CLAS12 Software Tutorial

Nathan Harrison
Jefferson Lab

CLAS Collaboration Meeting
June 14, 2017
Jefferson Lab

Outline

- Introduction and preliminary set-up (5 min)
- Software PSA (5 min)
- Simulations with GEMC (5 min)
- KPP raw data (5 min)
- Downloading and installing CLARA and COATJAVA (5 min)
- Decoding (5 min)
- Reconstruction (5 min)
- Analysis code (5 min)
- Machine Learning Demo – Mike Williams (~2 hours)

Introduction and preliminary set-up

- Instructions are written in blue

- Terminal commands are written in black, are indented, and start with a “> “ e.g.

```
    > echo “hello world”
```

- Green text means you should substitute the text with your own value

- Since there have been relatively few updates (from a user's point of view) since the tutorial at the March collaboration meeting (https://www.jlab.org/indico/event/201/), this tutorial will (hopefully) be fairly short and cover a few new things.

Introduction and preliminary set-up

(1) Open a terminal on your laptop and log into an ifarm CUE machine (requires two steps):

```
> ssh -Y username@login.jlab.org

> ssh ifarm
```

(2) This tutorial will assume you are using tcsh, to check your shell, do:

```
> echo $SHELL
/bin/tcsh
```

to switch to tcsh, do:

```
> chsh -s /bin/tcsh
```

It will also be assumed that your ~/.tcshrc file is empty; if this is not the case then your safest option is to comment everything out. If you made any changes in this step, log out and log in again.

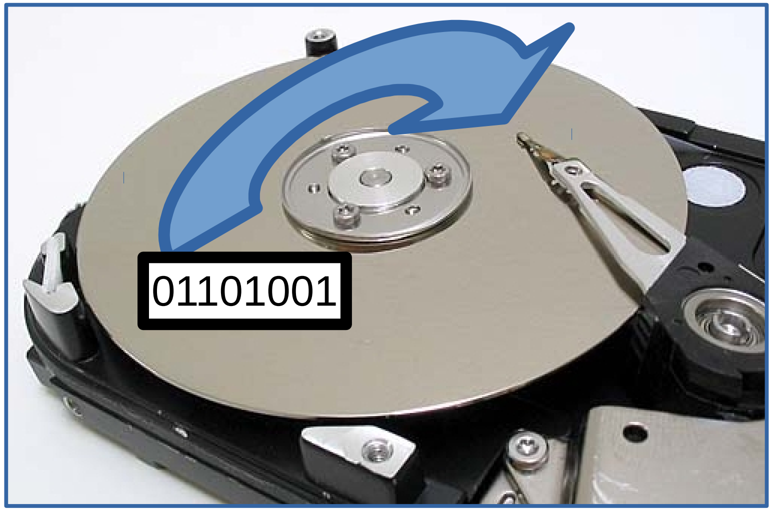
Introduction and preliminary set-up

(3) Select a location with sufficient disk space (e.g. /work or /volatile), copy the directory of ancillary files there, cd into that directory, and source the provided environment script:

```
> cd /volatile/clas12/username/

> cp -r /volatile/clas12/nathanh/demo_16jun17 .

> cd demo_16jun17

> source demo-env.csh
```

