Enchanting PythonS

to crunch data ...

JuanJo Ciarlante

@xjio bit.ly/jjo-cv

#PyDayMDZ

MR: what ?

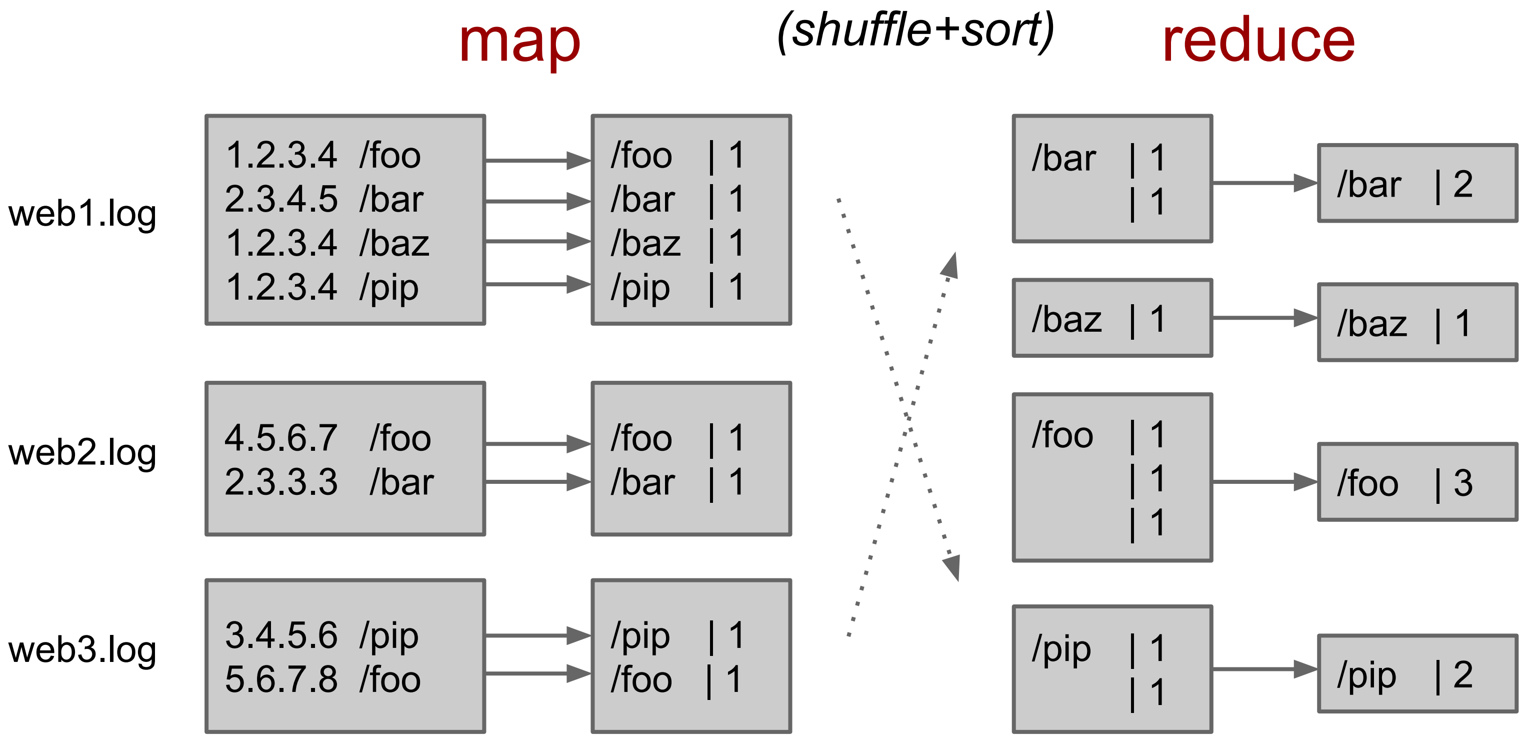
- framework for massive data processing
    - actually: data *transformation*

