PDB-Hadoop ReadMe

Jamie Al-Nasir

February 24, 2015

1 Introduction

PDB-Hadoop is a framework that facilitates the parallel execution of protein structure analysis tools to be carried out on the entire (or large subsets of) the Protein Databank (PDB) using the Apache Hadoop platform. Apache Hadoop is a software platform that allows for the processing of large scale datasets using clusters consisting of commodity hardware and is highly scalable.

PDB-Hadoop is designed so that structural Biologists can use the Hadoop platform without having to write the relatively complex Java code that Hadoop is implemented for. This framework is easily scalable and uses a *mapper* architecture that functions stand-alone or can be extended to include further *map-reduce* operations. This does away with the necessity for one to implement ones own map-reduce applications or re-write existing code for the *map-reduce* formalism, although the way in which PDB-Hadoop is implemented ensures this approach is still available for users wishing to exploit data aggregation properties of the *map-reduce* method.

PDB-Hadoop not only runs on local clusters but has also been implemented on commercial cloud providers such as Amazons Elastic MapReduce and Microsofts Azure HD. This has the added advantage of obviating the need to purchase specialised servers.

2 Acquisition of the PDB (Protein databank)

As of December 5th 2014 there are 105,383 structures deposited in the protein databank. Downloading the entire or sections of the PDB is commonly achieved using the Unix command *rsync* (which synchronises a local folder with a subfolder of the ftp archive of the PDB (rcsb.org) and can be carried out with the following command:

```
/usr/bin/rsync --progress -rlpt -v -z --delete --port=873
rsync.ebi.ac.uk::pub/databases/pdb/data/structures/divided/pdb/
PDB
```

This creates a local folder in the current directory named PDB and synchronises it with the entire PDB (in sub-divided folders). There is however a well-written script by Thomas Solomon circulating in the public domain which is useful for this purpose and allows further customisation of what is downloaded/synchronised.

3 Unzipping of the compressed .ent.gz PDB files

The files on the rcsb.org ftp archive are normally in compressed gzip format and require "unzipping" with *gunzip*. Assuming the files are in a folder entitled PDB, this can be achieved using the pre-written script that recursively processes the PDB folder as follows:

```
./pdb-gunzip.sh ./PDB
```

4 Scripts within this package and their function

pdb-gunzip.sh	given a folder, recursively extracts ent.gz files
pdb-to-line.sh	given a pdb file, converts this into a single line of text (used by pdb-build.sh)
pdb-build.sh	given a folder of pdb files (*.ent), builds a special single, large pdb file suitable for Hadoops hdfs
pdb-hadoop.sh	Hadoop mapper, this is run within a Hadoop job and handles execution of the user program on the pdb. This file requires customisation to set the path of the user program and temporary folder
Log-to-pdb.sh	Given a single concatenated PDB-Hadoop log file extracts individual PDB file output as separate files (these complement the input PDB files and are of the same name)

5 Conversion of a collection of PDB files into a single condensed file for Hadoop

Hadoop distributed file system (hdfs) is not optimised for a large number of small files that comprise the protein databank. This problem can be overcome by packaging the entirety of the PDB into a specialised, single pdb file (˜80gb, January 2014). This is achieved by running a script recursively that converts \n line delimiters to customised ˆ|line delimiters. The script is *pdb-to-line.sh* which is automatically run by *pdb-build.sh* script. The user runs pdb-build.sh script providing the directory of the thousands of PDB files and a target file to build, i.e. PDB-text-full.txt

```
./pdb-build.sh <PDBfolder> ./PDB-text-full.txt
```

6 Hadoop user Permissions

We suggest setting the permissions of the PDB-Hadoop folder to be accessible to all users for read and execute permissions so that Hadoops' YARN (Yet Another Resource Negotiator) may be able to execute the script within the Hadoop job.

```
chmod -R a+rX ˜/pdb-hadoop
```

