

B. TECH. PROJECT REPORT

On

Creating Regression Webpage *and* Storing Historical Regressions

BY
Rishabh Raj
140001026



DISCIPLINE OF COMPUTER SCIENCE and ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE

December 2017

Creating Regression Webpage and Storing Historical Regressions

A PROJECT REPORT

*Submitted in partial fulfillment of the
requirements for the award of the degrees*

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE and ENGINEERING

Submitted by:

Rishabh Raj

140001026

Guided by:

Dr. Surya Prakash

Asst. Professor, IIT Indore

RohitJeevnani

**Lead Member Technical Staff,
Mentor Graphics India Pvt. Ltd.**



**DISCIPLINE OF COMPUTER SCIENCE and ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE**

December 2017

CANDIDATE'S DECLARATION

I hereby declare that the project entitled “**Creating Regression Webpage and Storing Historical Regressions**” submitted in partial fulfillment for the award of the degree of Bachelor of Technology in “Computer Science and Engineering”, completed under the supervision of **Mr. Rohit Jeevnani, Lead Member Technical Staff, Mentor Graphics** and **Dr. Surya Prakash, Asst. Professor, IIT Indore**, is an authentic work.

Further, I declare that I have not submitted this work for the award of any other degree elsewhere.

Date: _____

Signature: _____

Rishabh Raj
140001026

CERTIFICATE by BTP Guide(s)

It is certified that the statement made by the student in the previous page (under CANDIDATE'S DECLARATION) is correct to the best of our knowledge.

Rohit Jeevnani
Lead Member,
Technical Staff,
Mentor Graphics India Pvt. Ltd

Dr. Surya Prakash
Asst. Professor,
IIT Indore

Preface

This report on “Creating Regression Webpage and Storing Historical Regressions” is prepared under the guidance of my mentor at Mentor Graphics, Mr. Rohit Jeevnani and BTP supervisor at IIT Indore, Dr. Surya Prakash.

Through this report I have tried to give a detailed description of how the project was completed in the course of 6 months in the company and the various technologies used for the purpose.

I have tried to the best of my abilities and knowledge to explain the content in a lucid manner. I have also added figures to make it more illustrative.

Rishabh Raj

B.Tech. IV Year

Discipline of Computer Science and Engineering

IIT Indore

Acknowledgements

I wish to thank my mentor in the company, Mr. Rohit Jeevnani and Dr. Surya Prakash for their kind support and valuable guidance.

It is their help and support, due to which I became able to complete the design and implementation of the project. Without their support this work would not have been possible.

Rishabh Raj

B.Tech. IV Year

Discipline of Computer Science and Engineering

IIT Indore

Table of Content

Candidate's Declaration	3
Certificate by BTP Guide	4
Preface	5
Acknowledgements	6
Table of Content	7
Introduction	8
Technologies Used	9
Related Work and Motivation	10
Work Done	
1) Developed a Regression monitoring web application	11
2) Storing historical regressions in a MongoDB database	19
Conclusion	22
Bibliography	23

INTRODUCTION

Developed a web application for the company which shows real time data functions or regressions.

Regressions are running of suites under a product against different testcases which results in Pass or Fail status along with various attributes of the testcases like runtime, memory consumed etc.

The regressions are run on a daily basis. Suites of the product 'Powerpro' are run in India whereas those of 'Slec' are run in the US.

The application basically shows the status of testcases of various suites under the two products, Powerpro (INDIA) and Slec (US).

Also designed a database in MongoDB to store the company's data items for their products mainly emphasizing on fast queries and also considerable data insertion rate.

TECHNOLOGIES USED

- 1) Python - For server sided scripting, Python is used. It is a high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords).
- 2) MySQL – The MySQL server is used as a database management system. It is an open-source relational database management system (RDBMS). Which means data is stored in the form of tables.
- 3) HTML – It is a standard markup language for creating webpages and web applications. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.
- 4) JavaScript - It is a high-level, dynamic, weakly typed, prototype-based, multi-paradigm, and interpreted programming language. It is used to make webpages interactive and provide online programs, including video games.
- 5) CSS - Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language.[1] Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG and XUL, and is applicable to rendering in speech, or on other media. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications.
- 6) MongoDB - MongoDB (from humongous) is a free and open-source cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with schemas.

