MAPREDUCE FOR SCALING K-MEANS

Motivating MapReduce

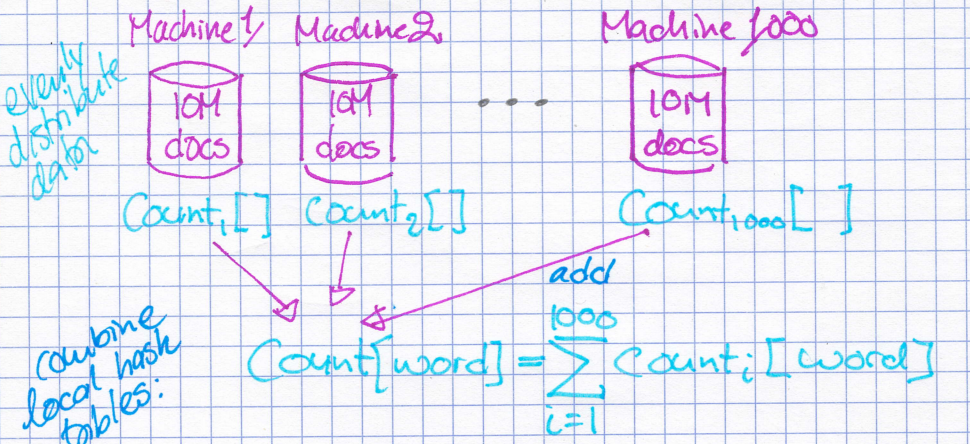
Counting words on a single processor
(The "Hello World!" of MapReduce)

Suppose you have 108 documents and 1 machine and want to count the # of occurences of each word in the corpus

Code:

```
count[ ] ← init hash table
for d in documents
    for word in d
        count[word] += 1
```

Naive parallel word counting
- word counts are independent across documents (data parallel)
- count occurences in sets of documents separately, then merge



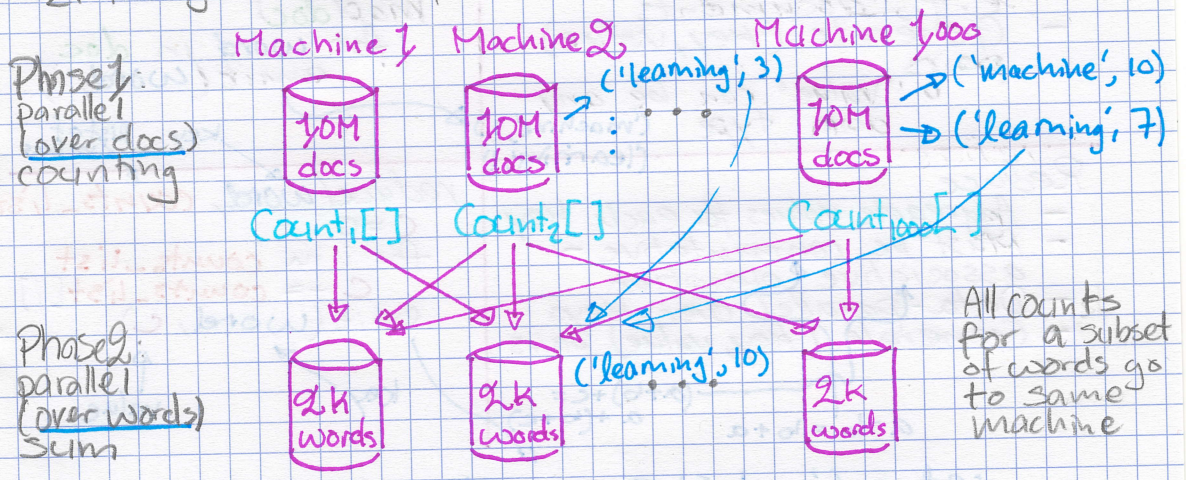
How do we do this for all words in vocab?

Back to sequential problem to merge counts...

Have to cycle through all words in vocab...

Counting words in parallel & merging tables in parallel

1. Generate pairs (word, count) in parallel
2. Merge counts for each word in parallel



How to map words to machine? Use a hash function!

h(word index) → machine index

Which words go to machine i?

