Map Reduce and Design Patterns
Lecture 7

Fang Yu

Software Security Lab.
Department of Management Information Systems
College of Commerce, National Chengchi University
http://soslab.nccu.edu.tw

Cloud Computation, April 28, 2015

Input and Output Patterns

Customizing Input and Output in Hadoop

- Input: generating data, external source input, and partition pruning
- Output: external source output

InputFormat

Hadoop relies on the input format of the job to do three things

- Validate the input configuration for the job
- Split the input blocks and files into logical chunks (assigned to a map task)
- Create `RecordReader` implementation to be used to create key/value pairs that are sent one by one to the mapper

OutputFormat

Hadoop relies on the output format of the job for two main tasks

- Validate the output configuration for the job
- Create `RecordWriter` implementation that will write the output of the job

Generating Data

Instead of loading data that comes from somewhere outside, it generates that data on the fly and in parallel.

- Fake the split
- Let `RecordReader` generate the data
- The number of records to create is pulled from the job configuration, and a random record is created using a simple random number generator

External Output Files

Writes data to a system outside of Hadoop and HDFS

- Skips storing data in a file system entirely and sends output key/value pairs directly where they belong
- Be sure the destination system can handle the parallel ingest it is bound to endure with all the open connections

External Input Files

Load data in parallel from a source that is not part of your MapReduce framework

- Implement the `InputSplit` to read inputs in parallel and generate chunks to mappers
- Implement the `RecordReader` to read key/value pairs

Partition Prunning

Dynamically load the data based on what is requested by the application

- By partitioning the data by a common value, you can look only where the data would exist.
- For example, if you are commonly analyzing data based on date ranges, partitioning your data by date will make it so you only need to load the data inside of that range.
- This should be handled transparently, so you can run the same MapReduce job over and over again, but over different data sets
- The input format determines where the data comes from based on a query, rather than passing in a set of files
- Partition pruning changes only the amount of data that is read by the MapReduce job, not the eventual outcome of the analytic