RELATED WORK and MOTIVATION

The part to employ the OOP (Object oriented programming) approach was very much required, the importance of which I realised in course of time.

If a change was required, I had to go through all the scripts and make the changes which was very much troublesome and also was quite difficult to debug.

By writing the code in form of class and objects and keeping the same part of code in a module made the task become easier.

Now, when a change needed to be made, I had to just change a small part in the corresponding module.

Code modularity was motivating to follow the OOP approach.

WORK DONE

1) Developed a Regression monitoring web application

RHEL CentOS 6 environment was used for development.

First of all, the configuration part had to be completed for the server side. Apache server had to be configured so that it can execute Python and JavaScript files when needed. To achieve that, changes in “httpd.conf” file had to be made to make the python scripts executable by the Apache server.

All the python scripts had to be given executable permissions and needed to be kept inside ‘/var/www/cgi-bin’ folder for the Apache server to recognize it.

Also, all the html files which would be created needed to be inside ‘/var/www/html’ folder.

I/O operation was quite needed in this project since ‘txt’ files acted as temporary storage of information later from which the data was extracted whenever required. All the temporary files were kept in ‘/tmp’ folder.

This was how the architecture was setup for the product:

There were 2 products, POWERPRO and SLEC. Each product had various Suites under them which in turn were run against a finite set of testcases written in ‘TCL’ language on regular intervals, say once per day. Running of the Suites against different testcases were called ‘Regressions’.

The summary of Regressions were dumped on a daily basis into a file called ‘qor’.

```

- sls /calypto/tests/qa_repository/testcases/unittests/cdb/flatedge/md.tcl
/calypto/tests/qa_repository/testcases/unittests/cdb/flatedge/md_1.v:
exit=0; wallclocktime=31; hostname=inngriidl58.inn.mentorg.com; cpu=2;
optimalcpu=2; realtime=27; leak=41; mem=65; verilog=1; bd_cpu=2;
bd_mem=65
- sls /calypto/tests/qa_repository/testcases/unittests/cdb/flatedge/md.tcl
/calypto/tests/qa_repository/testcases/unittests/cdb/flatedge/md_3.v:
exit=0; wallclocktime=11; hostname=inngriidl76; cpu=0; optimalcpu=0;
realtime=9; leak=41; mem=65; verilog=1; bd_cpu=0; bd_mem=65
- sls /calypto/tests/qa_repository/testcases/unittests/cdb/flatedge/md.tcl
/calypto/tests/qa_repository/testcases/unittests/cdb/flatedge/md_2.v:
exit=0; wallclocktime=21; hostname=inngriidl58.inn.mentorg.com; cpu=2;
optimalcpu=2; realtime=20; leak=41; mem=65; verilog=1; bd_cpu=1;
bd_mem=65
- sls /calypto/tests/qa_repository/testcases/unittests/cdb/flatedge/md.tcl
/calypto/tests/qa_repository/testcases/unittests/cdb/flatedge/red_loop0.v
: exit=0; wallclocktime=7; hostname=inngriidl76; cpu=0; optimalcpu=0;
realtime=6; leak=41; mem=65; verilog=1; bd_cpu=0; bd_mem=65
- sls /calypto/tests/qa_repository/testcases/unittests/cdb/flatedge/md.tcl
/calypto/tests/qa_repository/testcases/unittests/cdb/flatedge/red_loop0a.
v: exit=0; wallclocktime=20; hostname=inngriidl58.inn.mentorg.com; cpu=1;
optimalcpu=1; realtime=20; leak=41; mem=65; verilog=1; bd_cpu=1;
bd_mem=65
- sls /calypto/tests/qa_repository/testcases/unittests/cdb/flatedge/md.tcl
/calypto/tests/qa_repository/testcases/unittests/cdb/flatedge/red_loop1.v
: exit=0; wallclocktime=7; hostname=inngriidl76; cpu=0; optimalcpu=0;
realtime=6; leak=41; mem=65; verilog=1; bd_cpu=0; bd_mem=65
- sls /calypto/tests/qa_repository/testcases/unittests/cdb/flatedge/md.tcl

