Hierarchical Agglomerative Clustering

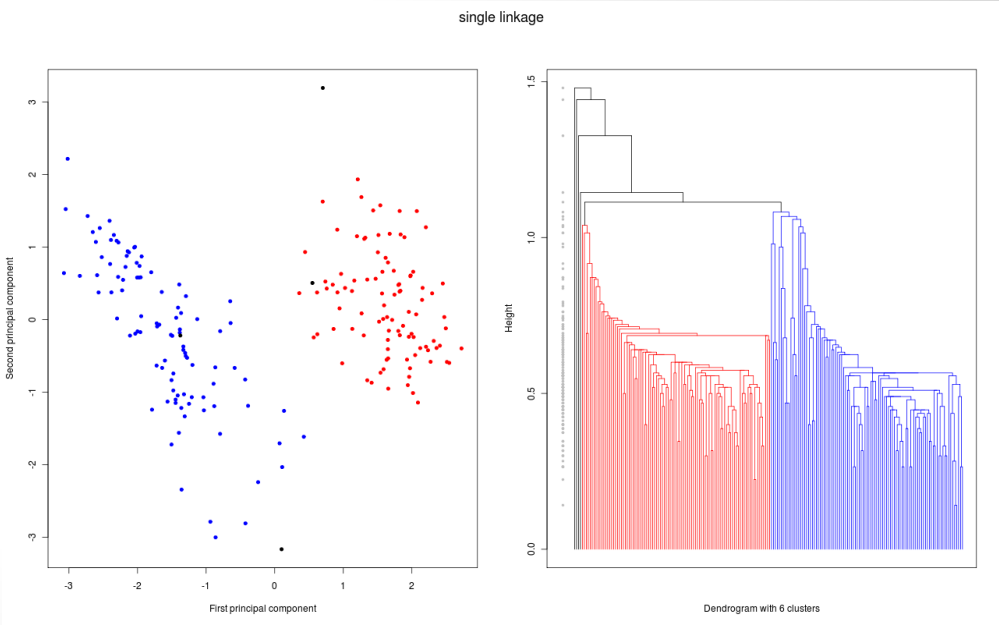
Sebastian Kunert, Bastian Köcher,
Sascha Wolke, Tobias Brandt

Algorithm

- Clustering of datapoints
- agglomerative = each datapoint starts in a cluster
- each iteration, the two nearest clusters are merged

Algorithm (2)

- Input:
  Datapoints,
  distance metric
- Output:
  History of cluster
  merges



Stratosphere Implementation

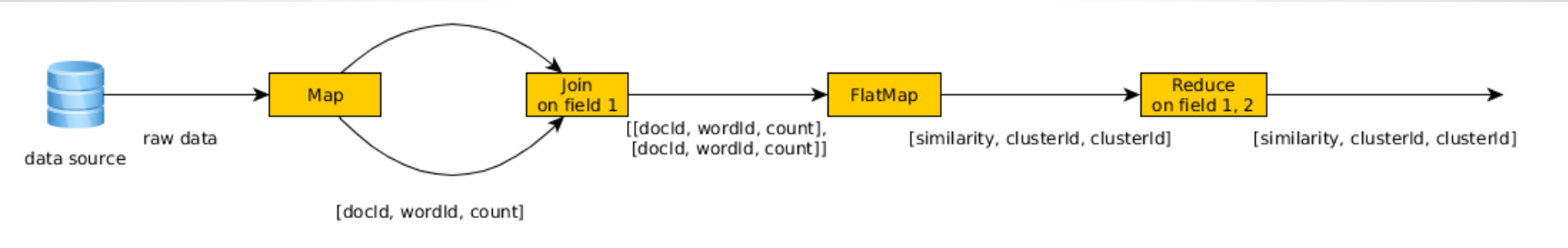
1. Compute initial distances (similarities) between all clusters
2. Find minimum distance
3. Merge clusterpair with minimum distance
4. Jump back to step 2 until required number of clusters is reached