* see commands.txt for copying/pasting

Software PSA – Use buffered I/O and multi-threading whenever possible

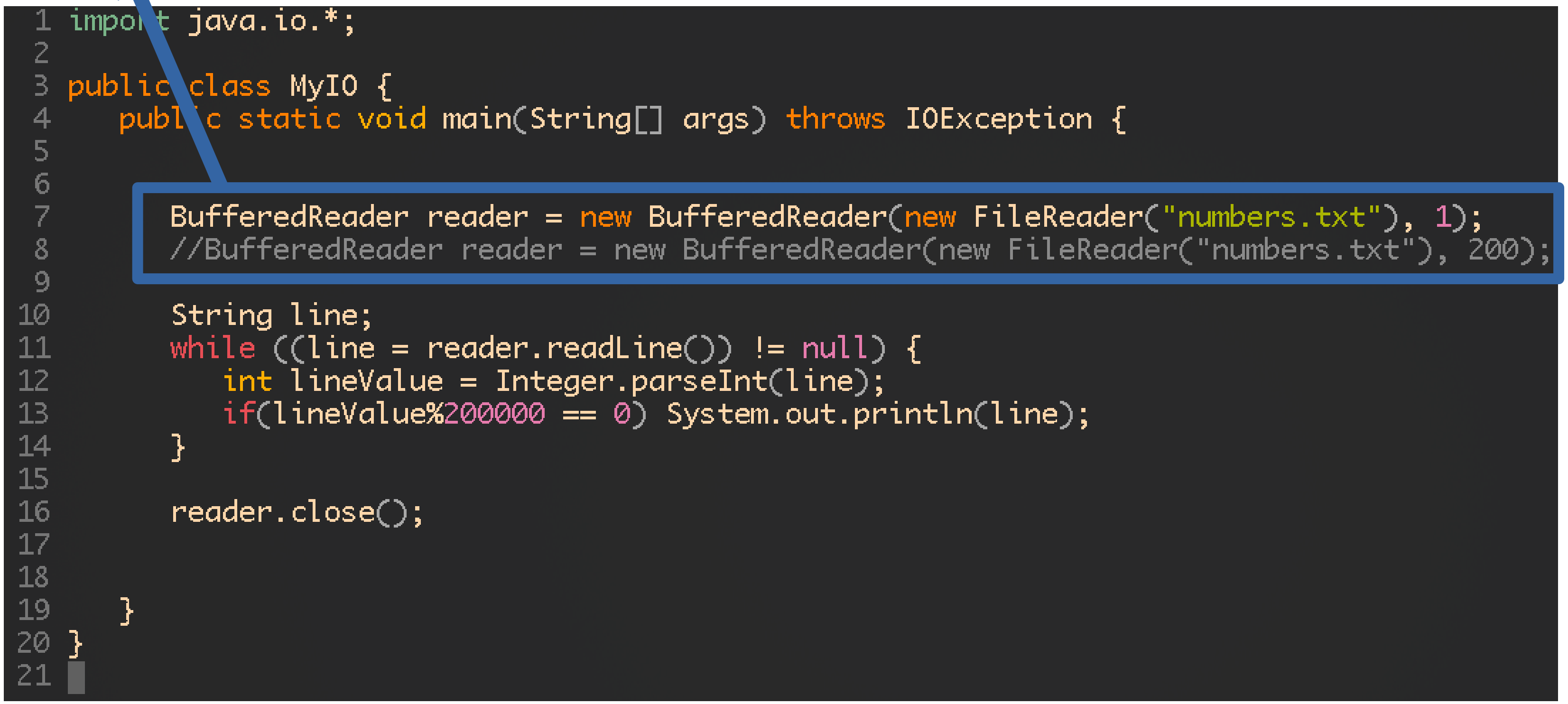


7200 RPMs => about 4 ms on average for the disk to spin to the necessary position to start reading the data.

(1) Run MyIO.java twice to see how buffered I/O can speed up your code:

```
> javac MyIO.java

> /usr/bin/time -p java MyIO
```

try both of these lines!



```
import java.io.*;

public class MyIO {
   public static void main(String[] args) throws IOException {


      BufferedReader reader = new BufferedReader(new FileReader("numbers.txt"), 1);
      //BufferedReader reader = new BufferedReader(new FileReader("numbers.txt"), 200);

      String line;
      while ((line = reader.readLine()) != null) {
         int lineValue = Integer.parseInt(line);
         if(lineValue%200000 == 0) System.out.println(line);
      }

      reader.close();


   }
}
```

Results:

buffer size = 1:
real: 3.32
user: 3.89
sys: 0.50

buffer size: = 200:
real 0.72
user 0.84
sys 0.22

Simulations with GEMC

(1) The commands for running GEMC have not changed since the last tutorial, so let's do something a little different and run the simulation on the batch farm. Take a look at the “auger-sim” file. Note that there are a few placeholders (MY_NAME, MY_INPUT_FILE, and MY_OUTPUT_DIR), let's replace these with the correct values:

```
> sed -i "s|MY_NAME|`whoami`|g" auger-sim

> sed -i "s|MY_INPUT_FILE|$PWD/gen.dat|g" auger-sim

> sed -i "s|MY_OUTPUT_DIR|$PWD|g" auger-sim
```

alternatively, you can simply make these changes with a text editor if you prefer. See https://scicomp.jlab.org/docs/text_command_file for an explanation of the various keywords in the auger submission file.