```

(Fig.1) 'Qor' file text format

'Qor' files were 'txt' files populated with testcase names. Each line in the 'qor' file had the testcase name followed by a colon (:) and then metric names along with their values separated by semi-colon (;).

Eg:-Testcase : Metric1=value1 ; Metric2=value2

Among the metrics there was a metric named 'exit' which only took binary values. '1' indicating the corresponding testcase failed whereas '0' denoted the passed testcase.

Two MySQL servers were setup, one in India and other one in the US. Two python scripts 'powerpro.py' and 'slec.py' had to be run continuously to insert data into the MySQL database.

'powerpro.py' is run in India whereas 'slec.py' is run in the US.

These two scripts are run continuously to check the appearance of new data and insert them in the database accordingly.

The MySQL database consisted of two tables, one for the product 'Powerpro' and the other for 'Slec'.

Data were parsed from the 'qor' file and then dumped into the database.

Each table had 13 columns namely, 1) Suite name, 2) Total testcases, 3) Status, 4) All passed testcases separated by commas, 5) All failed testcases separated by commas, 6) Pass count, 7) Fail count, 8) Date, 9) Runtime pass testcases separated by commas, 10) Memory pass

testcases separated by commas, 11) Runtime fail testcases separated by commas, 12) Memory fail testcases separated by commas and 13) ID.

Runtime pass testcases - Passed testcases runtime values (seconds)

Memory passtestcases - Passed testcases memory values (MB)

Runtime fail testcases - Failed testcases runtime values (seconds)

Memory fail testcases - Failed testcases memory values (MB)

Both the tables keeps data for two days i.e, today and yesterday.

An ID was assigned to the data's representing today and yesterday. Today's data had ID=1 and yesterday's data had ID=2.

The python script used the module 'PyMySQL' for creating the database connection.

The script one by one checked the date of regression of every suite from the 'summary' file and checked it with the date stored in the database having ID=1. When the date matched, the script just continued from there without doing anything further because today's data was already there in the database. When the dates were different, data having ID=2 were deleted from the database, all those data related to that particular suite having ID=1 now would have ID=2 and the new data is inserted having ID=1.

This way, data insertion was optimal since the work was not being repeated again and again.

All the data were taken from the 'qor' file. The script parsed the file going through it line by line. Testcase name and the metric 'exit' was extracted from each line and feeded into the database .

Testcases with 'exit' metric value as '1' went into the 'Failed testcase' column whereas with value as '0' went into the 'Passed testcase' column.

