Summer Student Report

Harshit Jain

EP – LBC Department
Supervisors
Balazs Voneki
Niko Neufeld

28 July, 2017

Abstract

The LHCb Event Filter Farm consists of more than 1500 server nodes with a total amount of roughly 65 TB operating memory .The memory is crucial for the success of the LHCb experiment, since the proton-proton collisions are temporarily stored on these memory modules. Unfortunately, the aging nodes of the server farm occasionally suffer losses of their memory modules. The lower the available memory, the lower performance we can get out of it. Inducing the users or administrators to pay attention to this matter is inefficient. One needs to upgrade it to an acceptable way. The aim of this project was to develop a software to monitor a set of test machines. The software stores the data of the memory sticks in advance in a database which will be used for future reference. Then it checks the memory sticks at a future time instant to find any failures. In the case of any such losses the software looks up in the database to find out which memory sticks have lost and displays all information of those sticks in a log file. The software is scalable and tested for a couple of test machines.

Evolution of the software

Part 1: Creating the database

Basic Terminologies:

- *SSH*: Secure Shell (SSH) is a cryptographic network protocol for operating network services securely over an unsecured network. The best known example application is for remote login to computer systems by users. SSH provides a secure channel over an unsecured network in a client-server architecture, connecting an SSH-Client application with an SSH Server.
- *Dmidecode*: dmidecode is a tool for dumping a computer's DMI (some say SMBIOS) table contents in a human-readable format. This table contains a description of the system's hardware components , as well as other useful pieces of information such as serial numbers and BIOS revision.

Accessing a test machine's information of memory sticks:

- Use the command **ssh "name of test machine"**(to connect to the required test machine.
- The command **sudo dmidecode -t 17** gives the information of all the memory devices on that machine. The information is presented in a human readable format.

How to redirect that information in a file

Use sudo dmidecode -t 17 > "filename.txt". This will direct the output into the file.

How to parse the information in the file

One needs a scripting language to do the parsing and also to connect to the putty terminal in a programmatic way. I chose Python which is well suited for this work and an internet friendly language for developers.

Connecting to ssh in python:

The python module **pxssh** automates the ssh login and other commands on the terminal.

```
1  import pxssh
2  try:
3      s = pxssh.pxssh()
4      hostname="test amchien you wanna connect to"
5      username="your username"
6      password="your password"
7      s.login(hostname, username,password)
8      mys3='sudo dmidecode -t 17 '
9      s.sendline( mys3 )
10     s.prompt()
11 except pxssh.ExceptionPxssh as e:
12     print("pxssh failed on login.")
13     print(e)
```

Figure 1. Using pxssh

Parsing the dmidecode file :

Each test machine has a couple of memory devices which have a unique Handle and 20 other properties such as Speed, Size etc. As this information is periodic I could use 2 for loops for capturing the data into lists and then feeding into the database. The first couple of lines in the files are different for each test machine and hence should be taken care of.

Why is a database required?

Earlier I was storing this data into a list of dictionaries, where each item of the list is a dictionary containing all these details of a handle. A database is more scalable than individual files and easy to handle.

Installation of MySQL

- I followed these steps to install MySQL on Centos on a test machine (say lab30)

```
##  INSTALL MySQL YUM REPOSITORY
sudo yum localinstall https://dev.mysql.com/get/mysql57-community-release-el7-11.noarch.rpm

##  INSTALL MySQL-SERVER
yum install mysql-community-server

##  START THE SERVER
sudo service mysqld start

##  GENERATE A TEMPORARY PASSWORD FOR ROOT
grep 'A temporary password is generated for root@localhost' /var/log/mysqld.log |tail -1

##  A PROMPT LIKE THIS WILL APPEAR WHICH SHOWS THE PASSWORD
2015-11-20T21:11:44.229891Z 1 [Note] A temporary password is generated for root@localhost: -et)QoL4MLid

##  CHANGE PASSWORD FOR A SECURE INSTALLATION AND COMPLETE OTHER DETAILS
/usr/bin/mysql_secure_installation
```

Figure 2. Installing MySQL

- MySQL can be started by the following command

  mysql -u root -p

  Enter your password and you are ready to go.

Connecting MySQL with Python:

- Use the command sudo **yum install MySQL-python.**

Starting the database connection and creating a database using Python script

```
1  import MySQLdb
2
3  def db_conn():
4      db = MySQLdb.connect("localhost","root","your password")
5      db1 = db.cursor()
6      db1.execute('create database if not exists test3')
7      db1.execute('use test3 ')
8      return db1,db
```

Figure 3. Starting the database connection

Database Design

- I have created tables corresponding to each test machine e.g. table names are lab30 , lab31 etc. The columns are the properties of each memory device in the machine and the rows are the different handles.
- The command desc tablename gives the details of a table. The Primary Key is set as the Handle because its unique.