$h: V_{vocab} \rightarrow [1, 2, \ldots, \#machines]$

Send counts of 'learning' to machine h['learning']

MapReduce abstraction

Map:
- Data-parallel over elements, e.g., documents
- Generate (key, value) pairs
  - "value" can be any data type

Reduce:
- Aggregate values for each key
- Must be commutative-associative operation
- Data-parallel over keys
- Generate (key, value) pairs

a+b = b+a

(a+b)+c = a+(b+c)

Word count example:

```
map(doc)
    for word in doc
        emit(word, 1)
```

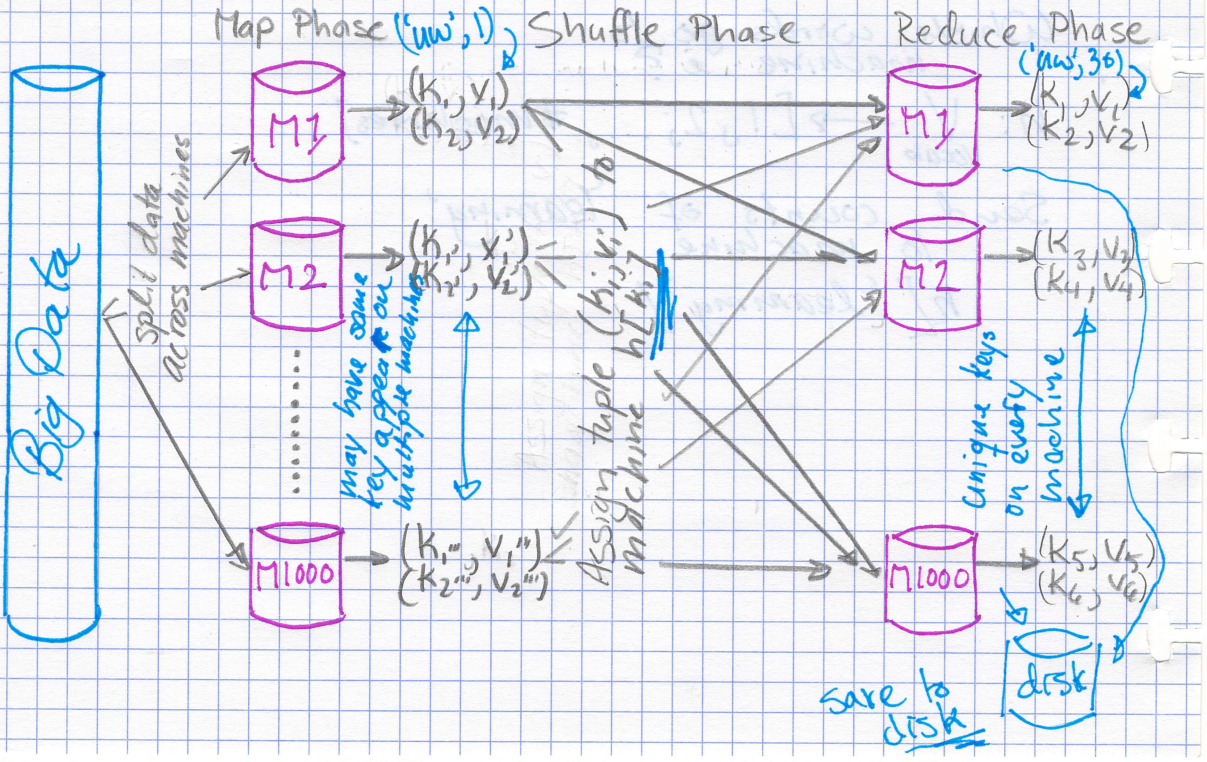
('machine', 1)
('learning', 1)

```
reduce(word, counts_list)
    c = 0
    for i in counts_list
        c += counts_list[i]
    emit(word, c)
```

key — list of values — key — value

reduce('uw', [1,17,0,0,12,2])
emit('uw', 32)