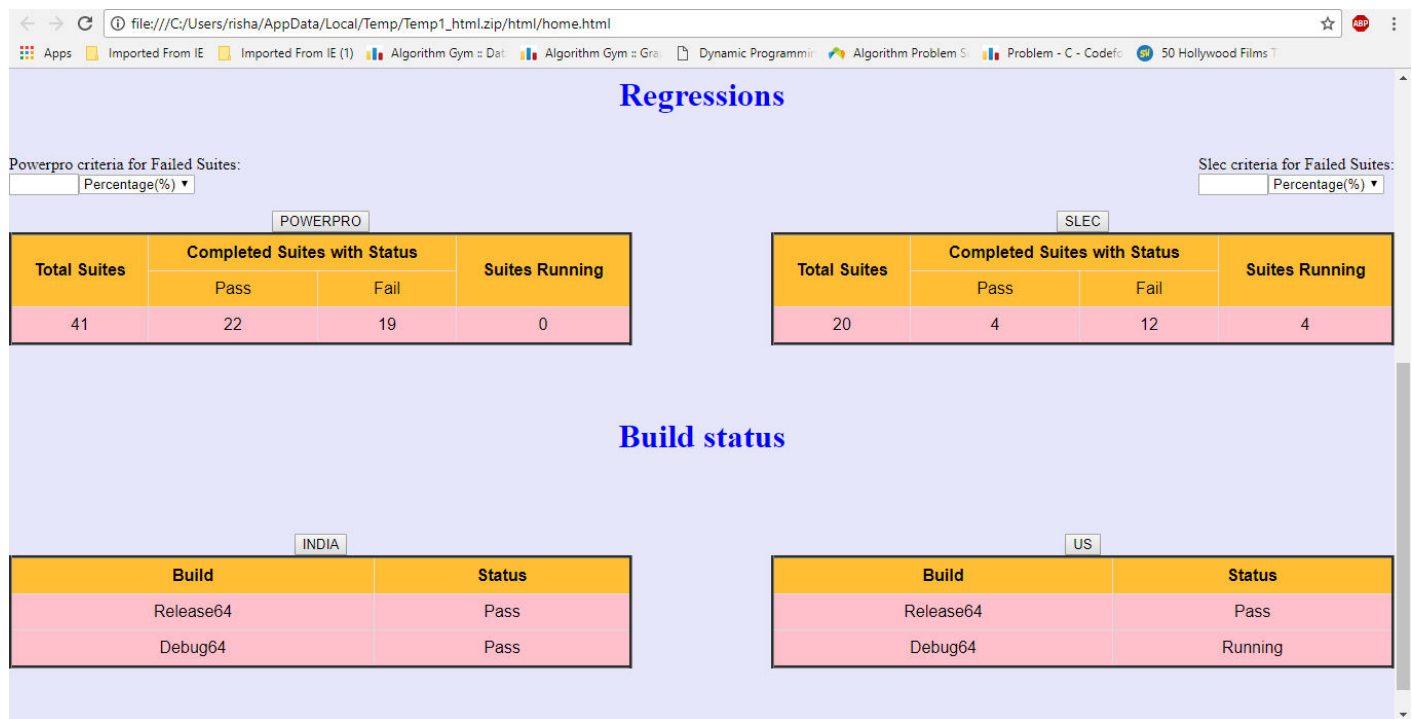
While inserting data, the pass count and fail count were also stored in the database so that in the next iteration the 'qor' file reading will start from the line after until what was read in the last iteration for a Suite.

The purpose of storing these two fields was to avoid the repetitive task and save time.

A technique called CGI (Common Gateway Interface) is used so that html forms can pass the values of their respective tags (which maybe given as input from the user) to the appropriate variables of the Python scripts via the Apache server. The Python script in turn does computations with the passed values and embeds the correct values in modified forms into the html page (HTML page are also generated by the python scripts so that it can be viewed as a webpage).

Whenever a page is opened, the corresponding Python script is called which gets the appropriate data from the MySQL database and writes the result into a new html file. The generated html file is then displayed to the user as a webpage.

The first page which is the homepage shows the status of both the products, POWERPRO and SLEC in brief. It just shows Total testcases, Passed testcases and Failedtestcases in the form of tables.



(Fig.2) Application homepage

Suite names in the first pages are buttons which when clicked takes to the regression webpage showing all the Suites status of the clicked product name.

In the backend, when the button was clicked, a form was submitted and a Python script was called. The python script retrieved data from the MySQL database for all the Suites under the product name which was clicked.

The returned data was in the form of list of tuples. A for loop was performed on the list items and one by one all the data was written in a 'html' file in the form of tables.

The page of a specific product shows all the Suite names with their statuses and their dates in tabular form.

The Suite names have links which when clicked upon takes to the webpage showing Regression status of all the testcases which were run against that particular Suite.

