CS 378 – Big Data Programming

Lecture 25

Input and Output Patterns

Input and Output Patterns

- We've used `InputFormat`s and `OutputFormat`s
  - Text
    - `TextInputFormat`, `TextOutputFormat`
  - Avro
    - `AvroKeyInputFormat`, `AvroKeyOutputFormat`

- These formats determine how data is:
  - Read from disk for map input
  - Written to disk for reduce output

File Formats

- How do they work?
- How to implement custom `FileFormat`s?

- Hadoop classes we'll examine:
  - `RecordReader`
  - `InputFormat`
- Complementary files for output:
  - `RecordWriter` and `OutputFile`

InputFormat

- What does an `InputFormat` do?
  - Validate input configuration (is the data there?)
  - Split input blocks/files into logical chunks
    - Logical chunks are of type `InputSplit`
    - Each is assigned to a mapper
  - Create the `RecordReader` that generates key/value pairs from the `InputSplit`
- `RecordReader` also has to fix up records that span splits

InputFormat

- Common input formats extend `FileInputFormat`
  - The base class for all file-based `InputFormats`
- We've used `TextInputFormat`
- Some other derived classes
  - `CombineFileInputFormat`
  - `KeyValueTextInputFormat`
  - `SequenceFileInputFormat`
- An `InputFormat` defines a `RecordReader`

RecordReader

- A `RecordReader` implementation reads a specific data schema
  - Text
  - Avro
- `RecordReader`
  - Reads bytes from an `InputSplit`
  - Creates a `WritableComparable` key, and
  - A `Writable` value
- So how does `LineRecordReader` work?

InputFormat

- `TextInputFormat` uses `LineRecordReader`
  - Reads an input split to get the next input line (`'\n'`)
  - At the beginning of an input split, find first newline
  - Reads past the split boundary until it finds an end-of-line
  - Key returned: position in the input split
  - Value: the input line

- `KeyValueTextInputFormat`
  - How is it different from `TextInputFormat`?

RecordReader Methods

- `initalize()`
- `getCurrentKey(), getCurrentValue()`
- `nextkeyValue()`
- `getProgress()`
- `close()`

OutputFormat

- What does an `OutputFormat` do?
  - Validate output configuration
    - Gets info from JobContext, TaskAttemptContext
  - Create the `RecordWriter` that generates key/value pairs for output

- `RecordWriter` methods:
  - `write()`
  - `close()`

InputFormat

- We discussed `FileInputFormat`s
- Input can come from other sources
  - Database
  - Web service
- Key point for all input formats:
  - Mapper does not know the details
  - It just processes calls to map()

Generating Random Data

- Random data can be used for testing when real data does not yet exist

- We need a custom input format to generate random data
  - No actual input is read
  - The `RecordReader` will generate random input

Assignment 11

- Use a custom input format to generate random messages
  - Select from a given set of words
  - Select words randomly
  - Length of the message (in words) is random
- The job reading this input will
  - Do word count on the random text
  - Compute mean and standard deviation of message length

Assignment 11

- Some important points
- In the RecordReader
  - Try to limit small object allocation by reusing objects
- In your mapper, limit small object allocation
- Use a combiner to limit data transfer