Initial Similarity Computation

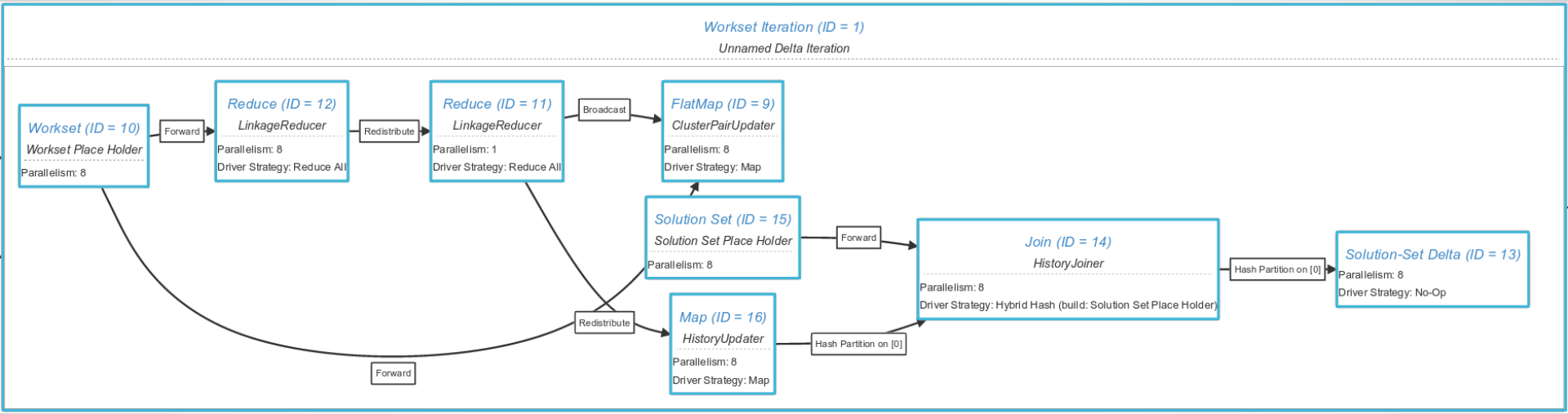
- Bag-of-Words dataset: `docId wordId count`
- Similarity Metric[1]: $sim(d_i, d_j) = \sum_{t \in V} w_{t,d_i} \cdot w_{t,d_j}$
  - words that appear only in one document are ignored
  - words that appear often in both are weighted heavily

[1] http://www.umiacs.umd.edu/~jimmylin/publications/Elsayed_etal_ACL2008_short.pdf

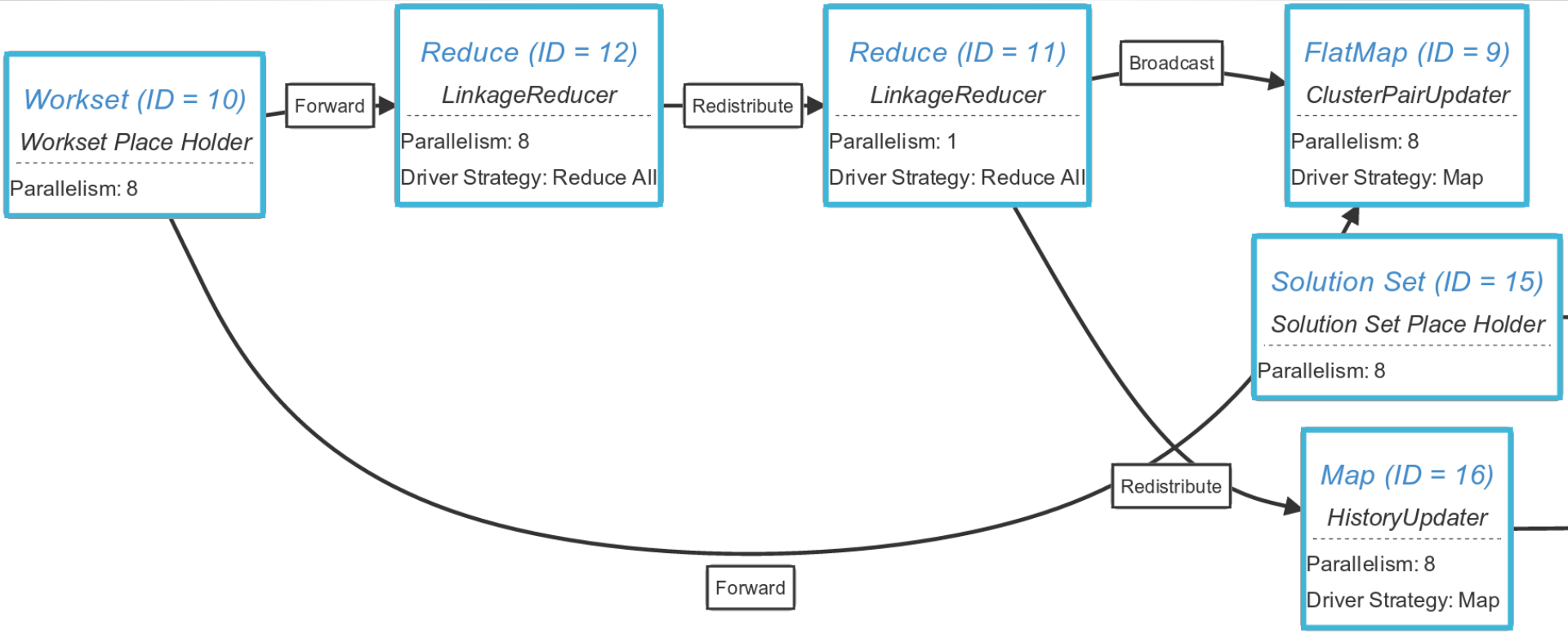
Initial Similarity Computation (2)