SUITE NAME	Total	Passed	Failed	STATUS	BUILD NAME
stress_full_flow_after_pro	34418	34386	32	Fail	2017-09-23-Sat-2230
stress_nightlies_bugtest	34172	34105	67	Fail	2017-09-23-Sat-2230
ppro_cgqor_large_weekly	35	35	0	Pass	2017-09-22-Fri-2230
ppro_powerleaks_weekly	15	14	1	Fail	2017-09-26-Tue-2230
ppro_glpa_large_weekly	10	10	0	Pass	2017-09-22-Fri-2230
ppro_weekly	5230	5224	6	Fail	2017-09-26-Tue-2230
ppro_cgmg_flop_slice	231	231	0	Pass	2017-09-23-Sat-2230

(Fig.3)Suites and Status for the product ‘Powerpro’

Here also, in the backend when a Suite name was clicked, a form was submitted with the passed value as the Suite name and a python script was called. The script here queried the database and retrieved the data items of that particular Suite only. Again, a for loop was performed and only by one each data was written in a ‘html’ file in the form of tables.

Testcases names also are hyperlinks pointing to the actual code written in ‘TCL’ language.

Test Case	Runtime(s)	Memory(MB)	Status
-cg -mg /calypto/tests/qa_repository/shasta_qor/scripts/NEW.CG_PA_RC/phy_synth_scripts/cg_pa_rc_qor_obs_stb.tcl /tests/qa_repository/shasta_qor/testcases/slec_bugtests_in_shasta/129_bug1840/pa_data/129_bug1840.tcl_integrated.def	49	260	Fail Link
-cg -mg /calypto/tests/qa_repository/shasta_qor/scripts/NEW.CG_PA_RC/phy_synth_scripts/cg_pa_rc_qor_obs_stb.tcl qa_repository/shasta_qor/testcases/slec_bugtests_in_shasta/111_bug1772_small/pa_data/111_bug1772_small.tcl_integrated.def	45	260	Fail Link
-cg -mg /calypto/tests/qa_repository/shasta_qor/scripts/NEW.CG_PA_RC/phy_synth_scripts/cg_pa_rc_qor_obs_stb.tcl /calypto/tests/qa_repository/shasta_qor/testcases/lcfg_work-run_full/pa_data/lcfg_integrated.def	69	265	Fail Link
-cg -mg /calypto/tests/qa_repository/shasta_qor/scripts/NEW.CG_PA_RC/phy_synth_scripts/cg_pa_rc_qor_obs_stb.tcl /calypto/tests/qa_repository/shasta_qor/testcases/densbit/pa_data/sdip_top_integrated.def	72	290	Fail Link
-cg -mg /calypto/tests/qa_repository/shasta_qor/scripts/NEW.CG_PA_RC/phy_synth_scripts/cg_pa_rc_qor_obs_stb.tcl ypto/tests/qa_repository/shasta_qor/testcases/STMicro/PPRO_CSDI/pa_data/c8fvp3_csd_i_pu_monopu_integrated.def	116	275	Fail Link

(Fig.4) Testcases hyperlinks of a particular Suite

Also, a link for ‘Rerun’ was there. Rerun of testcases happened when the testcase had failed and the number of testcases of a Suite that failed were < 100.

In generating Rerun regression page, database was not involved since the number of testcases which were rerun were usually small in number. So the ‘qor’ file were read in runtime and the data were embedded into the ‘html’ file at the same moment and then displayed to the user.