- based on ‘rows’/records as:
    - <key,value>

count’em all

- have: apache logs
- want: how many hits per page (urlpath) ?

e.g.: hitcount by urlpath



```
web1.log
1.2.3.4  /foo      /foo   | 1
2.3.4.5  /bar      /bar   | 1
1.2.3.4  /baz      /baz   | 1
1.2.3.4  /pip      /pip   | 1

web2.log
4.5.6.7   /foo     /foo   | 1
2.3.3.3   /bar     /bar   | 1

web3.log
3.4.5.6  /pip      /pip   | 1
5.6.7.8  /foo      /foo   | 1
```

```
/bar   | 1         /bar   | 2
       | 1

/baz   | 1         /baz   | 1

/foo   | 1         /foo   | 3
       | 1
       | 1

/pip   | 1         /pip   | 2
       | 1
```

MR: how ?

- **map**: picks data from input rows
  - *record   ---> key, data*
- (shuffle, sort) classifies by *key* to build:
  - *...   ---> key, [data1, data2, …]*
- **reduce**: aggregates, transforms - eg:
  - *key, […] ---> key, sum([...])*

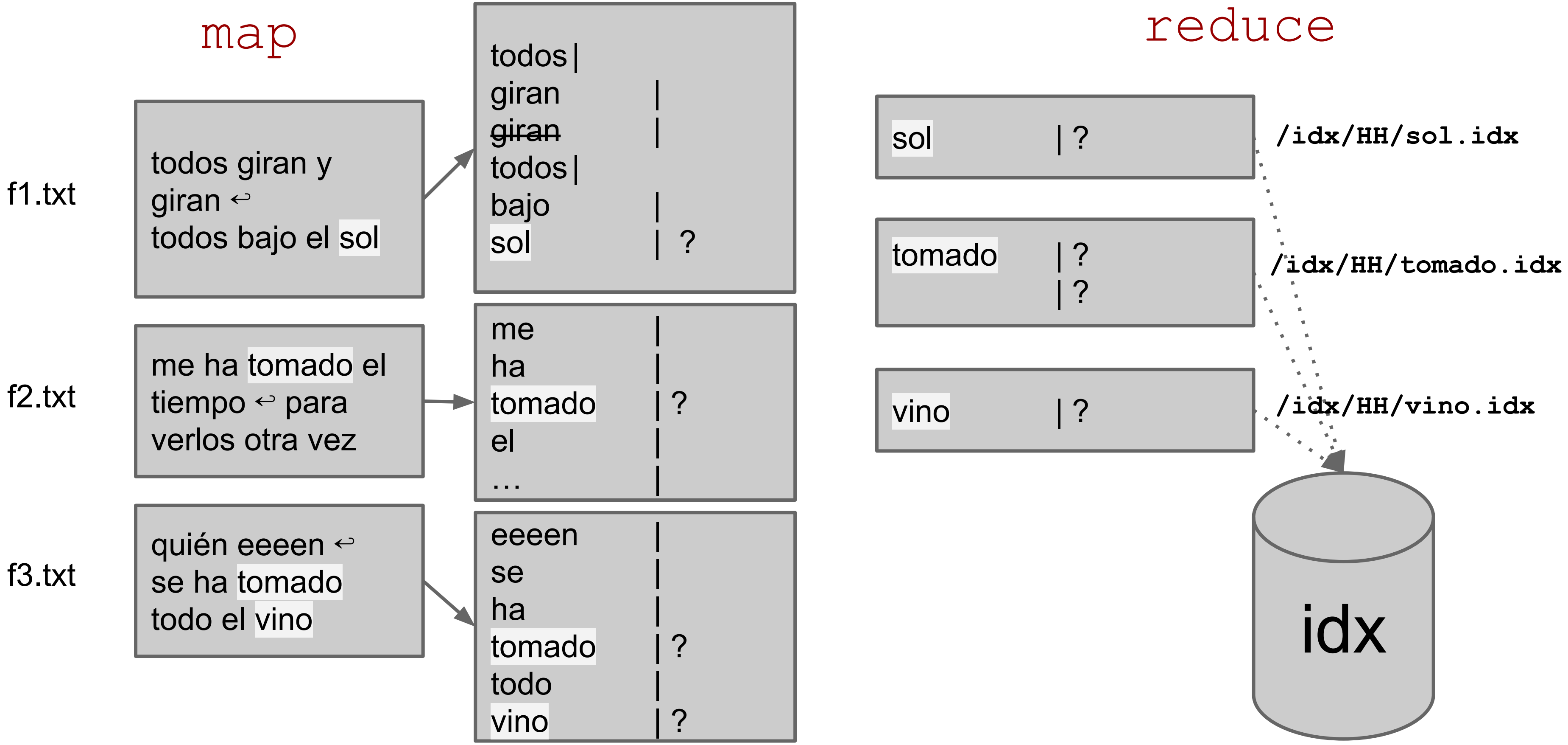
MR: why is cool ?