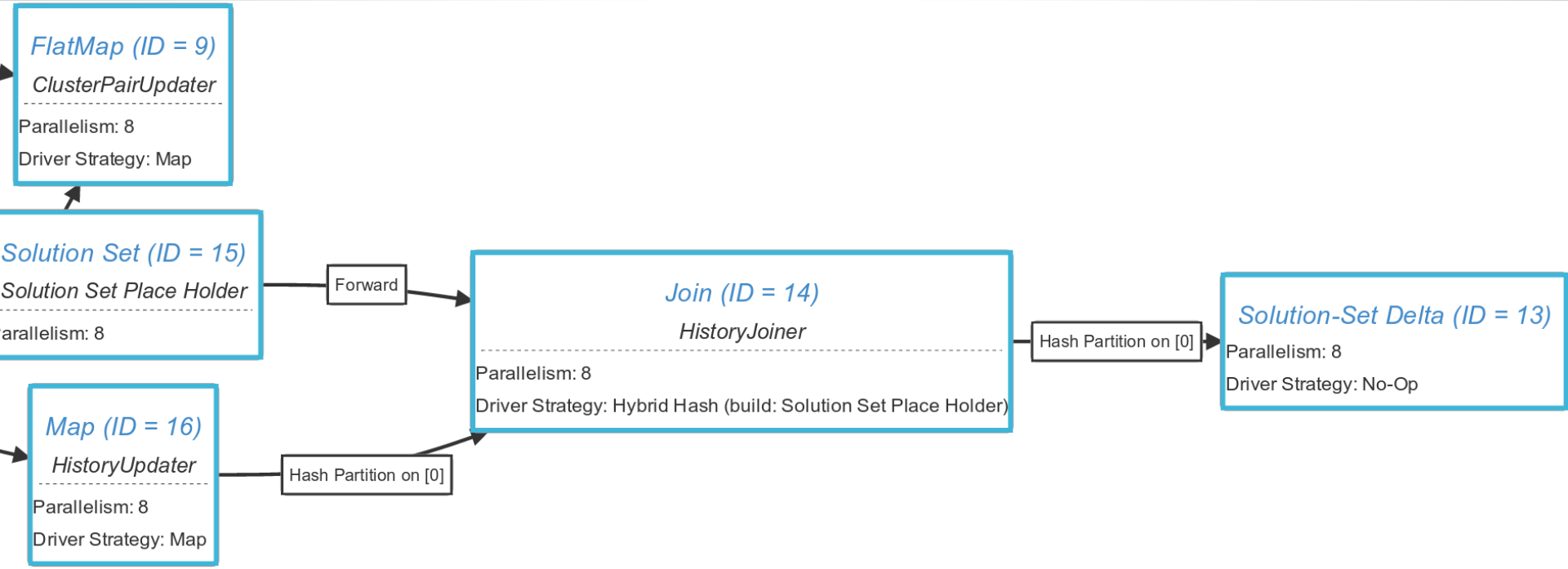
Iteration


Workset Iteration (ID = 1)
Unnamed Delta Iteration
Workset (ID = 10)
Workset Place Holder
Parallelism: 8
Forward
Reduce (ID = 12)
LinkageReducer
Parallelism: 8
Driver Strategy: Reduce All
Redistribute
Reduce (ID = 11)
LinkageReducer
Parallelism: 1
Driver Strategy: Reduce All
Broadcast
FlatMap (ID = 9)
ClusterPairUpdater
Parallelism: 8
Driver Strategy: Map
Solution Set (ID = 15)
Solution Set Place Holder
Parallelism: 8
Forward
Join (ID = 14)
HistoryJoiner
Parallelism: 8
Driver Strategy: Hybrid Hash (build: Solution Set Place Holder)
Hash Partition on [0]
Solution-Set Delta (ID = 13)
Parallelism: 8
Driver Strategy: No-Op
Redistribute
Map (ID = 16)
HistoryUpdater
Parallelism: 8
Driver Strategy: Map
Hash Partition on [0]
Forward

Iteration


Workset (ID = 10)
Workset Place Holder
Parallelism: 8
Forward
Reduce (ID = 12)
LinkageReducer
Parallelism: 8
Driver Strategy: Reduce All
Redistribute
Reduce (ID = 11)
LinkageReducer
Parallelism: 1
Driver Strategy: Reduce All
Broadcast
FlatMap (ID = 9)
ClusterPairUpdater
Parallelism: 8
Driver Strategy: Map
Solution Set (ID = 15)
Solution Set Place Holder
Parallelism: 8
Redistribute
Map (ID = 16)
HistoryUpdater
Parallelism: 8
Driver Strategy: Map
Forward

Iteration


FlatMap (ID = 9)
ClusterPairUpdater
Parallelism: 8
Driver Strategy: Map
Solution Set (ID = 15)
Solution Set Place Holder
arallelism: 8
Forward
Join (ID = 14)
HistoryJoiner
Parallelism: 8
Driver Strategy: Hybrid Hash (build: Solution Set Place Holder)
Hash Partition on [0]
Solution-Set Delta (ID = 13)
Parallelism: 8
Driver Strategy: No-Op
Map (ID = 16)
HistoryUpdater
Parallelism: 8
Driver Strategy: Map
Hash Partition on [0]

Scalability of HAC

- overall scalability is not great
- computation of similarity takes a while
  - we only use the upper triangle: $\frac{n * (n-1)}{2}$ pairs
  - number of pairs rises exponentially with documents
  - but is done only once at initialization phase
- in each iteration we have to find the minimum/maximum similarity
  → ungrouped reduce

Spark Implementation - Initialization

```
// docID, termID, term count
val docTermCounts = sc.textFile(inputFile).map(line => {...})