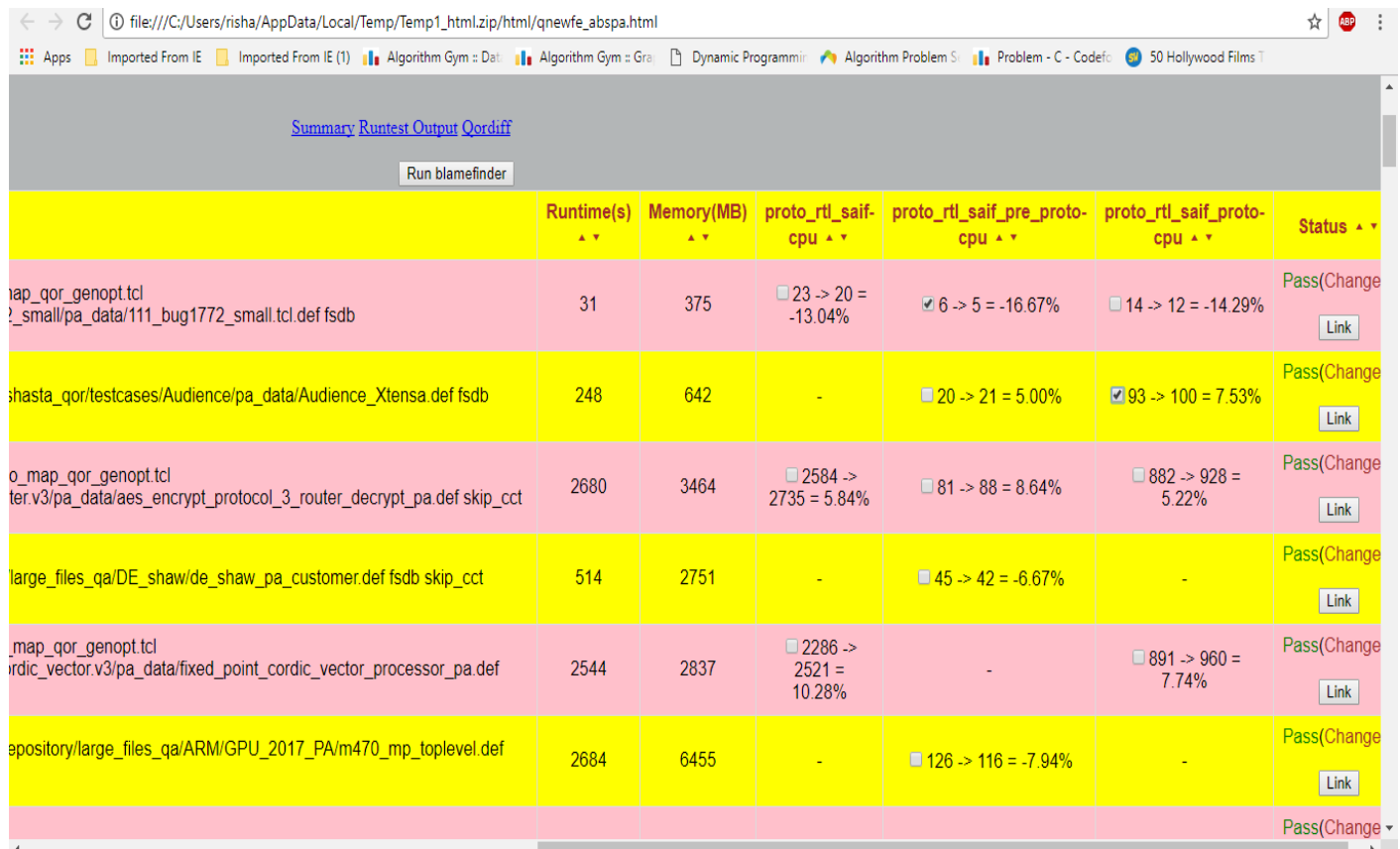
Link for ‘Rerun’ only appeared for the Suite in which the testcases were rerun. To accomplish this, while generating the Regression page, it was checked if there was a new ‘qor’ file inside a folder named ‘rerun’ in the filesystem. If the folder was found empty, no ‘Rerun’ link was given.

A hover was also generated whenever the mouse pointer was brought near the text part of the testcase name. Hover showed the top 3 metrices corresponding to that testcase. Top 3 metrices were extracted from the file called ‘qordiff.out’.

In the file, all ‘metrices’ were included, so a script checked for only the top 3 fields and then extracted the values accordingly.

The pages are in the form of tables which has features like Filter and Sorting. To achieve these features free JavaScript API's are used.

There are checkboxes against some fields in some particular columns. A send mail button is added. So, when a user checks some of the items and clicks the 'Send mail' button, it sends the mail (with text as what was checked by the user) to a particular recipient.