- *kiss*:
  - really simple model
- *scalability*:
  - parallel-friendly by design
- *data-locality*:
  - distributed FS
- *sync-free*:
  - no explicit required IPC/sync between tasks

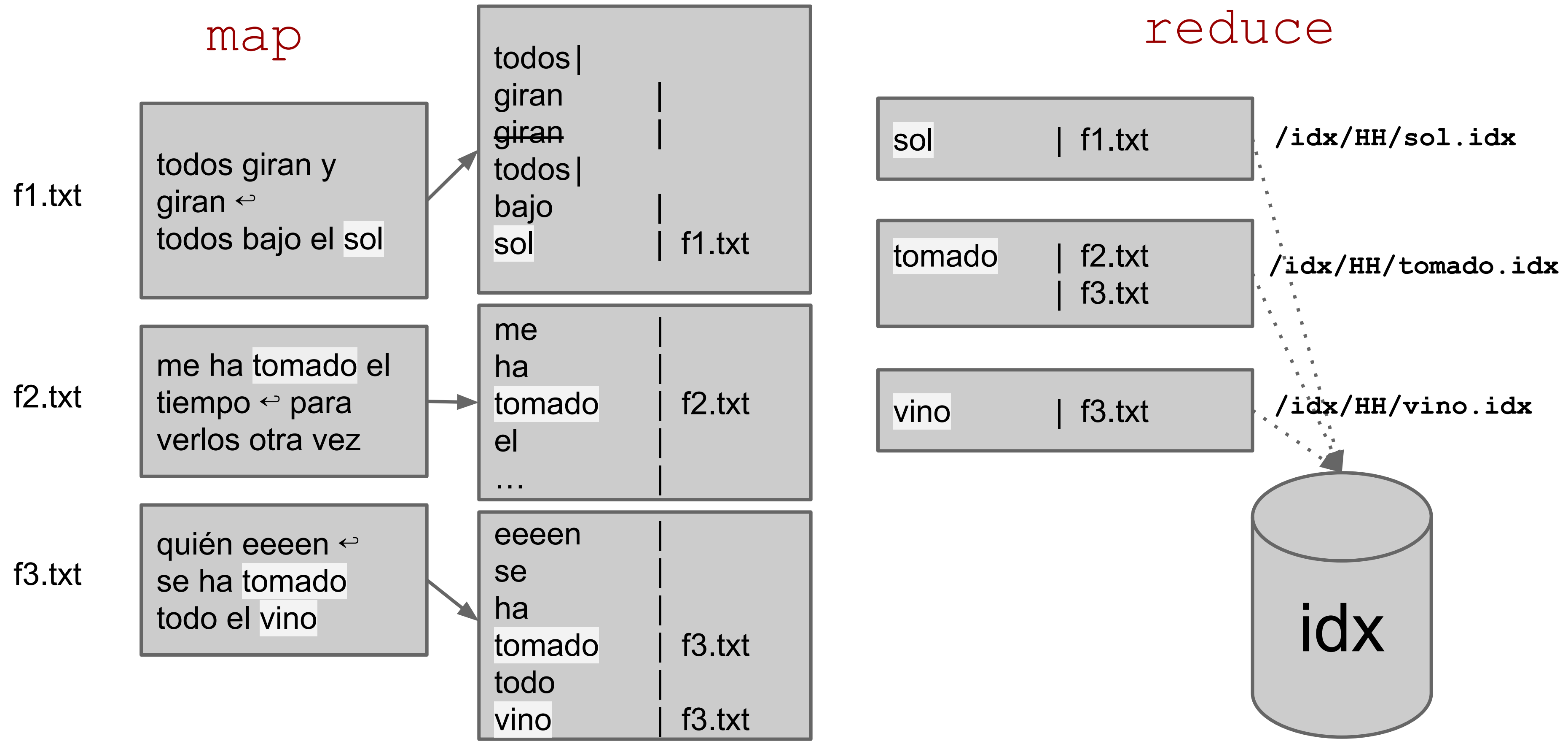
gimme that index

- have: corpus of documents
- want: to search them by word (grep)

e.g.: grep - filename by word



e.g.: grep - filename by word



todos giran y giran ↩ todos bajo el sol

me ha tomado el tiempo ↩ para verlos otra vez

quién eeeen ↩ se ha tomado todo el vino

todos	
giran	
~~giran~~	
todos	
bajo	
sol	f1.txt

me	
ha	
tomado	f2.txt
el	
…	

eeeen	
se	
ha	
tomado	f3.txt
todo	
vino	f3.txt

sol	f1.txt

tomado	f2.txt
	f3.txt

vino	f3.txt

MR: Hadoop

- floss \o/
- in Java :/, for Java :(
  - ¿ too much Javanic :-?
- => hadoop “streaming” \○/
  - arbitrary commands with pipelined data locality:
    input | python $\text{mr.py}_{\text{map}}$ | *s+sort* | python $\text{mr.py}_{\text{reduce}}$

