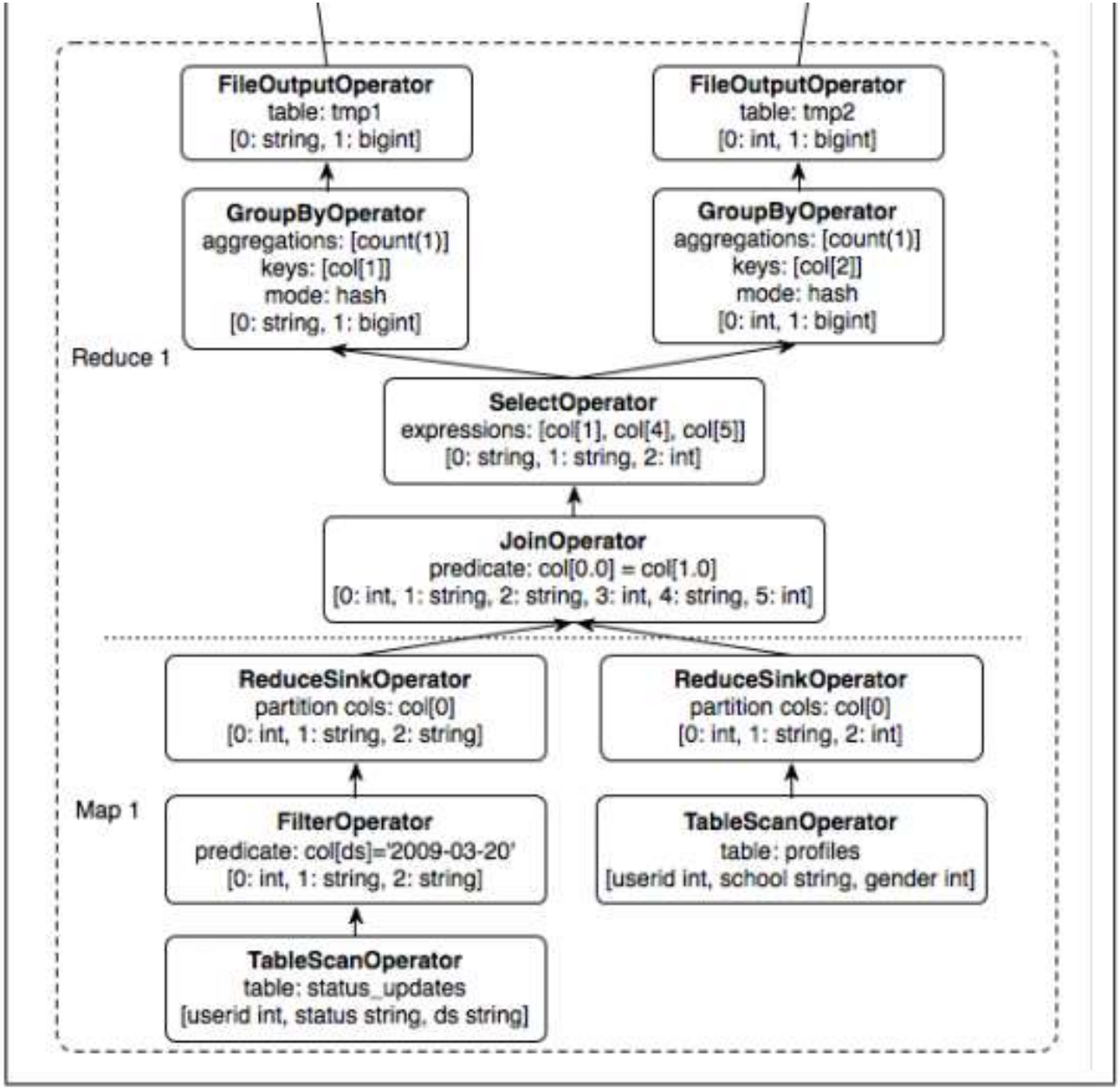


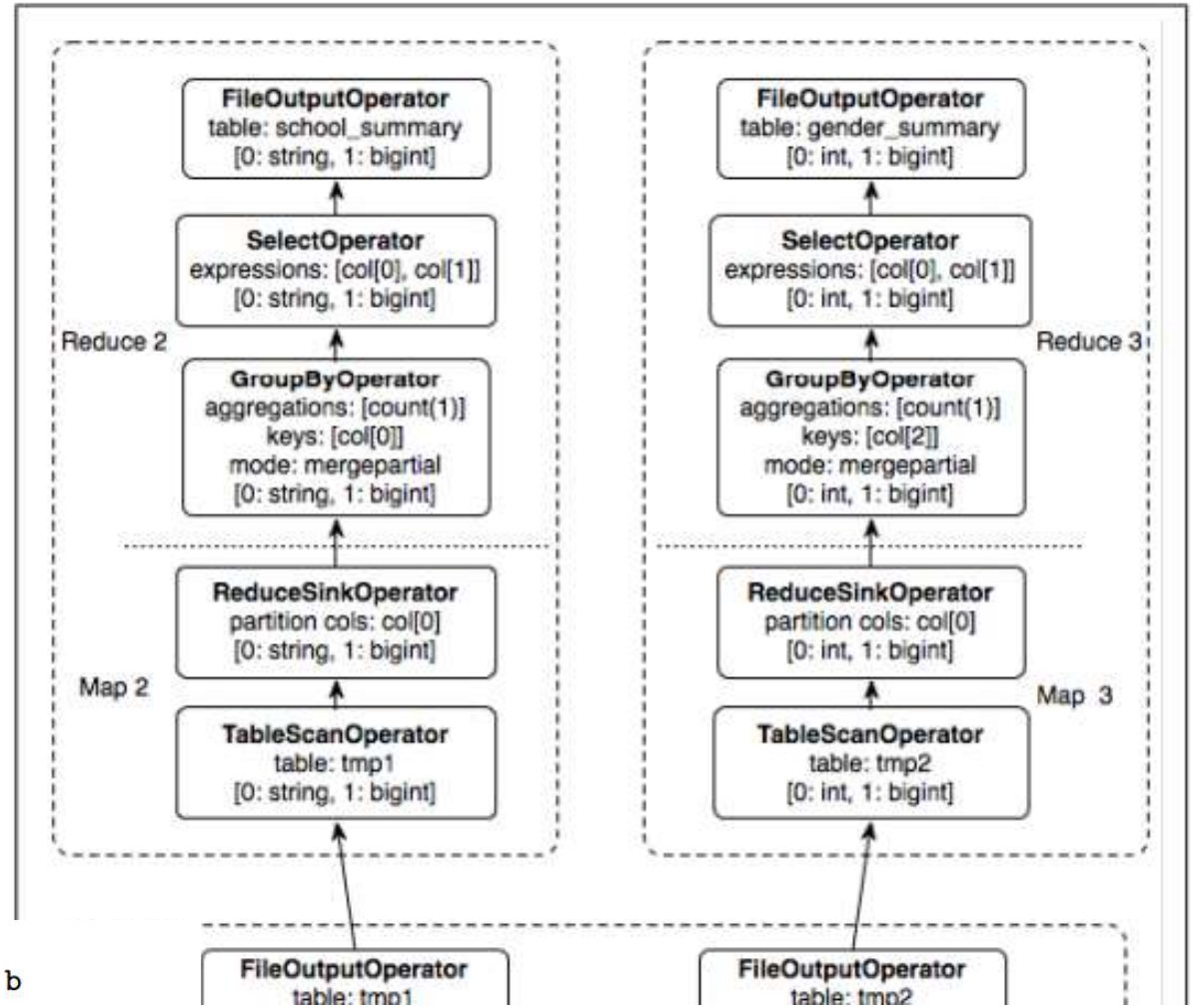
Figure 1: Hive Architecture

Resulting Query Plan (part 1)

- You don't need to understand. These are the MR jobs generated by the example



Resulting Query Plan (part 2)




```
FROM (SELECT a.status, b.school, b.gender
      FROM status_updates a JOIN profiles b
           ON (a.userid = b.userid and
               a.ds='2009-03-20' )
      ) subq1
INSERT OVERWRITE TABLE gender_summary
                       PARTITION(ds='2009-03-20')
SELECT subq1.gender, COUNT(1) GROUP BY subq1.gender
INSERT OVERWRITE TABLE school_summary
                       PARTITION(ds='2009-03-20')
SELECT subq1.school, COUNT(1) GROUP BY subq1.school
```

Take Aways

- Other ways to program M/R
  - More concise, easier to maintain
  - Particularly for data processing tasks that result in mutli-stage map/reduce programs
- Ethos: take the best from DBs
  - Declarative languages and optimization
  - Ad-hoc queries
- Ethos: and leave behind the stuff that's not parallel
  - Indexes, nested sub-queries

The (Un)Reasonable Debate

- Imperative programming
  - How humans think, step by step
  - Program encodes execution instructions
- Declarative programing
  - What! (Not how.)
  - Allows system to optimize execution
  - Non-intuitive (for many)
  - SQL != declarative programming. It is a specific instance that some love and some hate.
- PIG notable for trying to strike a happy balance
  - DB guys don't see the upside here