MapReduce

S = Massive Data

Mapper(S): s in S -> {(key,value)}

Shuffle({(key,value)}) -> group by "key"

Reducer ("key,value_i}) -> ("key, f(value_i))

Can repeat, constant # of rounds

[Tao + Lin + Xiao 2013]

Minimal MapReduce Algorithm
N = size of data
\( t = \text{number of machines} \)
\( m = \frac{N}{t} = \text{# objects per machine if distributed evenly.} \)
\( m < M = \text{Mem size} \)

1) At all times each machine has \( O(m) \) storage
2) Each machine sends/receives \( O(m) \) items
3) constant # rounds
4) Optimal computation: each machine performs \( O(\frac{T_{seq}}{t}) \) in total
   \( O(\frac{T_{seq}}{t}) \) per machine per round.

1)+2) prevents partition skew
\( m = \frac{N}{t} \) allows to scale to any # machines !
2) ensures total traffic is \( O(N) \)
   no straggling machine
   ensures is stateless (resilience, can use fake larger \( t \) + load balancing)
3) for practicality
4) energy cost is low

-----------------------------------------

Sorting

TeraSort:  http://sortbenchmark.org
Ellapsed time to sort 10^12 bytes = 1TB  (now 100 TB)
measured in TBs/minute
   -> record (2013)  1.42 TB minute on 102.5 TB.

2009: Hadoop 100 TB in 172 min  (0.572 TB / min)  (3452 machines)
500 GB in 1 minute (on 1406 machines)
previous used fewer, but expensive machines

How does it work?

\[ \text{parameter } k = t \ln (N \cdot t) \]

-----------------------------

Map 1:
For all \( s \) in \( S \), with prob \( \frac{k}{N} \)
\[ \rightarrow \{<1,s> <2,s> \cdots <t,s>\} \quad \text{<original TeraSort, only send to 1>} \]

Reduce 1:
On each node: \( <j, \{s_1 \ldots s_{\lnot k}\} = Q> \) (same \( Q \))
\[ \rightarrow \text{sort}(Q), \text{choose } t-1 \text{ even spaced items } b_1, b_2, \ldots, b_{t-1} \]
\[ b_j = j[k/t] \text{th item} \]
\[ b_0 = -\infty, b_t = \infty \]

\[ \text{Map 2:} \]
\[ \text{For all } s \text{ in } S: \text{ find } j \text{ s.t. } b_{\lnot j-1} < s \leq b_j \]
\[ \rightarrow <j, s> \]

Reduce 2: \( <j, \{s, s', \ldots\} = S_j> \)
\[ \rightarrow <j, \text{sort}(Q_j)> \]

-------------------------------

Central Limit theorem (Chernoff Bound) \( \frac{k}{2} < |Q| < k \) w.h.p.

Need:
(1) \( |Q| = O(m) \) fine for \( t = O(m / \log (N)) \)
(2) for all \( j \), \( |S_j| = O(m) \)

Given (2), then \( T_j = \frac{(N)}{t} \log (N/t) \)
\[ \sum_j T_j = \frac{(N)}{t} \log (N/t) = N \log (N/t) < N \log N \]

Prove (2):
eps-net: Given \( k = (1/\epsilon) \ln (1/\epsilon \cdot \delta) \) samples, w.p > 1-
\delta:
\[ \text{each interval of size } \epsilon N \text{ has at least one point} \]
\[ \text{-> each } |S_j| \leq N/t + 2*\epsilon N \]
(not completely obvious, symmetric difference)
set \( \epsilon N = N/t \rightarrow t = 1/\epsilon \)
\[ \rightarrow k = t \ln (t/delta) \]
\[ w.p.h = w.p > 1-1/N \rightarrow k = t \ln (tN) \]

------------

Makes many tasks Minimal: e.g. Prefix Sum:
Sort (2 rounds)
Reduce2: also computes $\text{agg}(S_j) = \text{sum}(S_j) = g_j$

$\rightarrow \{<1,g_j>, <2,g_j>, ..., <j,g_j>\}$

$\rightarrow \{<j,s> \text{ for all } s \text{ in } S_j\}$

Map3: identity

Reduce3: node $j$:

\[
W_j = \sum_{i=1}^{j} g_j
\]

for $s_i$ in $S$

\[
W_j += w_i
\]

\[
p_i = W_j
\]

-----------------------------------------

Sliding Aggregates

$S$ has $N$ objects: ordered, each $s_i$ has weight $w_i$

integer $l < N$

distributed aggregate \text{agg} (e.g. Sum, Min, Max)

for $S_1$ and $S_2$ have \text{agg}(\text{agg}(S_1), \text{agg}(S_2)) = \text{agg}(S_1 \text{ union } S_2)$

$\text{window}(i) =$ $l$ largest items not exceeding $s_i$

sliding window statistics

Rounds 1+2 $\rightarrow$ Sort $\rightarrow$ $S_1, ..., S_t$

Round 3 $\rightarrow$ use rank (prefix sum w/ $w_i = 1$) to have each $|S_j| = m$

*exacty*

Round 4:

Map 4: (really Reduce 3)

$+ \text{Send } A_j = \text{agg}(S_j) \text{ to all machines}$

$\{<1,A_j>, <2,A_j>, ..., <t,A_j>\}$

$+ \text{Send } <[(i-l)/t], w_i> \text{ for all } s_i \text{ in } S_j$

Reduce:

\[
\text{window}(i) = \text{agg}\left(\text{agg}_{l = i-l}^{[(i-l+1)/t]}*k w_i, A_{[(i-l+1)/t]}, A_{[(i-l+2)/t]}, ..., A_{[(i-1)/t]}\right)
\]

\[
\text{agg}_{\{s_l \text{ in } S_j, l<i\}} w_i
\]

can be done in $O(m)$ time

** each $s_i$ important for at most 2 units, we know which ones **