高レベル並列プログラミング

インターネット時代のビッグ・データは増加速い、処理難しい

For internet-scale data, traditional infrastructures and programming paradigms are no more applicable. New frameworks like MapReduce are developed and studied for handle PB-level data.

But, programming with MapReduce is in low-level thus difficult!

MapReduceに基づく、高レベル並列プログラミング

• Users write high-level (simpler) programs;
• Our framework transform them to MapReduce programs which can be deployed on large 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

例：GTAアルゴリズム

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

Generators

Testers

Aggregators

Inits

Subs

Segs

EvenSum

LimitSum

LimitLength

Ascending

MaxValue

CountAll

…

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

実際の例

Knapsack 問題の計算

A few lines of code, but scalable and efficient, using our framework:

```scala
//GTA+
def knapsack(contex: SparkContext, x: spark.RDD[DS.KnapsackItem]): = {
val allSelects = new AllSelects[KnapsackItem] //generator
val withLimit_100 = new WeightLimit(100) // tester
val gta = generate(allSelects).
filter ((x, (0,Limit)) => aggregate (x,`:perValue) .
val rst = gta.postProcess(x,map (gta.f(_), get).reduce (gta. combine(_ _))))
```

Using Hadoop-APIs maybe need 100s lines

高機能とスケーラビリティを持つ

Get near linear speedup be increasing CPUs

Double the nodes of cluster can gain almost twice speedup

結論

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

Reference