SE ZG522: Big Data Systems

08 Design Strategy Hadoop map/reduce

Dr. Kamlesh Tiwari
Assistant Professor, Department of CSIS,
BITS Pilani, Pilani Campus, Rajasthan-333031 INDIA

April 03, 2022 ONLINE (WILP @ BITS-Pilani Jan-May 2022)

http://ktiwari.in/bds

Design Strategy: Divide-and-conquer for Parallel/Distributed Systems

- Divide if size of data is large
- Three parts, Divide, Conquer and Combine
- Some problems: Merge sort, Integer Multiplication, Defective Chessboard

[1] Merge sort

```
MERGE-SORT (A,p,r)
1  if p < r
2     q = ⌊(p + r)/2⌋
3     MERGE-SORT (A,p,q)
4     MERGE-SORT (A,q+1,r)
5     MERGE (A,p,q,r)
```

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq c \\ aT(n/b) + D(n) + C(n) & \text{otherwise} \end{cases}$$

```
MERGE(A, p, q, r)
1   n1 = q − p + 1
2   n2 = r − q
3   let L[1..n1 + 1] and R[1..n2 + 1]
    be new arrays
4   for i = 1 to n1
5       L[i] = A[p + i − 1]
6   for j = 1 to n2
7       R[j] = A[q + j]
8   L[n1 + 1] = ∞
9   R[n2 + 1] = ∞
10  i = 1
11  j = 1
12  for k = p to r
13      if L[i] ≤ R[j]
14          A[k] = L[i]
15          i = i + 1
16      else A[k] = R[j]
17          j = j + 1
```

Average case $T(n) = O(n \log n)$. Best and Worst?

[2] Integer Multiplication

- How do you multiply integers? How much time it takes?
- If $x = x_1 \times 10^{n/2} + x_0$
- Then $xy = x_1y_1.10^n + (x_1y_0 + x_0y_1).10^{n/2} + x_0y_0$

$$T(n) \leq 4T(n/2) + c.n$$

- $(x_1 + x_0) \times (y_1 + y_0) = x_1y_1 + x_0y_1 + x_1y_0 + x_0y_0$

Integer Multiplication: Karatsuba's Algorithm

Recursive Multiply parts

Algorithm 1: Rec-Mul (x,y)

1 Let $x = x_1 \times 10^{n/2} + x_0$
2 Let $y = y_1 \times 10^{n/2} + y_0$
3 p = Rec-Mul $(x_1 + x_0, y_1 + y_0)$
4 x_1y_1 = Rec-Mul (x_1, y_1)
5 x_0y_0 = Rec-Mul (x_0, y_0)
6 **return** $x_1y_1 \times 10^n + (p - x_1y_1 - x_0y_0) \times 10^{n/2} + x_0y_0$

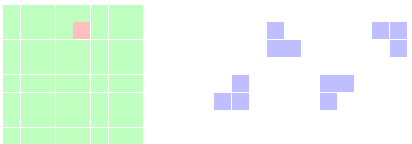
Time complexity

$T(n) \leq 3T(n/2) + c.n$

$O(n^{\log_2 3}) = O(n^{1.59})$

[3] Defective Chessboard

- Consider a chessboard of size $2^k \times 2^k$ where one cell is defective. Your task to cover it using a triomino.



Obviously

- Triomino cannot cover the defected one
- Triomino should not overlap
- Triomino must cover all other squares

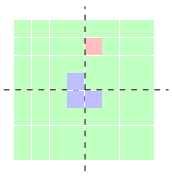
Note: $4^k - 1$ is divisible by 3

Defective Chessboard

Divide and conquer

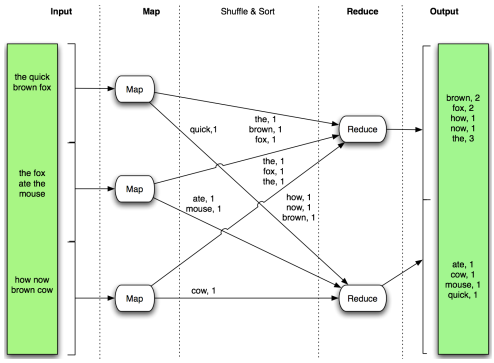
1. **Divide:** in smaller size instances.
2. **Recursive step:** use same framework till it is trivially solvable
3. **Conquer:** combine solutions of smaller instances to get overall solution

- $2^k \times 2^k$ size board is divided in four $2^{k-1} \times 2^{k-1}$ size board
  - Let $T(n)$ be time to tile $2^k \times 2^k$ board



$$\begin{aligned} T(n) &= t_d + 4T(n/2) + t_c = 4T(n/2) + c \\ &= c + 4c + ... + 4^{n-2}c + 4^{n-1}d \\ &= c\frac{4^{n-1}-1}{4-1} + 4^{n-1}d \\ &= \left(\frac{c}{12} + \frac{d}{4}\right) \times 4^n - \frac{c}{3} = \Theta(4^n) \end{aligned}$$

Map/Reduce framework as a Construct



map is a function that executes on each partition.
We have seen PK-Means and MR-DBSCAN

Sample MAP code

```
public static class Map extends Mapper<LongWritable,Text,Text,IntWritable>
{
    public void map(LongWritable key, Text value, Context context) throws
IOException,InterruptedException {
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
            value.set(tokenizer.nextToken());
            context.write(value, new IntWritable(1));
        }
```


Sample Reduce code

```
public static class Reduce extends Reducer<Text,IntWritable,Text,IntWritable> {
    public void reduce(Text key, Iterable<IntWritable> values,Context context)
        throws IOException,InterruptedException {
            int sum=0;
            for(IntWritable x: values)
            {
                sum+=x.get();
            }
            context.write(key, new IntWritable(sum));
        }
}
```


Sample Main code

```
Configuration conf= new Configuration();
Job job = new Job(conf,"My Word Count Program");
job.setJarByClass(WordCount.class);
job.setMapperClass(Map.class);
job.setReducerClass(Reduce.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);
Path outputPath = new Path(args[1]);
//Configuring the input/output path from the filesystem into the job
FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));
```


Hadoop: Introduction, Architecture, and Map-reduce Programming on Hadoop

- Open-source sw framework fr storing and processing Big Data in a distributed manner on large cluster of community hardware.
- Will traget three challenges
  1. Storing very large amount of data
     blocks in dataNode (specify size) with replication. NameNode stored what is stored where. Horizontal scaling
  2. Hetrogeneous data
     Different schema can be applied at the time of accessing.
  3. Maintain processing speed
     By sending processing to data not the otherwise
- Reliable, Economical (using community hw), Scalabile
- Master slave. Job and task tracker
- YARN is Yet another Resource Negotiator
- V1: map/reduce + HDFS
- V2: map/reduce + YARN + HDFS

Thank You!

Thank you very much for your attention!

Queries ?