(2) Submit the auger file and confirm the batch farm received the job:

```
> jsub auger-sim

> jobstat -u `whoami`
```

* if you've never submitted a job before, you may have to create a jlab certificate using jcert (takes a few seconds).

it will most likely take at least 10 minutes for your job to finish (look for farm_sim.evio), in the meantime, let's move on with the pre-simulated file sim.evio.

* see gemc.jlab.org for more information
* see ~/.farm_out/ for files with job output/error messages

KPP raw data

- KPP raw data is located on tape at /mss/clas12/kpp/data/
- Faster access is temporarily available at /cache/clas12/kpp/data/
- Copy one KPP file to your working directory; run 809, file 902 is a good one:

```
> cp /cache/clas12/kpp/data/clas_000809.evio.902 .
```

Downloading and installing CLARA and COATJAVA

(1) Get the CLARA install script and make it executable:

```
> wget --no-check-certificate https://claraweb.jlab.org/clara/_downloads/install-claracre-clas.sh

> chmod +x install-claracre-clas.sh
```

(2) Set the CLARA_HOME environmental variable (note that you are setting it to a directory that does not yet exist):

```
> setenv CLARA_HOME $PWD/myClara/
```

(3) Run the install script, this will install both CLARA and COATJAVA. Point the COATJAVA variable to the COATJAVA installation:

```
> ./install-claracre-clas.sh -v 4a.6.0

> setenv COATJAVA $CLARA_HOME/plugins/clas12/
```

New!

* more information at claraweb.jlab.org and https://github.com/JeffersonLab/clas12-offline-software

* CLARA and COATJAVA should work on any Mac or Linux system with only one prerequisite – Java version 1.8 or higher

Decoding

(1) Now let's decode our various raw files, note GEMC files and data files are done differently:

```
> $COATJAVA/bin/evio2hipo -r 11 -t -1.0 -s 1.0 -o sim.hipo sim.evio

> $COATJAVA/bin/decoder -t -0.5 -s 0.0 -i clas_000809.evio.902 -o clas12_000809_a00902.hipo -c 2
```

Reconstruction

Changes and improvements since the last tutorial!

(1) Create “files.list” containing the hipo files to be cooked (file names only, no path):

```
> ls sim.hipo > files.list
```

(2) CLARA parameters can now be loaded from a file. Take a look at cook.clara and update the placeholders:

```
> sed -i "s|MY_WORKING_DIR|$PWD/|g" cook.clara

> sed -i "s|MY_NAME|`whoami`|g" cook.clara

> sed -i "s|MY_FILE_LIST|$PWD/files.list|g" cook.clara
```

(3) Launch the CLARA CLI:

```
> $CLARA_HOME/bin/clara-shell
```

(4) Try typing “help”, “help set”, and “show config” to get a feel for the CLI. Source the cook.clara file and run the reconstruction locally.

```
> source cook.clara

> run local
```

(5) Optional: try submitting a job to the batch farm with “run farm” instead of “run local”

