Calibration Data Warehousing

ARA Operations Call
October 19th, 2016
Jordan Hanson

Storing Calibration Data

- We must keep two criteria in mind
  - Storage strategy must encourage people to use the data
  - Storage strategy must be efficient and organized
  - Of course, open to suggestions
- SQL (sqlite3) is already being used in AraRoot
  - Stores pedestals, station geometries
  - We can also store antenna radiation patterns, **calibration pulser waveforms**, gains, part numbers etc
- I have created a C-program to load calibration pulser data into a SQL data base

Calibration Pulse Lab Data

- Collected by Liz Freidman at the University of Maryland, who shared it with me
- There were different pulse units
- There were different temperature runs
- There were different data taking modes (averaged and single-shot)
- There were different sampling frequencies

Code, SQL, and the data

- The code/data is now uploaded at osu0234@ruby.ocs.edu/users/PAS0654/osu0234, and Dropbox
- Send me all calibration data you have
- Example calibration pulser data format:

```
Room-Temp-Runs/408_A_1ns_avg_room.txt
Pulser ID: 408-A
Sampling: 1ns per sample
Mode: averaging
Temperature run: room temperature
```

Code, SQL, and the data

```
void split_calibration_pulser_tags
(
std::string title,std::string &number,std::string &letter,
std::string &time,std::string &mode,std::string &tempRun
) {...}

static int callback(void *NotUsed, int argc, char **argv, char **azColName)
{
    int i;
    for(i=0;i<argc;i++)
    {
        printf("%s = %s\n", azColName[i], argv[i] ? argv[i] : "NULL");
    }
    printf("\n");
    return 0;
}
```

Code, SQL, and the data

```
int main(int argc, char **argv)
{
…
//Read in the list of files to pull into the database.
    std::ifstream all_files(argv[2]);
    while (all_files.good() && !all_files.eof())
    {
        std::getline(all_files,current_file);
        files.push_back(current_file);
    }
    all_files.close();
```

Code, SQL, and the data

```
…
 rc = sqlite3_open(argv[1], &db);
…
for(si=files.begin();si!=files.end()-1;++si) //Looping over all data
files
{
…
     infile>>t; tString<<t;
     infile>>s; sString<<s;
     sqlCurrent = "insert into '"+dataTitle+"' values("+tString.str()...
     statements.push_back(sqlCurrent);
     tString.str(std::string());
     sString.str(std::string());
```

Code, SQL, and the data

```
while(performedActions!=totalActions)
{
     rc = sqlite3_exec(db,statements[performedActions].c_str(),
     callback,0,&zErrMsg);
     if(rc!=SQLITE_OK)
     {
          fprintf(stderr, "SQL error: %s, action #%i\n",zErrMsg,performedActions);
          sqlite3_free(zErrMsg);
          break;
     }
     else
     {
          ++performedActions;
     }
}
```

Compilation and Execution

- g++ sqlLoadVector.cxx -o sqlLoadVector.cxx -lsqlite3
- Example: ./sqlLoadVector calpulser_database.sqlite cal_pulser_files.dat
- cal_pulser_files.dat is a file listing all the paths to calibration data files (cannot contain “_” delimiters)

Work with the Data

- sqlite3 calpulser_database.sqlite
- In sqlite3: “.tables” to list all tables
- In sqlite3: “select * from 'Room-Temp-Runs/420_A_250ps_avg_room'”
- Output: (time,voltage,pulser,letter,mode,tempRun)

```
-1.245e-09|-0.00232031|420|A|250ps|avg|room
-1.24e-09|-0.00236719|420|A|250ps|avg|room
-1.235e-09|-0.00239844|420|A|250ps|avg|room
-1.23e-09|-0.00244531|420|A|250ps|avg|room
-1.225e-09|-0.0025|420|A|250ps|avg|room
```

Next Steps

- More logical organization of tables (just one listing of meta-data)
- Optimization of code
- More data!
- Add sqlite database output to AraRoot
- Questions, suggestions?