	Runtime(s)	Memory(MB)	proto_rtl_saif-cpu	proto_rtl_saif_pre_proto-cpu	proto_rtl_saif_proto-cpu	Status
map_qor_genopt.tcl ?_small/pa_data/111_bug1772_small.tcl.def fsdb	31	375	<input type="checkbox"/> 23 -> 20 = -13.04%	<input checked="" type="checkbox"/> 6 -> 5 = -16.67%	<input type="checkbox"/> 14 -> 12 = -14.29%	Pass(Change Link)
shasta_qor/testcases/Audience/pa_data/Audience_Xtensa.def fsdb	248	642	-	<input type="checkbox"/> 20 -> 21 = 5.00%	<input checked="" type="checkbox"/> 93 -> 100 = 7.53%	Pass(Change Link)
o_map_qor_genopt.tcl ter.v3/pa_data/aes_encrypt_protocol_3_router_decrypt_pa.def skip_cct	2680	3464	<input type="checkbox"/> 2584 -> 2735 = 5.84%	<input type="checkbox"/> 81 -> 88 = 8.64%	<input type="checkbox"/> 882 -> 928 = 5.22%	Pass(Change Link)
large_files_qa/DE_shaw/de_shaw_pa_customer.def fsdb skip_cct	514	2751	-	<input type="checkbox"/> 45 -> 42 = -6.67%	-	Pass(Change Link)
map_qor_genopt.tcl rdic_vector.v3/pa_data/fixed_point_cordic_vector_processor_pa.def	2544	2837	<input type="checkbox"/> 2286 -> 2521 = 10.28%	-	<input type="checkbox"/> 891 -> 960 = 7.74%	Pass(Change Link)
epository/large_files_qa/ARM/GPU_2017_PA/m470_mp_toplevel.def	2684	6455	-	<input type="checkbox"/> 126 -> 116 = -7.94%	-	Pass(Change Link)
						Pass(Change

(Fig.5) Checkboxes against some testcases with a 'Run blamefinder' button

There is also a feature called 'Changed' which is basically the percentage difference between a metric of a testcase of Suite under a product of today and yesterday.

This changed feature is displayed in graph form (lines) in a webpage of Today and Yesterday. To achieve this, a JavaScript API is used which is available for free from the internet.

When certain testcases are still running, the webpage automatically refreshes itself every 2 minutes.

Also, every generated webpage has a common navigation list with hovering feature at the top. It contains lists of lists. One can also go to a particular Suite Regression webpage by going to 'Regressions' -> Suite name.

The CSS part of navigation list was copied from a website. It also has links to various other products of the company.

Since this part was common, when the first time the webpage was loaded, the code pertaining to only the common part was saved to a 'txt' file. So, whenever a different webpage was to be followed, the contents of the saved 'txt' file was copied to the 'html' file and rendered accordingly.

JavaScript is used to make the pages more interactive and dynamic in nature.

CSS is used to give styling to the pages.

2) Storing historical regressions in a MongoDB database

Designed a database in NoSQL using MongoDB to store the historical regressions of all testcases under all Suites of all the products.

First, experiments were done on a smaller data set to find out which was better suited for the problem, MySQL or MongoDB?

Experiment showed that while MySQL insertion rate dropped drastically when number of rows were increasing since the table had indexes. Insertion rate in MySQL was more quicker than MongoDB when there were not enough records, but when the number of records started increasing, insertion rate in MySQL dropped significantly whereas that of MongoDB almost remained same with a slight change in insertion rate.

In terms of query, MongoDB had slower data retrieval rate than that of its MySQL counterpart in the beginning. Here also like insertion, query started becoming quite slow with increase in number of records in the case of MySQL. In MongoDB, the query rate did not drop suddenly but uniformly.

Also, MySQL required more memory for its database storage whereas MongoDB required almost one third of what was used by MySQL.

Thus it was concluded that MongoDB performs better than MySQL when data was quite huge and hence, finally MongoDB was made as a suitable choice.

