Hadoop – PIG

PIG – Why?

- **Data-parallel language (“Pig Latin”)**
  - Relational data manipulation primitives
  - Imperative programming style
  - Plug in code to customize processing

PIG – Compare

Fixed, one block
not easy to run in parallel

SQL

PIG

Easy to program

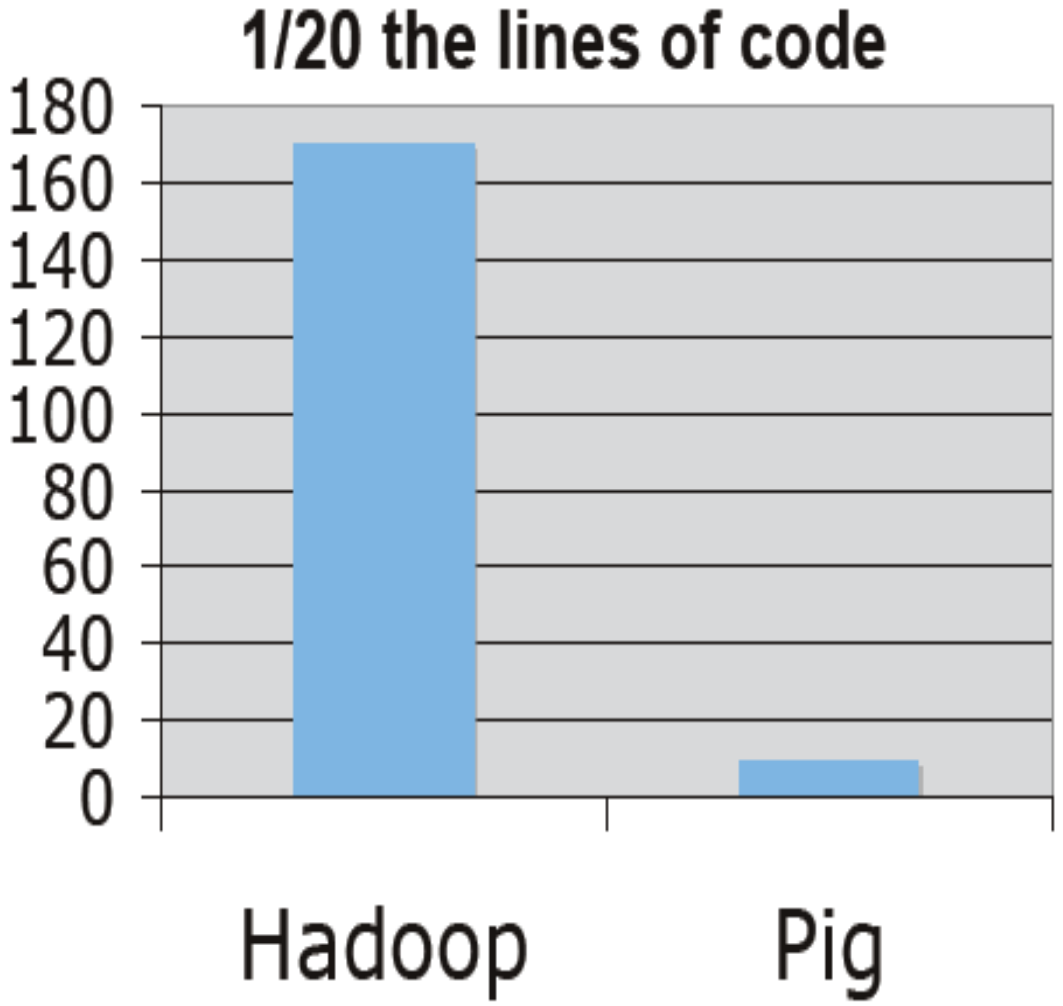
small step
easy to run in parallel

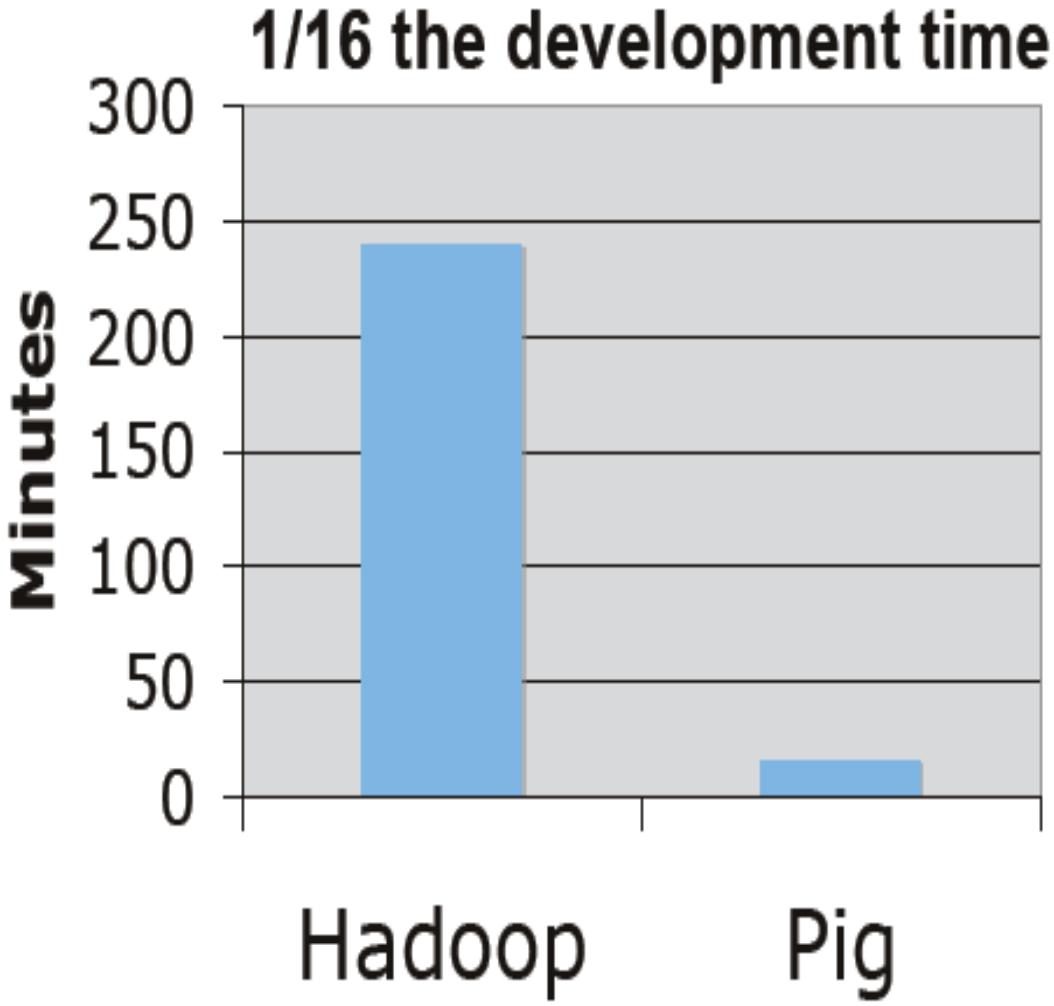
PIG

Map Reduce

Difficult to program

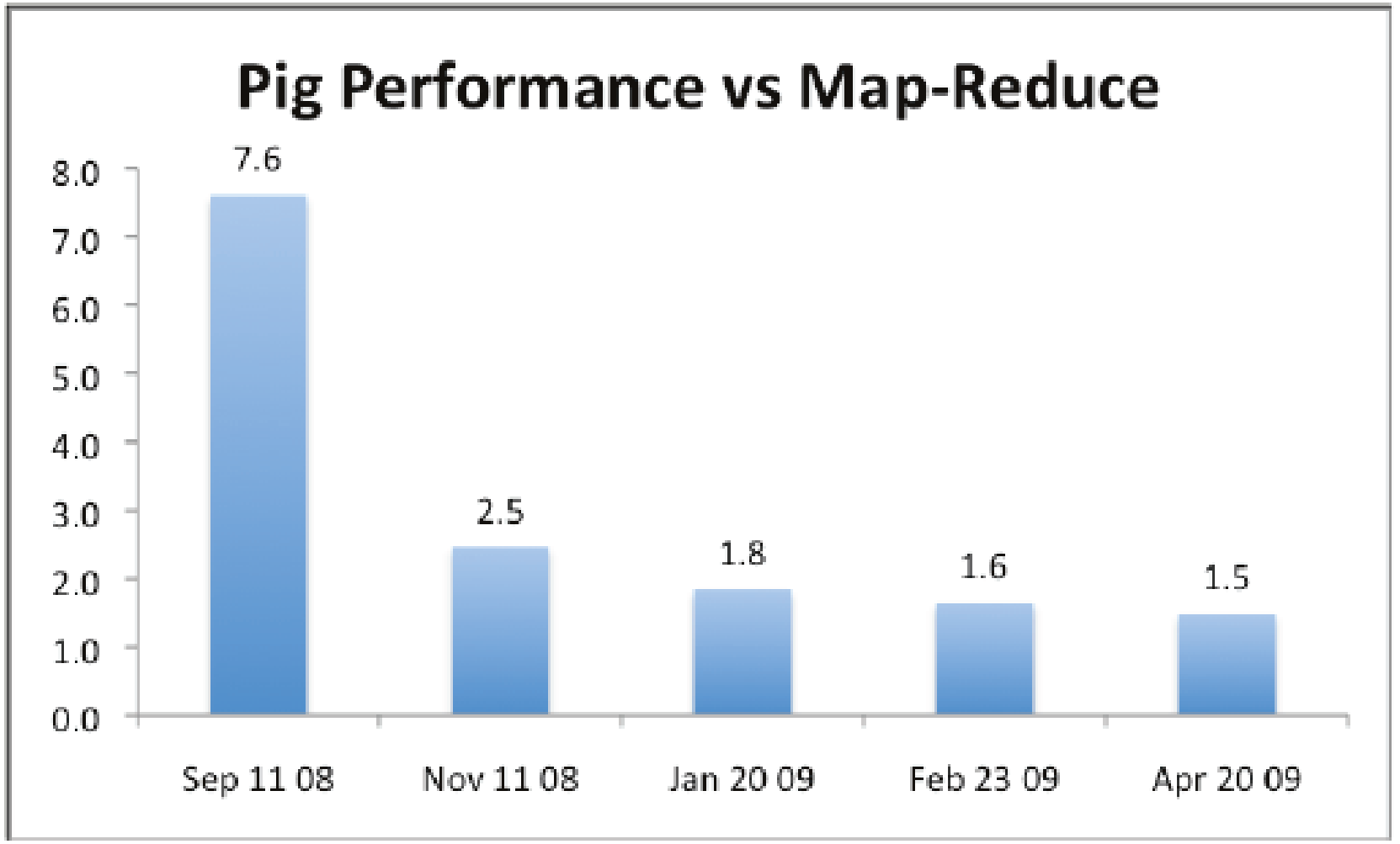
PIG – Compare -2 (development)





PIG – Compare -3 (speed)

