Parallel S3 to HPCC Spray Module for Compressed Archives

RNET Technologies, Aug. 2015

The Parallel Spraying (ParSpray) module has been integrated with HPCC Systems CloudFormation (CF) template. The module provides functionality for directly spraying ZIP archives from Amazon S3 into an HPCC Systems cluster using parallel processes on HPCC slave nodes. The integration ensures that once the HPCC cluster is deployed on AWS, the user needs minimal actions to be able to use the module. All necessary software is installed through the CF setup process (this includes MPI for parallel execution on Slave nodes, among other software).

The following actions are needed to be performed in addition to the setup steps for HPCC Systems on AWS CF. Follow the below instructions in addition to the guidelines in EasyFastHPCCoAWS package (https://github.com/tlhumphrey2/EasyFastHPCCoAWS):

1. When creating the S3 bucket containing the source files for CloudFormation, include the following additional files:
    a. the RNET setup script: *rnet_parspray_setup_platform.sh* (available from s3://rnet-parspray-files with public access rights)
    b. the new JSON template, which runs the RNET setup script (available from s3://rnet-parspray-files) – this needs to be used instead of the original template.
    c. your AWS "credentials" file. This will be copied to .aws folders by the above (any other preferred way of getting the credentials in?).

    When the CloudFormation cluster is set up, the supplied RNET script will install necessary software and setup the environment and directories required for the ParSpray module to function.
2. Now open an ssh session into one of the slave nodes and switch to the *hpcc* user.
3. Go to folder ~/rnet-parspray-files (contents should already be copied from a public s3 bucket with the same name: s3://rnet-parspray-files)
4. To Spray a ZIP archive containing XML files directly from S3 to the deployed HPCC systems, run the ParSpray program using the following command from the above working directory. The output of this command includes several timing information regarding various steps of the work.

    mpiexec --hostfile ./machines.list -n <number of slave instances/nodes> --map-by node /usr/bin/java -cp ./external_jars/:./aws_jars/*:./ParSpray_src/bin ParSpray <S3 bucket name containing the ZIP archive of XML files> <Name of the ZIP archive file (S3 key) > [Spray options]*

    Current Options for XML are *rowtag=<XML Row Tag(default: tag)> encoding=<XML File format(default: utf8)>*

    Alternatively, the following simple script command can be run, which looks less complicated than the above MPI command (the same ParSpray options are passed here):

./SprayS3Zip.sh *<S3 bucket name containing the ZIP archive of XML files> <Name of the ZIP archive file (S3 key) > [Spray options]*

Currently the number of parallel processes should be equal to the number of slaves (i.e., the number of partitions for each file). So if we have 2 slave instances, we need to set *–n* option in MPI command equal to 2 and the files will be sprayed in two partitions, one on each instance. If multiple slaves are setup on an EC2 instance, and the file needs to be partitioned into more pieces on each Slave instance, *-n* option needs to be set to the total number of Slaves and the files will be partitioned into that many partitions.
For the alternative *SprayS3Zip.sh* script, the number of processes is internally calculated and is equal to the number of EC2 slave instances. If this is not desired (e.g., as in cases with multiple slaves per node), the MPI command should be used.

Example: Multiple ZIP archives are uploaded a to public S3 bucket s3://rnet-test1. It contains some sample XMLs provided by LexisNexis, as well as some generic XML files, plus a couple of simple CSV and JSON files. The following shows an example command spraying the sample ZIP archives in that S3 bucket on a 2-node HPCC deployment:

mpiexec --hostfile ./machines.list -n 2 --map-by node /usr/bin/java -cp ./external_jars/:./aws_jars/*:./ParSpray_src/bin ParSpray rnet-test1 sample_dataset.zip rowtag=item encoding=utf8*

OR simply

./SprayS3Zip.sh *rnet-test1 sample_dataset.zip rowtag=item encoding=utf8*

Other well-formed example ZIP archives, which was originally provided by John Holt (as a single XML file, replicated into a ZIP archive) are EMW_*.zip files, with XMLs sourced from 02535398_0001_8735 records.XML (an Elsevier abstracts sample dataset).

5. Now go to the ECL Watch to verify the logical files added to the Thor cluster.
6. A sample ECL script is also provided in the *~/rnet-parspray-files/my_ecl_sample_dataset.ecl*, which can be executed for the *sample_dataset.zip* archive.

Current TODOs and known limitations:

The parallel spraying module is being provided for initial evaluation. It has been tested on EC2 instances of type c3.xlarge on AWS (Amazon Linux AMI, kernel 3.14.35-28.38.amzn1.x86_64). The following items list some limitations/TODOs for the current version of the module. The limitations are currently being addressed. We also like to hear ideas on other needed functionality and how we can improve the features of the module:

- Only XML ZIP archives are natively supported at this point. Support for CSV and JSON formats exists, but currently through calling HPCC's *dfuplus* tool internally (after reading and extracting ZIP files in memory). The provided spray options should have the same format as accepted by *dfuplus* tool for the desired file format (as described in dfuplus help). If a ZIP file contains non-XML CSV or JSON record sets, they will be sprayed but the spraying may be slower than XML files, partially because unlike XML, the data are written to storage in order for *dfuplus* utility to access them for spraying. We use the nowait=1 option of *dfuplus* in order to make progress asynchronously. The original (uncompressed) data files (extracted from the ZIP archive by the module) will stay in /tmp, and the user may later delete them. If nowait option is not used, the temporary files will be copied to shared memory region /dev/shm and will be deleted automatically.
  Lack of native support for non-XML data formats will be fixed next. We need to add internal support for parsing/reading those formats and then utilize high-performance in-memory spraying.

  Example for non-XML spraying:
  mpiexec --hostfile ./machines.list -n 2 --map-by node /usr/bin/java -cp ./external_jars/:./aws_jars/*:./ParSpray_src/bin ParSpray rnet-test1 geo_person_data_20entry.json.zip maxrecordsize=8000*

  OR simply

  ./SprayS3Zip.sh *rnet-test1 geo_person_data_20entry.json.zip maxrecordsize=8000*

- XMLs need to have an outer encompassing tag for the records (this is the equivalent of "row tag" in the ECL Watch spray wizard). If this does not work as is desired by LexisNexis, we will be glad to hear more specific information as to what the target XML files will look like, so that we can support all desired forms of XML data.
- All XML files in an archive currently need to have the same record tag name (this will be fixed soon, if needed).
- All files in a ZIP archive will be sprayed, no custom subset spraying at this time (this will be supported next, if desired).
- Despraying is not completed yet (will be completed soon). We also need to know the desired despray functionality, whether it is desired to despray back to S3 in the form of a ZIP archive or in the form of uncompressed data files? Any input is appreciated.
- The module has not yet been tested with Elsevier entries of 100s/1000s of XML files (those John was referring to as the main use case). We will test those once we get access to such data.
- Other planned performance optimizations are also pending and will be performed after the above limitations are fixed.
- Integration with ECL Watch IDE is another feature that, if desired, will need coordination with LexisNexis team to be added.