// initialize documents with cluster id = document id as (clusterID, docID) tuples
var documents = docTermCounts.map(_.docID).distinct.map(docID => (docID, docID))
// calculate similarity matrix with ((firstDocID, secondDocID), similarity) tuples
val termCount = docTermCounts.map(dtc => (dtc.termID, (dtc.docID, dtc.count))).groupByKey()
var similarities = termCount.flatMap { case (termID, counts) =>
    counts.flatMap { case (leftClusterID, leftTermCount) =>
      counts.flatMap { case (rightClusterID, rightTermCount) =>
          if (leftClusterID < rightClusterID)
              Some((leftClusterID, rightClusterID), leftTermCount*rightTermCount)
          else
              None
      }
    }
}.reduceByKey(_+_)
```

Works like a cross

Only process one triangle

Spark Implementation - Iteration

```
for(i <- 1 to iterationCount) {
   // similarity: ((cluster1, cluster2), similarity) tuples
   val clusterToMerge = similarities.reduce((a,b) => if (a._2 > b._2) a else b)

   // extract cluster id's
   val (removedClusterID, mergedClusterID) = clusterToMerge._1

   // move documents into new clusters
   documents = documents.map { case(clusterID, docID) =>
      if (clusterID == removedClusterID)
         (mergedClusterID, docID)
      else
         (clusterID, docID)
   }
   // ...



}}
```

Spark Implementation - Iteration

```
for(i <- 1 to iterationCount) {
  // ...
  similarities = similarities.flatMap { case ((firstClusterID, secondClusterID), similarity) =>
    if (firstClusterID == removedClusterID && secondClusterID == mergedClusterID) {
      None
    } else if (firstClusterID == removedClusterID) {
      if (mergedClusterID < secondClusterID)
        Some((mergedClusterID, secondClusterID), similarity)
      else
        Some((secondClusterID, mergedClusterID), similarity)
    } else if (secondClusterID == removedClusterID) {
      if (firstClusterID < mergedClusterID)
        Some((firstClusterID, mergedClusterID), similarity)
      else
        Some((mergedClusterID, firstClusterID), similarity)
    } else
      Some((firstClusterID, secondClusterID), similarity)
  }.reduceByKey(math.max)
}}
```