Analysis Code

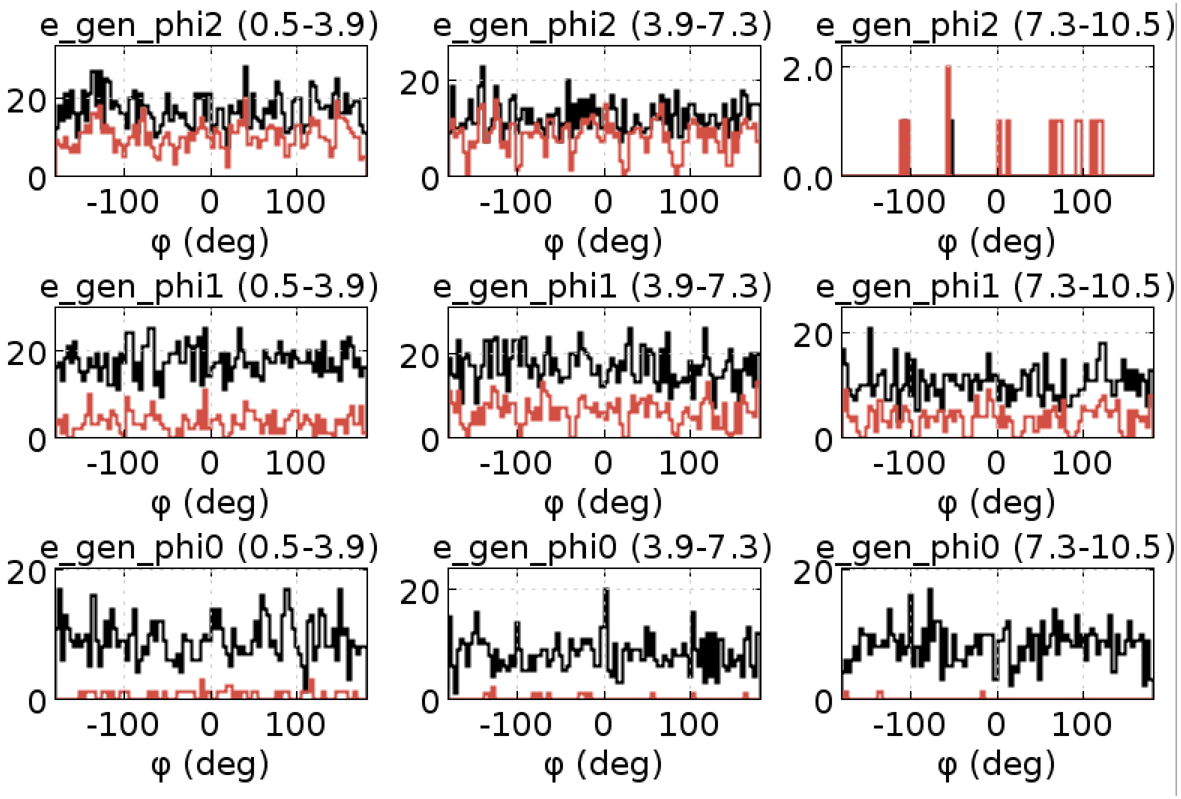
(1) Browser the structure of the data files using eviodump (also works on hipo files):

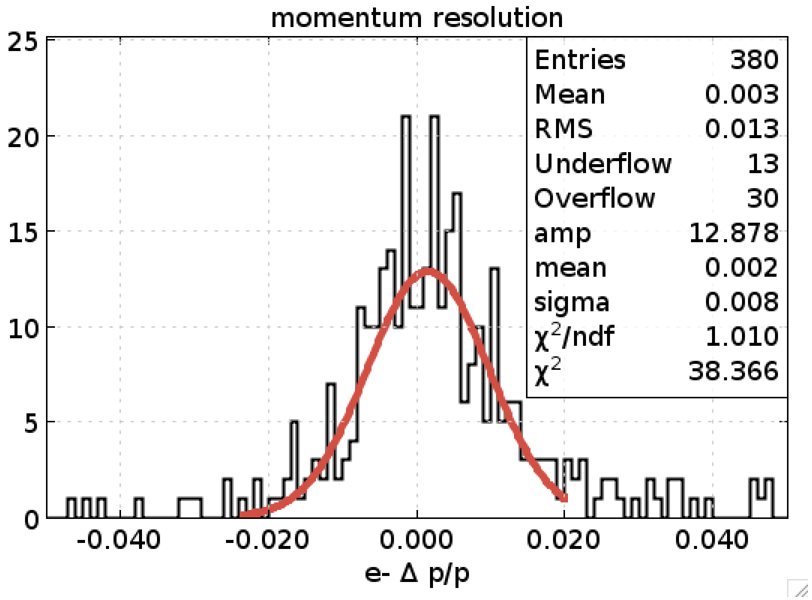
```
> $COATJAVA/bin/eviodump out_sim.hipo
```

(2) Take a look at MyAnalysis.java and compile/run it:

```
> javac -cp "$COATJAVA/lib/clas/*" MyAnalysis.java

> java -cp "$COATJAVA/lib/clas/*:." MyAnalysis
```

(3) Double click or right-click on a canvas to enlarge or adjust the options or open the fit panel.


e_gen_phi2 (0.5-3.9)
e_gen_phi2 (3.9-7.3)
e_gen_phi2 (7.3-10.5)
e_gen_phi1 (0.5-3.9)
e_gen_phi1 (3.9-7.3)
e_gen_phi1 (7.3-10.5)
e_gen_phi0 (0.5-3.9)
e_gen_phi0 (3.9-7.3)
e_gen_phi0 (7.3-10.5)
φ (deg)


momentum resolution
Entries 380
Mean 0.003
RMS 0.013
Underflow 13
Overflow 30
amp 12.878
mean 0.002
sigma 0.008
χ2/ndf 1.010
χ2 38.366
e- Δ p/p