A record in MongoDB is a document, which is a data structure composed of field and value pairs. MongoDB documents are similar to JSON objects. The values of fields may include other documents, arrays, and arrays of documents.

The advantages of using documents are:

- Documents (i.e. objects) correspond to native data types in many programming languages.
- Embedded documents and arrays reduce need for expensive joins.
- Dynamic schema supports fluent polymorphism.

MongoDB stores data records as BSON documents. BSON is a binary representation of JSON documents, though it contains more data types than JSON.

The value of a field can be any of the BSON data types, including other documents, arrays, and arrays of documents.

Following data items are required to be dumped into database:

- 1) Date
- 2) Suite
- 3) Testcase
- 4) Metric

Here, Metric is the attribute associated with each of the testcase which takes numerical/string values as fields.

So, basically I designed the database such that data dumping and query both are balanced in terms of optimizations.

Queries will focus on finding metric with following scenarios:

- 1) Date, Suite, Testcase given
- 2) Date, Suite given and Testcase not given
- 3) Date given and Suite, Testcase not given
- 4) Suite, Testcase given and Date not given
- 5) Suite given and Date, Testcase not given
- 6) Testcase given and Date, Suite not given

If metric is not given then it should display all the associated metrics.

Data in MongoDB is stored in JSON like format as a tree structure to make the queries faster.

MongoDB documents stored two fields, 'path' and 'metric' string.

Path is a string comprised of Date, Suite, Testcase separated by commas(,)

Example – “Date,Suite,Testcase”

Metric is just a string consisting of metric values.

Indexing is done on the 'path' field.

Indexes support the efficient execution of queries in MongoDB.

Without indexes, MongoDB must perform a collection scan, i.e. scan every document in a collection, to select those documents that match the query statement. If an appropriate index exists for a query, MongoDB can use the index to limit the number of documents it must inspect.

Indexes are special data structures that store a small portion of the collection's data set in an easy to traverse form. The index stores the value of a specific field or set of fields, ordered by the value of the field. The ordering of the index entries supports efficient equality matches and range-based query operations. In addition, MongoDB can return sorted results by using the ordering in the index.

Indexing makes the queries fast though making the data insertion, updation and deletion rate slow.

This happens because the index also needs to be updated following those operations which is an overhead.

Regex matching is done with the stored data in the MongoDB collection when a query is given.

It makes use of indexing which makes the data retrieval quite fast otherwise MongoDB would have to go through all the data items.

All these data insertion and query in MongoDB database is done via Python script.

A module called 'PyMongo' is used which is responsible for MongoDB database server connection and then inserting and displaying the related data to the standard output.

Data items were taken from all the files inside a directory in the linux system.

This was the hierarchy of the files located:

Suite/Date/Qor_files.

This means under the Suite directories, there were Dates folder and under the Dates folder were located the 'qor' files of that Suite on that particular Date.

This project progressed in two parts :-

- 1) Entering all the data located in the above mentioned directory at once.
- 2) When part 1 was completed, only those Dates of the Suite were chosen which was recently created and the same thing was done for the 'qor' files under the Date folder.

Part 1 ran for only once while part 2 had to be run continuously.

This ensured that the data items were not redundant and also the process time of part 2 was quite reduced.

CONCLUSION

It can be concluded that the application which was developed is working fine with no errors so far. Large scale testing was done before the product release to ensure smooth working of the product application.

The Regression Webpage which was developed by me from scratch is now being used by the company employees day to day providing information visually and making their work easy.

Also, I got to learn many new technologies like Python, JavaScript, MongoDB etc. in the process which I believe will surely prove beneficial to my technical areas.

Working in the Linux environment helped me get familiar with the Linux OS and its filesystem.

BIBLIOGRAPHY

- 1) *Learn Python the Hard Way* - Zed Shaw
- 2) *MySQL Cookbook* -Paul Dubois
- 3) *MongoDB in Action* - Kyle Banker
- 4) *MySQL* -<https://dev.mysql.com/doc>
- 5) *MongoDB* -<https://docs.mongodb.com>