7 Preparing the user program for execution – configuring the *pdb-hadoop.sh* script

In order configure the user program for parallel execution on Hadoop, one must edit the main PDB-Hadoop script *pdb-hadoop.sh* file to specify the path to the

user program and if necessary a preferred temporary folder for the job. The following lines can be amended:

```
#//---------------------------------------------------------------------------
# Use _LEGACY_PROGRAM_ to define the user program you wish to parallelise
# using Hadoop. Provide the filepath to the program and any command-line options
_LEGACY_PROGRAM_="/path/to/local/python/Artemis.py"
# Use _TEMP_FOLDER_ to define a readable folder for this program to work in
# ** Ensure there is a trailing /
_TEMP_FOLDER_="/tmp/"
# Use _MAX_PDB_SIZE_ to ignore processing of files > given size (in bytes)
_MAX_PDB_SIZE_=
#//---------------------------------------------------------------------------
```

PDB-Hadoop variables that must/can be set prior to executing a job:

1. `_LEGACY_PROGRAM_`

   [Required] to specify the path to the program to be execute in parallel fashion on Hadoop.

2. `_TEMP_FOLDER_`

   [Optional] to specify the temporary folder (default "/tmp"), which must be writable to YARN (i.e. the Hadoop user).

3. `_POST_PROC_PROGRAM_`

   [Optional] to specify the path to the users *post-processing* program that takes output from the execution of user program (defined above) and performs textual processing on the results.

4. `_MAX_PDB_SIZE_`

   [Optional] If set PDB-Hadoop ignores processing of files greater than specified size (in bytes)

7.1 Testing PDB-Hadoop with *wc -l*

For testing purposes one can specify a linux command such as *"wc -l"* as the

`_LEGACY_PROGRAM_`

in the main *pdb-hadoop.sh* script.

7.2 Prepending user program Job with Linux *timeout* command

In some cases it may be useful to limit the execution time of the user program and this can be achieved by prepending the user job with the Linux *timeout* command, for instance to specify a 4 minute time constraint for a user program:

```
_LEGACY_PROGRAM_="timeout 4m /path/to/local/python/Artemis.py"
```

A common reason to do this is to ensure that the user program Job only runs for a shorter period than Hadoops' default container time limit of 5 minutes. If required a time constraint for the container can be set within Hadoop and the Linux *timeout* command can be prepended to the job path to ensure the timeout set in Hadoop is not exceeded by the user program Job. NB: A user program job that overruns the Hadoop timeout will result in the containiner being terminated by Hadoops' YARN (Yet Another Resource Negotiator).

8 Executing the user program Job on Hadoop

**** Requirements:** Local file system ./pdb-text-full.txt (generated by pdb-build.sh) must be copied to Hadoop's hdfs, this can be achieved as follows:

```
hadoop fs -put /user/hduser/ /path/to/local/pdb-hadoop/pdb-text-full.txt
```

The job can then be executed by issuing the Hadoop command at the shell prompt:

```
hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
-Dmapred.sort.avoidance=0 \
-Dmapred.reduce.tasks=0 \
-D stream.non.zero.exit.status.is.failure=false \
-input  /user/hduser/pdb-text-full.txt
-output /user/hduser/pdb-legacy-output \
-mapper "/path/to/local/pdb-hadoop/pdb-hadoop.sh" \
```

