P-to-K Mapreduce in MPI

Aaron Rodgers
CS 525 Presentation, 2014-12-02
Outline

- Code structure
- Straightforward MPI_Reduce() solution
- In-transit reduction solution
- Pathologic datasets
  - Sparse Keys
  - Poor Partitions
## Code Structure

- We have $p$ mappers (8-32), $k$ reducers (4).
- At each mapper $p$:
  - Generate $100M/p$ key-value pairs across $p$ nodes.
    - Limit keys to 1M possible unique values.
      - (Simulates a map operation.)
  - Perform a local sort.
  - Perform a local reduction.
  - Divide kvpairs into $k$ partitions.
- Reduce partitions to $k$ reducers.
Code Structure

At each mapper p:

Sort

K  V
8211 1
3422 1
4510 1
1413 1
4492 1
7121 1
...

Reduce

K  V
0011 1
0011 1
0011 1
0013 1
0013 1
0013 1
...

Partition

K  V
0011 3
0018 8
...

K  V
0013 4
0021 12
...

Code Structure

p mappers

a
b
c
d
e
f
g
h

k reducers

a+c+e+g

b+d+f+h

Reduction
MPI_Reduce()

Our baseline solution is to perform \( k \) calls to MPI_Reduce():

```c
for(i = 0; i < K; i++){
    kvpair *tmp = malloc(p_lengths[i] * sizeof(kvpair));
    MPI_Reduce(partitions[i], tmp, p_lengths[i] * 2, MPI_INT, op, i,
               MPI_COMM_WORLD);
}
```
MPI_Reduce()
hybrid_a()

Can we do better?

The following algorithm shows an in-transit reduction on a hypercube.

The algorithm is a hybrid between an all-to-one reduction and an all-to-all reduction.
hybrid_a()

p=8,
k=2
hybrid_a()
hybrid_a()
hybrid_a()
hybrid_a()
hybrid_a()
hybrid_a()
hybrid_a()
hybrid_a()

Cost Analysis:

- Let \( m \) be the size of one partition.
- **All-to-one phase:**
  - at each step \( i \), each node sends message size \( km \)
  - there are \( \log_2(p) - \log_2(k) \) of these steps
- **All-to-all phase:**
  - at each step \( i \), each node sends message size \( \frac{km}{2^i} \)
  - there are \( \log_2(k) \) of these steps
hybrid_a()

Cost Analysis:

\[
\sum_{i=1}^{\log_2\left(\frac{p}{k}\right)} (t_s + t_w km) + \sum_{i=1}^{\log_2 k} (t_s + t_w \frac{km}{2^i})
\]

\[= \log_2\left(\frac{p}{k}\right)(t_s + t_w km) + t_s \log_2 k + t_w m(k - 1)\]
hybrid_a()

hybrid_a(), k=4, sp=100M, up=1M

- network_reduce_time:
- initial_reduce_time:
- initial_sort_time:
- pair_generation_time:
Comparison

MPI_Reduce, k=4, sp=100M, up=1M

hybrid_a, k=4, sp=100M, up=1M
Pathological Datasets: Sparse Keys

Suppose that instead of this:

Your key distribution looked like this:
Pathological Datasets: Sparse Keys

Consider the MPI_Reduce() operation that we must define for reducing two kvpair arrays:

```c
void op_func(int *in, int *inout, int *len, MPI_Datatype *type) {
    int i;
    in++;
    inout++;
    for (i = 1; i < *len; i += 2) {
        *inout = *in + *inout;
        in += 2;
        inout += 2;
    }
}
```
Pathological Datasets: Sparse Keys

When reducing two arrays, MPI_Reduce() expects one input array to function as an output array.

This means we cannot reduce:

```
1,(count)  3,(count)  5,(count)
+           +
2,(count)  4,(count)  6,(count)
```

Instead, we must reduce:

```
1,(count)  2,0  3,(count)  4,0  5,(count)  6,0
+           +
1,0  2,(count)  3,0  4,(count)  5,0  6,(count)
```
Pathological Datasets: Sparse Keys

MPI_Reduce, k=4, sp=100M, up=4M, pl=4

hybrid_a, k=4, sp=100M, up=4M, pl=4
Pathological Datasets: Poor Partitions

Suppose that instead of this:

Your key distribution looked like this:
Pathological Datasets: Poor Partitions

As before, MPI_Reduce() suffers performance degradation. Each partition must have size key_space/k.

However, with hybrid_a(), we can use samples from each node to compute intelligent partitions.
Pathological Datasets: Poor Partitions

From each node, select k-1 local samples (after local reduction). For k=2:

1 2 3 80

1 3 5 6

2 3 4 5

1 2 4 90

Sort

Pick k-1 global samples

3 4 5 4

3 4 4 5

1 1 1 2 2 2 3 3 3 3 4 4 5 5 5 6 80 90
Pathological Datasets: Poor Partitions

Same reduction algorithm, first without and then with sample partitioning.

hybrid_a, k=4, sp=400M, up=512M, naive

hybrid_a, k=4, sp=400M, up=512M, sample
Miscellaneous
Sources

- Gufler et. al. "Handling Data Skew in Mapreduce." Technische Universität Munchen, Munchen, Germany. Free University of Bozen-Bolzano, Bolzano, Italy.