performance
1.5x Hadoop



How to Run - PIG

- Interactive shell
- Script file
- Embed in host language (e.g., Java)

- *(coming : Graphical editor)*

PIG – What is – Examples 1

- We have -- table urls: (url, category, pagerank)
- **SQL way:**
  - SELECT category, AVG(pagerank)
    FROM urls WHERE pagerank > 0.2
    GROUP BY category HAVING COUNT(*) > 106
- **PIG way:**
  - good_urls = FILTER urls BY pagerank > 0.2;
  - groups = GROUP good_urls BY category;
  - big_groups = FILTER groups BY COUNT(good_urls)>106;
  - output = FOREACH big_groups GENERATE category, AVG(good_urls.pagerank);

PIG – What is – Examples 2

- Topic: **Detect faces in many images**

```
I = load ‘/mydata/images’ using ImageParser() as (id, image);
F = foreach I generate id, detectFaces(image); -- UserDefinedFunctions
store F into ‘/mydata/faces’;
```

PIG – What is – Examples 3

- Topic: **Find sessions that end with the "best" page**

Visits

user	url	time
Amy	www.cnn.com	8:00
Amy	www.crap.com	8:05
Amy	www.myblog.com	10:00
Amy	www.flickr.com	10:05
Fred	cnn.com/index.htm	12:00