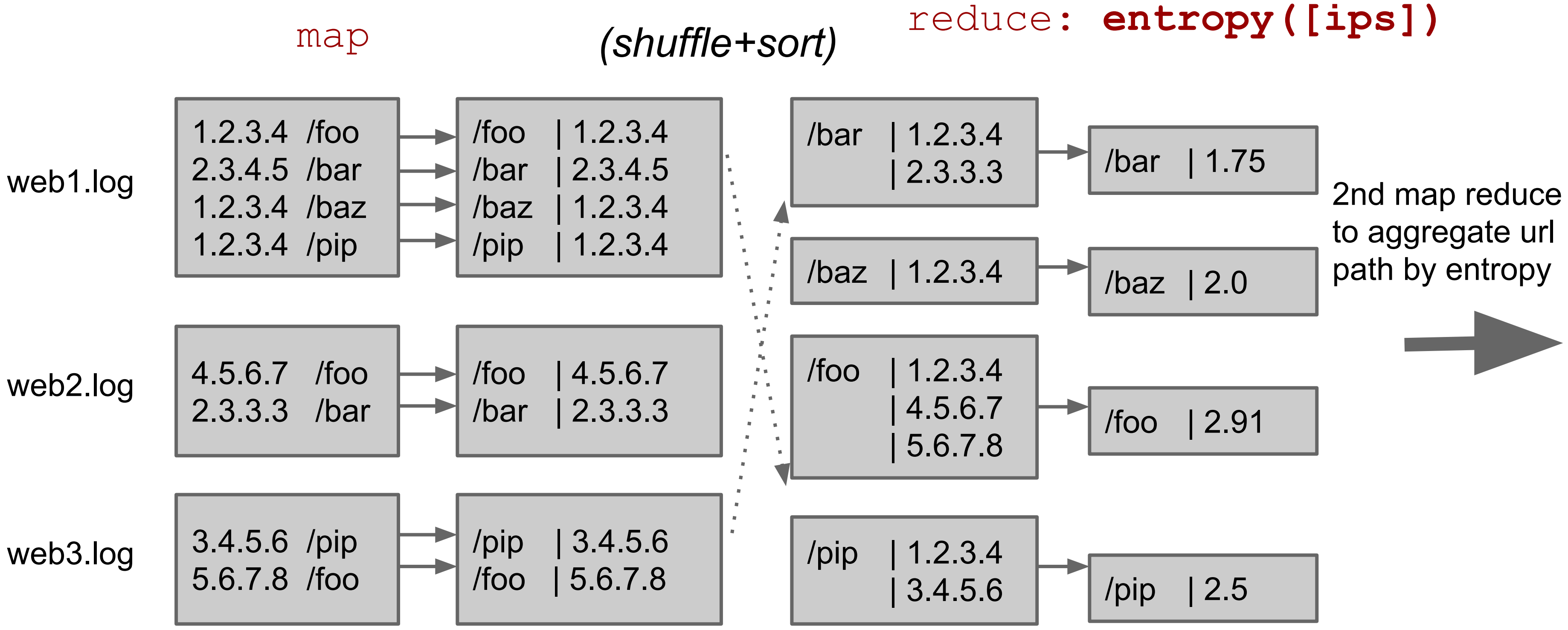
MR: some python libs

- **MRJob**
  - 👍 **local**, **hadoop**, Elastic MR (AWS)
  - 👎 not hadoop ‘native’
- **hadoopy**
  - 👍 optimized for hadoop, **supports HDFS bin formats**
  - 👎 only hadoop
- **discoproject.org**
  - 👍 **100% python**
  - 👎 python only, down to the DFS

speaking of diversity ...

- have: apache logs
- want: to know how diversity of client IPs per page
  - shamelessly use entropy(concatenated_IPs_bits) as a *proxy* value for relative diverisity

e.g.: urlpath diversity



MRjob: hitcount.py

https://github.com/jjo/src-juanjo/blob/master/python/mrjob/j01-hitscount.py

```python
from mrjob.job import MRJob

class MRHitCount(MRJob):
    def mapper(self, _, line):
        ip, path =line.split()
        yield path, 1

    def reducer(self, key, values):
        yield key, sum(values)

if __name__ == '__main__':
    MRHitCount.run()
```

MRjob: grep.py

https://github.com/jjo/src-juanjo/blob/master/python/mrjob/j02-grep.py

```python
from mrjob.job import MRJob
from mrjob.compat import get_jobconf_value

class MRGrep(MRJob):
    def mapper(self, _, line):
        for word in line.split():
