CS 378 – Big Data Programming

Lecture 22

MetaPatterns
Review

• Assignment 9 – Filtering
  – Filter searcher sessions (Viewed Carfax report)
  – Biased random sample of:
    • Bouncer, browser, searcher

• Questions/issues:
  – Any mapper side filtering?
  – For searcher sessions: apply random filter first, or second?
  – Values passed via Configuration
  – Random output: how is this graded?
MetaPatterns

• Most big data processing will use multiple jobs

• “Data pipelines” are common
  – Multiple map-reduce jobs
  – Output of one job is input to the others
  – The output can be an end in itself

• Why are multiple jobs required?
MetaPatterns

• We’ll discuss two classes of meta-patterns

• Job chaining
  – Multiple jobs solving a multi-stage problem
  – When processing cannot be done in one job
  – When one output is input to multiple jobs

• Job merging
  – Combining multiple activities into the same job
Job Chaining

• Since job chaining is common, some tools exist or are under development to help

• Examples:
  – Oozie
  – Azkaban
  – Luigi

• For more details, see:
  – http://www.slideshare.net/jcrobak/data-engineermeetup-201309
  – http://www.crobak.org/2012/07/workflow-engines-for-hadoop/
Workflow Issues

- Dependency structure/management
- Monitoring
- Error recovery
- Reporting
- Restart
Job Chaining

• Basic notion for job-chaining: dependency graph
  – Explicitly represented in tools
  – A concept that’s represented in the code we’ll consider

• Dependency graph:
  – Directed, acyclic graph (DAG) where:
  – Nodes represent data sets, and processing steps
  – Edges represent data flows (dependencies)
Job Chaining

• For single map-reduce jobs, we selected the number of mappers and reducers
  – Parallelism
  – Controlling the amount of data a reducer receives

• When chaining jobs, we must consider file sizes
  – They should be on the order of one block size or more
  – If output files are small, use CombineFileInputFormat
Job Chaining

- Consider what we do in `run()` of a single job
  - Define input info for a job
    - Input file location(s)
    - Input format type, key/value types
    - Mapper class(es)
  - Define output info for a job
    - Output location(s)
    - Output format type, key/value types
    - Reducer class
Job Chaining

• If we want our Java app to launch multiple jobs, what do we need to do?

• Create and configure multiple Job instances

• Connect output of one job to input of another job
  – How?

• Launch each job, wait for it to complete
  – How?
Job Methods

• So far we’ve used:
  – job.waitForCompletion()

• Other methods on Job:
  – isComplete()
  – isSuccessful()
  – killJob()
  – mapProgress()
  – reduceProgress()
  – submit()
  – getCounters()
Job Chaining

• Suppose output of first job is input to two jobs
  – These jobs can be run in parallel

• How would we launch two jobs to run in parallel?
• How would we monitor their progress?

• If another job combines the output of these two jobs
  – How would we know when to start this new job?
  – What possible scenarios do we need to consider/handle?
Job Chaining - Approaches

• Control/manage jobs explicitly in run() method

• Shell scripts

• JobControl and ControlledJob classes
  – Done in run() method
  – Some benefits, some restrictions
Job Chaining - Example

• Read log files and create sessions
  – Output into four category files

• In parallel:
  – Read bouncer sessions, count impression types
  – Read browser sessions, calculate mean, stdev listing views
  – Read searcher sessions, calculate VDP click through rate