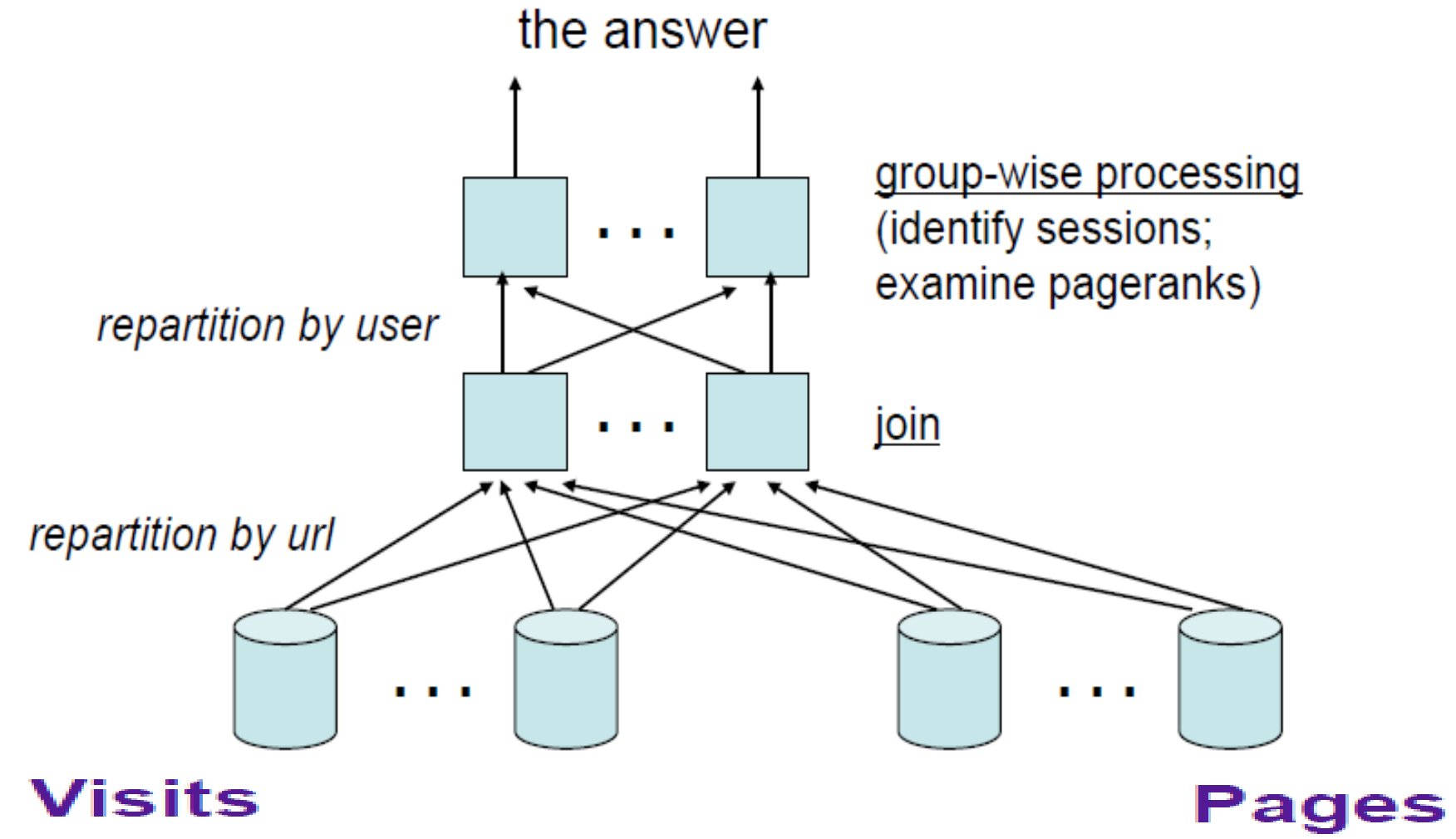
.

Pages

url	pagerank
www.cnn.com	0.9
www.flickr.com	0.9
www.myblog.com	0.7
www.crap.com	0.2

⋮

PIG – What is – Examples 3 cont.



PIG – What is – Examples 3 cont.

```
Visits = load ‘/data/visits’ as (user, url, time);
Visits = foreach Visits generate user, Canonicalize(url), time;
Pages = load ‘/data/pages’ as (url, pagerank);
VP = join Visits by url, Pages by url;
UserVisits = group VP by user;
Sessions = foreach UserVisits generate flatten(FindSessions(*));
HappyEndings = filter Sessions by BestIsLast(*);
store HappyEndings into '/data/happy_endings';
```

PIG – Diff. Operators (key words)

- **operators**:
  - FILTER
  - FOREACH … GENERATE
  - GROUP
- **binary operators:**
  - JOIN
  - COGROUP
  - UNION

For all details see http://pig.apache.org/docs/r0.14.0/basic.html

PIG – User Defined Functions (UDF)

- Simple function

```
-- myscript.pig
REGISTER myudfs.jar;
A = LOAD 'student_data' AS (name: chararray, age: int, gpa: float);
B = FOREACH A GENERATE myudfs.UPPER(name);
DUMP B;
```

```
package myudfs;
public class UPPER extends EvalFunc<String>
{
  public String exec(Tuple input) throws IOException {
      if (input == null || input.size() == 0 || input.get(0) == null)
          return null;
       try{
          String str = (String)input.get(0);
         return str.toUpperCase();
       }catch(Exception e){
          throw new IOException("Caught exception processing input row ", e);
}   }  }
```

MORE Information http://pig.apache.org/docs/r0.14.0/udf.html