Field	Type	Null	Key	Default	Extra
Handle	varchar(20)	NO	PRI	NULL	
Array_Handle	varchar(20)	YES		NULL	
Error_Information_Handle	varchar(20)	YES		NULL	
Total_Width	varchar(20)	YES		NULL	
Data_Width	varchar(20)	YES		NULL	
Size	varchar(20)	YES		NULL	
Form_Factor	varchar(20)	YES		NULL	
Sets	varchar(20)	YES		NULL	
Locator	varchar(20)	YES		NULL	
Bank_Locator	varchar(20)	YES		NULL	
Type	varchar(20)	YES		NULL	
Type_Detail	varchar(20)	YES		NULL	
Speed	varchar(20)	YES		NULL	
Manufacturer	varchar(20)	YES		NULL	
Serial_Number	varchar(20)	YES		NULL	
Asset_tag	varchar(20)	YES		NULL	
Part_Number	varchar(20)	YES		NULL	
Rank	varchar(20)	YES		NULL	
Configured_Clock_Speed	varchar(20)	YES		NULL	
Minimum_Voltage	varchar(20)	YES		NULL	
Maximum_Voltage	varchar(20)	YES		NULL	
Configured_Voltage	varchar(20)	YES		NULL	

Figure 4. Table Structure

All parameters are strings because during the parsing of the file everything was parsed as a string which may be Unknown or contain some data with its unit.

Inserting data into the table

```
1  file1=open("respective dmidecode file",'r')
2  file2=file1.readlines()
3  while (i < leng):
4          my_dic={}
5          list2=[]
6          handle=file2[i].split(',')
7          handle2=handle[0].split(' ')
8          handle3=handle2[1]
9          #looping through each handle
10         for j in range(i+2,i+23):
11                 temp=file2[j].split(':')
12                 list2.append(temp[1].strip())
13         #inserting in database
14         que3='insert into '+par2+' values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,
15         args=(handle3,list2[0],list2[1],list2[2],list2[3],list2[4],list2[5],li
16         db1.execute(que3,args)
17         i=i+24
```

Figure 5. Inserting data in the table

All this is encoded in a function refdb_make which is called once for each lab node in a for loop. This creates the tables for each node.

Part 2 Checking for stick failures in future

Criterion for a stick failure:

Stick failures are basically the memory devices whose parameters such as width, size, speed, manufacturer etc are unknown. These unknown sticks should be reported.

- The file check.py checks the dmidecode of the test machines at a future time instance against the reference database created in the past.

- The dmidecode data of the test machines is inserted in a similar manner in the database as tables lab30_new, lab31_new etc. All the rows in both the old and new tables for a lab node are matched. If any stick in the new table goes missing/unknown the instance and the lost data is reported in a log file.

```
1  LOG FILE
2  DATE AND TIME 2017-07-27 00:03
3  NODE CHECKED: lab30
4  EXPECTED MEMORY STICKS ARE: 8
5  MEMORY STICKS FOUND: 8
6  MEMORY STICKS LOST ARE:
7  All SET
8  ____________________
9
```

Figure 6. Log File when there are no stick failures

```
1  LOG FILE
2  DATE AND TIME 2017-07-27 00:00
3  NODE CHECKED: lab31
4  EXPECTED MEMORY STICKS ARE: 8
5  MEMORY STICKS FOUND: 7
6  MEMORY STICKS LOST ARE:
7  Handle 0x0059
8  Memory Device
9      Array Handle:0x0054
10     Error Information Handle:Not Provided
11     Total Width:72 bits
12     Data Width: 72 bits
13     Size:8192 MB
14     Form Factor: DIMM
15     Set: None
16     Locator:DIMM_B1
17     Bank Locator: NODE 1
18     Type:DDR4
19     Type Detail:Synchronous
20     Speed:2133 MHz
21     Serial Number:Samsung
22     Asset Tag:41D13165
23     Part Number:DIMM_B1_AssetTag
24     Rank:M393A1G40DB0-CPB
25     Configured Clock Speed:1
26     Minimum Voltage:1866 MHz
27     Maximum Voltage:Unknown
28     Configured Voltage:Unknown
29
```

Figure 7. Log file when a stick goes missing

When some sticks go missing. (Presently I altered the new files manually to check if its correctly reporting and error because currently both tables should be same but they may be different in the future.)

Softwares and References

- Python 2.7.5
- My-SQL community-server 5.7.19
- MySQL-python-1.2.5 (https://pypi.python.org/pypi/MySQL-python/1.2.5)
- LHCb online (https://lbtwiki.cern.ch)
- http://www.nongnu.org/dmidecode/


Future Scope

The database design could be improved where instead of keeping a table for each machine there should be a single table with hostname (test machine) as an additional column. I was not able to do that because of time constraints and initially I was not even designing a database for my project.

Conclusion

By the means of this project I was able to learn quite a lot about python, MySQL and bash scripting. Apart from the project I got to learn how things work in the LHCb Computing Department at CERN and a general overview of how software engineers and physicists work at CERN in a collaborative manner. I also engaged in group meetings and discussions . It was a summer like never before and I got to work in an international and diverse environment.