Incoop: MapReduce for Incremental Computations
by Bhatotia et al
What is Incoop?

- Hadoop based framework
- Designed for improved efficiency of incremental programs
- Developed at the Max Plank Institute by Bhatotia et al.
Why Incoop?
Why run incremental computation on Incoop?

- Lots of applications are incremental
  - Machine Learning, wc over a range of docs etc

- Easy to write, input = Hadoop programs

- Great speedups
What differs Incoop from Hadoop?

- Incremental HDFS
- Incremental map and incremental reduce through contraction phase
- Memoization-aware scheduler
HDFS recap

- Large, fixed sized chunks - 64MB
- Append only filesystem
- Serial reads and writes
What’s bad about HDFS?

- Even small changes to input data results in unstable partitioning!
- This makes it difficult to reuse results
The problem with HDFS
Partitioning

HDFS

Mapper
Mapper
Mapper

Input file
Input file
Input file
The problem with HDFS
Partitioning
The problem with HDFS

Partitioning

HDFS

Input file

Input file

Input file

Mapper

Mapper

Mapper
Incremental HDFS

- Splits input data based on content
- Variable length chunk sizes
- Done at the input creation phase
- Follows the HDFS API
Solution with incremental HDFS

INC-HDFS

Mapper

Mapper

Mapper
Solution with incremental HDFS
Solution with incremental HDFS
What differs Incoop from Hadoop?

- Incremental HDFS
- Incremental map/reduce and contraction phase
- Memoization-aware scheduler
Incremental Map Phase

- Persistently stores result between iterations
- Creates a reference to the result in the memoization server (via hashing)
- Later iterations fetches results pointed to by the memoization server
Incremental Map Phase

(a) First run

(b) Incremental run
Incremental Reduce phase

● More challenging than the Map Phase

● Coarse grained memoization
  ○ Reducers copies map input only if result not already computed

● Fine-grained memoization
  ○ Combiners
What are combiners?

- A step between mappers and reducers

- Traditionally used to reduce the bandwidth between mappers and reducers

- Used in incoop to split reduce tasks and allow for better memoization
Incremental Reduce phase

(a) First run

(b) Contraction

(c) Incremental run
What differs Incoop from Hadoop?

- Incremental HDFS
- Incremental map/reduce and contraction phase
- Memoization-aware scheduler
Memoization Scheduling

- Built using memcached
- Per node work queue for good use of data locality and memoization
- Work stealing
Results - incremental runs

![Graph showing speedup over incremental changes for various data mining tasks: WordCount, BiCount, Co-Matrix, K-Means, and KNN. The x-axis represents incremental changes (%), and the y-axis represents speedup (w.r.t. Hadoop). The graph illustrates how speedup decreases with increasing incremental changes.](image-url)
Results - Scheduler

![Bar chart showing runtime comparisons for K-Means, WordCount, KNN, CoMatrix, and BiCount applications between Hadoop and Incoop schedulers. The x-axis represents the applications, and the y-axis represents runtime (Hadoop normalized = 1). The chart indicates that the Incoop scheduler generally performs better than the Hadoop scheduler.]
Results - Overheads

![Performance overhead for the first job run](image)

(a) Performance overhead for the first job run
Results - Overheads

(b) Space overhead
Criticisms

- Lack of comparison against other frameworks
- How were the percentual incremental changes generated?
- Garbage collection is pretty naïve. Odd-even runtime workloads sees no memoization.
- How realistic are the incremental results for real world workloads wrt Inc-HDFS?
Questions?