MAPREDUCE CODE

Input Data

Each mapper process a portion of the input data and each will be given a line

- text is usually freeform so we use something like a regular expression
- sometimes is regular, like here

2012-01-01 12:01 San Jose 12.99 Amex

tab delimited, split by tab and extract values ... then use values needed to create key-value pairs for the mapreduce job

Defining Mapper Code

```
def mapper():
    for line in sys.stdin:
        data = line.strip().split("\t")
        *date, time, store, item, cost, payment = data
```

We want to note that no matter what data we get, the mapper will keep working

```
*F len(data) == 6:
    ...
```

Between map and reduce there shuffle and sort!

(Hadoop streaming allows you to write in any language)

Reducing key-value pairs are split among reducers. They will be sort (from shuffle and sort)

```
Miami 12.34
Miami 99.07
Miami 3.14
Nyc 98.77
Nyc 88.99
```

```
def reducer:
    salesTotal=0          # to check when the store changes
    oldKey = None
    for line in sys.stdin:
        data = line.strip().split("\t")
        if len(data) != 2:
            continue
        thisKey, thisSale = data
        if oldKey and oldKey != thisKey:
            print "!"
            salesTotal=0
        oldKey = thisKey
        salesTotal += float(thisSale)
    if oldKey != None
        print "!"
```

weird data:

if oldKey is set and we finished processing a store, print it

set variables for next loop

to print the result for the last store

testing it

```
head -50 purchases.txt > testFile
cat testFile | ./mapper.py
```

for the reducer ...

```
cat testFile | ./mapper.py | sort | ./$reducer.py
```

```
hadoop -fs -cat output/part-0000 | less
```

↓

see the hadoop output!

You can now run it on hadoop and use program on the job tracker

MAPREDUCE PATTERNS

Filtering patterns, sampling, top-N
summarization pattern: counting, min/max
statistics, index
structural patterns, combining datasets

Filtering patterns
- simple filter: just a function
- Bloom Filter: more efficient, probabilistic
- sampling: make a smaller dataset from a larger
  by pulling out samples
- random sampling
- top N

Summarization patterns
give quick easy high level understanding
of your data
- Inverted index
- Numerical summarizations
  (word/record count, mean, median, std dev.)

Combiners (between map and reduce phase)
Transfering all the data from mappers
to reducers is a lot of bandwith,
so we do some pre-reduction before
passing the data to reducers

Structural patterns
Migrating data to hadoop (take advantage of
hierarchical data)
You need:
- Data sources linked by foreign key
- Data must be structured and row-based

Don't need to do joins, save time reformating your data
by