Drop pair

Replace removed cluster ID with cluster1' < cluster2'

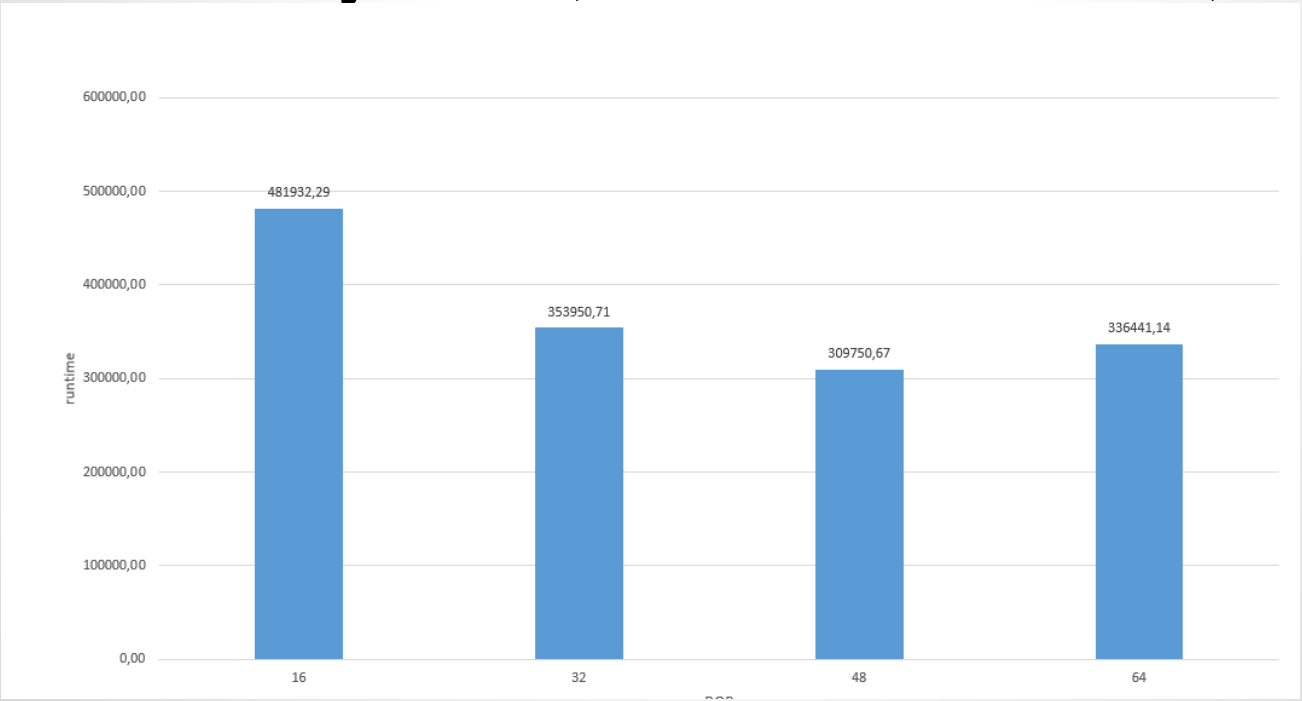
Don't change pair

Status

- stratosphere implementation (done)
- spark implementation scala (done)
- spark implementation java (done)
- stratosphere on cluster (done for small data set)
- spark on cluster (tbd)

HAC on Cluster

- dataset: nytimes, 2000 documents, 2.4mb



Problems during development

1. HAC is not meant for parallel execution
2. Bugs in stratosphere
3. Missing features in stratosphere

Bugs

- [922] coGroup on solutionset lead to a NullPointerException
- [940] Missing exception lead to incorrect usage of a coGroup in a Deltaiteration
- [941] Deadlock after using bigger dataset
- [1000] Job failed after some time because of an IndexOutOfBoundException

- Also added 2 ideas to simplify the usage of stratosphere

Future Improvements

- merge multiple clusters in one iteration
- move away more from the standard algorithm, more approximation
- more spark tests and optimizations on cluster -> was not finished due to time limitations

Questions?

Thank you for your attention!