9 Format of PDB-Hadoop job output - user program results

The pdb-hadoop job (executed by mapper: pdb-hadoop.sh) will execute the user program in a parallelised fashion on Hadoop and the output will be in the following format:

```
Extracted/writing file /tmp/pdb3zz1.ent
pdb3zz1.ent-0000000001  PDB Dihedral angle calculation program
pdb3zz1.ent-0000000002  By Jamie Al-Nasir
pdb3zz1.ent-0000000003  Royal Holloway dept. of Computer Science
pdb3zz1.ent-0000000004  http://jamie.al-nasir.com
pdb3zz1.ent-
pdb3zz1.ent-0000000005  Using PDB file: /tmp/pdb3zz1.ent
pdb3zz1.ent-
pdb3zz1.ent-0000000006  Computation for 712 residues(s) in the structure
pdb3zz1.ent-
pdb3zz1.ent-0000000007  Phi     Psi     Omega   Ch. Residue
pdb3zz1.ent-0000000008  0.00    -103.12 168.38   A  VAL1
pdb3zz1.ent-0000000009  -123.59 147.71  -178.71  A  VAL2
pdb3zz1.ent-0000000010  -66.19  143.08  -179.97  A  PRO3
pdb3zz1.ent-0000000011  -67.74  146.86  176.65   A  PRO4
pdb3zz1.ent-0000000012  -64.85  151.53  -179.54  A  ALA5
```

```
pdb3zz1.ent-0000000013  80.26   11.45   176.39   A  GLY6
pdb3zz1.ent-0000000014  -119.22 158.85  179.83   A  THR7
pdb3zz1.ent-0000000015  -51.76  -44.05  178.34   A  PRO8
pdb3zz1.ent-0000000016  -68.62  -34.23  179.35   A  TRP9
pdb3zz1.ent-0000000017  -63.71  -45.73  178.06   A  GLY10
pdb3zz1.ent-0000000018  -68.99  -38.64  175.79   A  THR11
pdb3zz1.ent-0000000019  -61.31  -38.36  177.48   A  ALA12
......
Extracted/writing file /tmp/pdb3zzs.ent
pdb3zzs.ent-0000000001  PDB Dihedral angle calculation program
pdb3zzs.ent-0000000002  By Jamie Al-Nasir
pdb3zzs.ent-0000000003  Royal Holloway dept. of Computer Science
pdb3zzs.ent-0000000004  http://jamie.al-nasir.com
pdb3zzs.ent-
pdb3zzs.ent-0000000005  Using PDB file: /tmp/pdb3zzs.ent
pdb3zzs.ent-
pdb3zzs.ent-0000000006  Computation for 585 residues(s) in the structure
pdb3zzs.ent-
pdb3zzs.ent-0000000007  Phi     Psi     Omega   Ch. Residue
pdb3zzs.ent-0000000008  0.00    150.04  176.62   A  SER7
pdb3zzs.ent-0000000009  -60.85  157.36  172.78   A  ASP8
pdb3zzs.ent-0000000010  -139.54 163.05  177.34   A  PHE9
pdb3zzs.ent-0000000011  -123.13 149.23  177.06   A  VAL10
pdb3zzs.ent-0000000012  -112.04 123.65  -179.08  A  VAL11
```

Each pdb file that is extracted from pdb-full.txt by the Hadoop mapper will have a line stating "Extracted/writing file /tmp/pdb3zz1.ent" followed by the results from the user program. Lines from the user program are prefixed with the original pdb filename and a line-number to facilitate sorting of the output by Hadoop. In this case the user program parallelised was dihedrals-64 a Linux command-line tool for computation of dihedral/torsional angles in peptide molecules.

10 Extracting the Legacy application results from Hadoop as a single file

Hadoop stores the output of the job on hdfs as a collection of text files of size dependent on the Hadoop cluster configuration. It is often most convenient to extract these concatenated as a single text file saved to the local file system and this can be achieved using Hadoops getmerge command as follows:

```
hadoop fs -getmerge /users/username/job-folder/ PDB-Hadoop-log.txt
```

11 Post-processing of user program results

PDB-Hadoop facilitates the processing of output data from the user program prior to deposition on HDFS. This allows the user to extract and distil information from the user job within the parallelised map process that is running on Hadoop.

12 Extracting individual files from the PDB-Hadoop generated concatenated log file

As previously explained the Hadoops *getmerge* command can be used to obtain a single concatenated log file. In some cases it is required to extract the output of a PDB-Hadoop job as individual files. For instance some legacy programs merely output a report for the input PDB file, while others output a new PDB file. Extraction of the newly generated PDB files (or user program reports) is achieved with the PDB-Hadoop *log-to-pdb.sh* script. This takes the single concatenated log-file input and splits this into individual PDB files/reports. In each case this output can be captured in individual files using the PDB-Hadoop *log-to-pdb.sh* script as follows:

```
./log-to-pdb.sh ./PDB-Hadoop-log.txt
```