Data Aware Caching for Big-Data Applications Using MapReduce

Yaxiong Zhao, Jie Wu
Outline

• MapReduce and big-data
• Cache description
• Protocol
  – Relationship between job types and cache organization
  – Cache item submission
  – Lifetime management of cache item
  – Cache request and reply
• Experiment results
• Conclusion
Outline

• MapReduce and big-data
• Cache description
• Protocol
  – Cache request and reply
  – Lifetime management of cache item
• Experiment results
• Conclusion
MapReduce

• MR is a simple model of distributed parallel computing
  – Capture the data parallelism and serialization in Map, Reduce, and the intermediate phases
  – Versatile and robust

• Big-data refers to the applications that work on unconventionally large amount of data
  – Large corpse of data
  – New data is generated at high rate
  – Summarizing business insights requires fast processing and low cost
Outline

• MapReduce and big-data
• Cache description
• Protocol
  – Cache request and reply
  – Lifetime management of cache item
• Experiment results
• Conclusion
Why caching?

• Big-data requires small insights
  – A small amount of results are obtained from a huge amount of input
  – Repeating computations are expensive
  – Many processing are repetitive
    • Sorting, a sorted data split is obtained in a global sorting
    • Summary, the summary of each data split is computed
  – Processing results of data splits are generally small
    • Easy to store without too much cost
    • Some results may be large in size but save a lot of computation: sorting, data transformation.

• Caching is a great aid to the efficiency of MR
Map phase cache description

• Source data split ID + operation
  – Data file is stored in HDFS
  – Each data split is a fixed size chunk
  – Map phase operations are performed on each data split
  – The users are allowed to define their own operations

• MR’s java implementation provides such interface to obtain the data split ID

• Operations need to be defined by users
Reduce phase cache description

- Unlike the Map phase, reduce phase needs the partition method to determine what data records are processed together
- Each data record has a key
  - Partition method, usually certain Hash function, computes the reducers the data record should go
  - We are not interested in what reducer the data record goes
  - Instead, we want to know what data records goes to the “same” reducer
    - We want to cache the results of the processed data from Map phase that are processed at one reducer
    - These are the results that could be reused
Outline

• MapReduce and big-data
• Cache description
• Protocol
  – Cache request and reply
  – Lifetime management of cache item
• Experiment results
• Conclusion
Cache request and reply
Lifetime Management

• We only consider storage cost

• Fixed Storage Quota
  – A fixed fraction of the total space used for input data is used for storing caches

• Optimal Utility
  – Consider the storage expense with the saved computation time
  – This usually can be obtained from analyzing historical jobs
    • Storage expenses can be obtained from public Cloud pricing model
Outline

• MapReduce and big-data
• Cache description
• Protocol
  – Cache request and reply
  – Lifetime management of cache item
• Experiment results
• Conclusion
Job completion time

![Graph showing job completion time comparison between Dache and Hadoop.]
Conclusion

• Dache requires only a slight modification in the input format and task management of the MapReduce framework

• Testbed experiments show that it can eliminate redundant computations and saves computation time