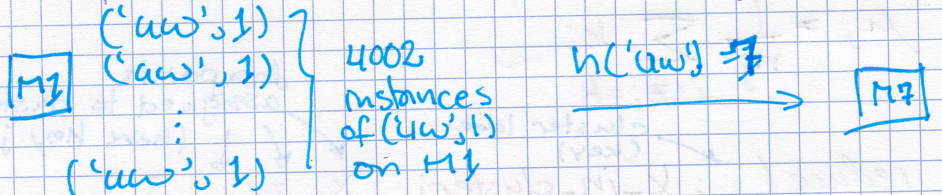
MapReduce execution overview and combiners



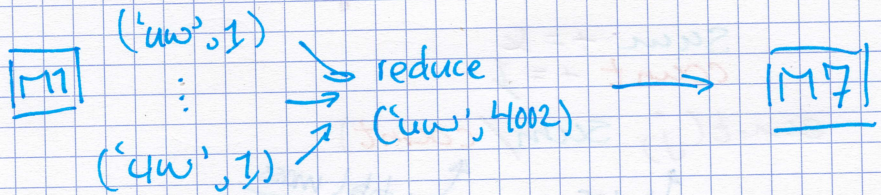
Improving performance:

Combiners

- naive implementation of MapReduce is very wasteful in communication during shuffle:



- Combiner: Simple solution: Perform reduce locally before communicating for global reduce
  - Works because reduce is commutative - associative



MapReduce for k-means

MapReducing 1 iteration of k-means

Classify: Assign observations to closest cluster center

$$z_i \leftarrow \arg\min_j \|\mu_j - x_i\|_2^2$$

Map: For each data point, given $(\{\mu_j\}, x_i)$, emit (z_i, x_i)

Recenter: Revise cluster centers as mean of assigned observations

$$\mu_j = \frac{1}{n_j} \sum_{i: z_i = k} x_i$$

Reduce: Average over all points in cluster j $(z_i = k)$

Classification step as Map

Classify: Assign observations to closest cluster center

$$z_i \leftarrow \arg\min_j \|\mu_j - x_i\|_2^2$$

map($[\mu_1, \mu_2, \ldots, \mu_k]$, x_i) → $z_i \leftarrow \arg\min_j \|\mu_j - x_i\|_2^2$; emit($z_i$, x_i)

- $[\mu_1, \mu_2, \ldots, \mu_k]$: set of cluster centers
- x_i: data point
- z_i: cluster label
- x_i: data point

e.g. emit(2, [17, 0, 1, 7, 0, 0, 5])

- 2: $z_i = 2$ (assigned to cluster 2)
- [17, 0, 1, 7, 0, 0, 5]: data point x_i

Recenter step as Reduce

Recenter: Revise cluster centers as mean of assigned observations

$$\mu_j = \frac{1}{n_j} \sum_{i:z_i=k} x_i$$

data points assigned to cluster j (have key j)

cluster label (key)

```
reduce( j , x_in_clusterj : [x_1, x_3, ..., ] )
    sum = 0      <- total mass in cluster
    count = 0    <- total # of obs. in cluster
    for x in x_in_clusterj
        sum += x
        count += 1
    emit(j, sum/count)
```

emit(j, ...): j = cluster label

sum/count = total mass / total # obs

Some practical considerations

k-means needs an iterative version of MapReduce
- Not standard formulation

Mapper needs to get data point and all centers
- A lot of data
- Better implementation: mapper gets many data points

Summary of parallel k-means using MapReduce

Map: classification step; data parallel over data points

Reduce: recompute means; data parallel over centers

Other applications of clustering

Clustering Images

- For search, group as:
  - Ocean
  - Pink flower
  - Dog
  - Sunset
  - Clouds
  - ...

Structuring web search results
- Search terms can have multiple meanings
- Example: "cardinal"
- Use clustering to structure output

Grouping patients by medical condition
- Better characterize subpopulations and diseases

Products on Amazon
- Discover product categories from purchase histories

~~"furniture"~~
"baby"

- Or discovering groups of users

Discovering similar neighborhoods
- Task 1: Estimate price at a small regional level
- Challenge:
  - Only a few (or no!) sales in each region per month
- Solution:
  - Cluster regions with similar trends and share information within a cluster
- Task 2: Forecast violent crimes to better task police
- Again, cluster regions and share information!
- Leads to improved predictions compared to examining each region independently