Screwdriver :並列忘れがちなMapReduceプログラミングフレームワーク

Screwdriver: A Parallelism-Oblivious MapReduce Programming Framework

劉 雨　胡 振江
国立情報学研究所

GTA Programming Model

Lot's of problems can be naturally specified in term of:

Generate : generates all possible solution candidates.
Test: filters the intermediate data.
Aggregate : Sums valid intermediate data.

A GTA fusion framework can automatically create efficient scalable MapReduce programs form these specifications [ESOP 2012, Emoto et al].

High Level GTA Programming Framework on MapReduce

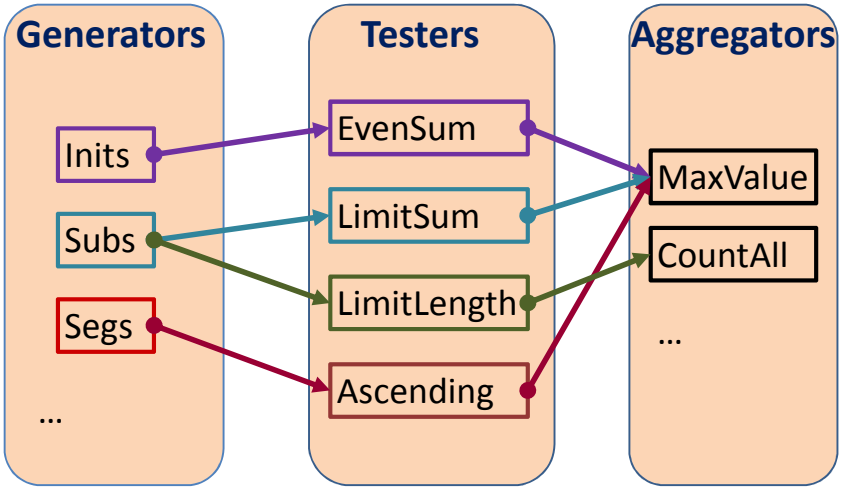
We propose a high level programming framework for users to simply write GTA programs which can be deployed on large Hadoop cluster.

The Schematic Diagram

Light-weight Programming interfaces	*Filter-embedding fusion*	*Efficient MR programs*
Users can choose necessary build-blocks provided by our framework	**The specifications can be optimized to efficient list homomorphisms**	**Fully parallel and scalable Hadoop jobs are generated automatically**

Programming using GTA - Building Blocks

- Even-Maximum-Prefix-Sum problem
- Knapsack problem
- Maximum Ascending Segments problem
- Count n-length subs problem



Programming with GTA is just to define the specification by using GTA build-blocks

An Example

The GTA Implementation Knapsack problem

Compute the maximum total value of selected items under a limitation of total weight

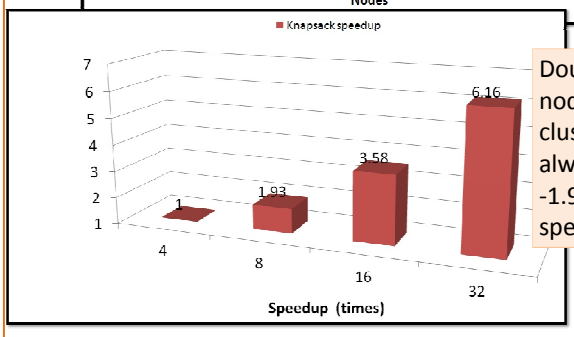
```
1 import ...
2 public class Knapsack extends MapReducerCreator {
3
4   private final Generator efftGen;
5
6   @Override
7   public Generator  createMapReducer() {
8     return efftGen;
9   }
10
11  public Knapsack() {
12
13    //test: limited sum of value
14    LimitedSumTest<KnapsackItem> test = new LimitedSumTest<KnapsackItem>(100) {
15      @Override
16      public IntWritable element(KnapsackItem elem) {
17        return new IntWritable(Math.min(maxSum, elem.getWeight()));
18      }
19      @Override
20      public Boolean postprocess(IntWritable val) {
21        return val.get() < maxSum;
22      }
23    };
24
25    //generator: gen all possiable candidates
26    SublistGenerator<KnapsackItem> generatoro =
27              new SublistGenerator<KnapsackItem>(new [illegible]);
28
29    //GTA fusion
30    efftGen = generatoro.filter(test).aggregate(new MaxSumAggregator<KnapsackItem>(){
31      @Override
32      public IntWritable singleton(KnapsackItem itm) {
33        return new IntWritable(itm.getVal());
34      }
35    });
36  }
37 }
```

test

generator

GTA fusion

Evaluation on Hadoop Clusters



Double the nodes of cluster can always gain 1.7 -1.9 times speedup

Conclusions

- **Performance:** Good speedup and scalability
- **Programmability:** Clear and Light-weight interface
- **Practicability:** Various problems have been resolved

Reference

Kento Emoto, Sebastian Fischer and Zhenjiang Hu. ***Generate, Test, and Aggregate --A Calculation-based Framework for Systematic Parallel Programming with MapReduce***. *22nd European Symposium on Programming (ESOP 2012), Tallinn, Estonia, March 24 - April 1, 2012*

NII
連絡先： 胡 振江(Zhenjiang HU)／東京都 千代田区 一ツ橋 国立情報学研究所 教授
TEL : 03-4212-2530　FAX : 03-4212-2533　Email : hu@nii.ac.jp