Other M/R Interfaces:
PIG II

EN 600.420
Instructor: Randal Burns
17 April 2017
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
    
    \[
    \{ ('alice', 'lakers'), ('alice', ('iPod', 'apple')) \}
    \]

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

\[
\begin{align*}
\text{fan of} & \rightarrow \{ ('lakers') \} \\
\text{age} & \rightarrow 20
\end{align*}
\]
PIG Expressions

- Simple set that have to be parallelizable

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

Let fields of tuple \( t \) be called \( f_1, f_2, f_3 \)

<table>
<thead>
<tr>
<th>Expression Type</th>
<th>Example</th>
<th>Value for ( t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>'bob'</td>
<td>Independent of ( t )</td>
</tr>
<tr>
<td>Field by position</td>
<td>$0</td>
<td>'alice'</td>
</tr>
<tr>
<td>Field by name</td>
<td>( f_3 )</td>
<td>'age' ( \rightarrow 20 )</td>
</tr>
<tr>
<td>Projection</td>
<td>( f_2 $.0 )</td>
<td>{ ('lakers') { ('iPod') } }</td>
</tr>
<tr>
<td>Map Lookup</td>
<td>( f_3 # 'age' )</td>
<td>20</td>
</tr>
<tr>
<td>Function Evaluation</td>
<td>\text{SUM}(f_2$.1)</td>
<td>1 + 2 = 3</td>
</tr>
<tr>
<td>Conditional Expression</td>
<td>( f_3 # 'age' &gt; 18? ) \text{'adult'}: \text{'minor'}</td>
<td>'adult'</td>
</tr>
<tr>
<td>Flattening</td>
<td>\text{FLATTEN}(f_2)</td>
<td>{ ('lakers', 1) { ('iPod', 2) } }</td>
</tr>
</tbody>
</table>

Table 1: Expressions in Pig Latin.
Bags and Co-Groupings

- Pig programming uses the pattern 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)
Compiling 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)

Compiling to MR (ii)

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