The MapReduce programming model

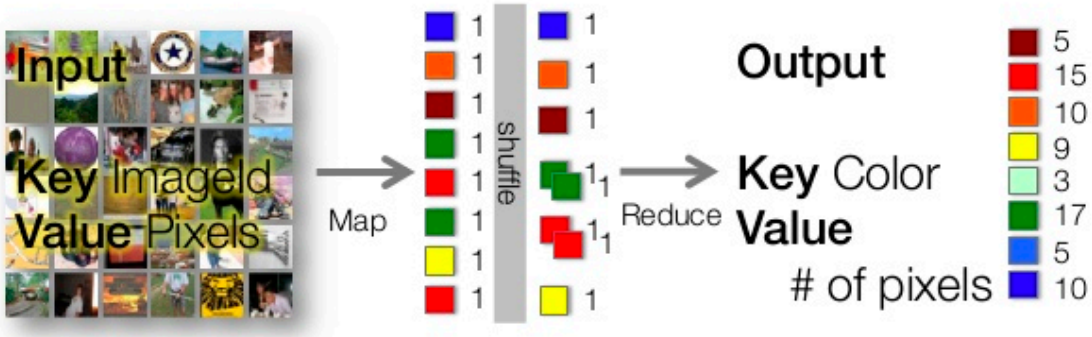
Input a list of (key, value) pairs

Map apply a function f to all pairs

Reduce apply a function g to all values with key k (for all k)

Output a list of (key, value) pairs

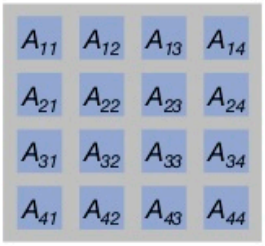
Computing a histogram
A simple MapReduce example




```
Map(ImageId, Pixels)
for each pixel
  emit
    Key = (r,g,b)
    Value = 1
```

```
Reduce(Color, Values)
emit
  Key = Color
  Value = sum(Values)
```

Many matrix computations are possible in MapReduce



(3,4) -> 5
(1,2) -> -6.0
(2,3) -> -1.2
(1,1) -> 3.14
...
"Coordinate storage"

Column sums are easy

Input Key (i,j) **Value** A_{ij}

Map((i,j), val)	**Reduce**(j,Values)
emit	emit
Key = j, Value = val	Key = j, Value = sum(Values)

Other basic methods

can use common parallel/out-of-core algs

Sparse matrix-vector products **y** = **Ax**

Sparse matrix-matrix products **C** = **AB**

The MapReduce programming model

Input a list of (key, value) pairs

Map apply a function f to all pairs

Reduce apply a function g to all values with key k (for all k)

Output a list of (key, value) pairs

Map function f must be side-effect free
All map functions run in parallel

Reduce function g must be side-effect free
All reduce functions run in parallel