            yield word, get_jobconf_value('map.input.file')

    def reducer(self, key, values):
        yield key, str(values)

if __name__ == '__main__':
    MRGrep.run()
```

MRjob: urlentropy.py

https://github.com/jjo/src-juanjo/blob/master/python/mrjob/j04-entropy.py

```python
class MREntropyPerURL(MRJob):
    # 1st MR: urlpath -> entropy([ips])
    def input_mapper(self, _, line):
        ip, path = line.split()
        yield path, ip

    def urlpath_to_entropy(self, key, values):
        yield key, entropy_bits(values)

    # 2nd MR: aggregate all urlpaths by same entropy_val (omitted)
    # Pipe-line both MRs:
    def steps(self):
        return [self.mr(mapper=self.input_mapper, reducer=self.urlpath_to_entropy),
                self.mr(mapper=self.swap_values, reducer=self.values_per_key)]

if __name__ == '__main__':
    MREntropyPerURL.run()
```

Hacktime \o/

- these slides:
  - http://bit.ly/jjo-mrpy-14
- some MR py libs:
  - mrjob, hadoopy, hadoop, happy
- interesting datasets:
  - https://snap.stanford.edu/data/ networks
  - http://aws.amazon.com/datasets/ diverse data
- complete source code for this slides
  - https://github.com/jjo/src-juanjo/tree/master/python/mrjob