CHAPTER – 7

SYSTEM ARCHITECTURE AND RESULT ANALYSIS

7.1 Inception

In the way of our research approach most of the theoretical assumptions are transformed as a mathematical working model. We aggregate different research contributions and investigate our problem in a systematic constructive manner which is accomplished as quantitative approach. We take a responsible task of data sampling and illustration of data with strong mathematical methods. The overview of our research results are evaluated form important considerable references which avoid duplication and disclose information genuinely.

7.2 Experimental Setup – About Python Language

We undertake our simulation process in window s10 environment of 64bit processor, 2Giga htz speed, RAM 4GB and secondary storage with 4 GB. Our auditor assessment research work is a result oriented where we used python programming language for results simulation. It is a troublesome task to exhibit the results computations manually.

Python programming language was proposed by "Guido Van Rossum"in 1991. Python code is readable and express simplicity programming style which supports many memory management, multiprogramming and object oriented programming concepts. Python have different versions which support functional programming; module programming, dynamic name resolution and map reduce filter methods. As python is portable and is used for rapid prototyping by the support of libraries. Python enables compilation of byte code with virtual machine that processes the interpreter utilities. As python language consists of minimum reserved words which have specific functions. Python supports indentation concept which supports the code in a better readable way. Python takes preliminary data types as Booleans, number and strings. The string operators in python can be used for slicing, repetition and concatenation. There are different string methods which help to transform the text in a user defined manner. Python supports different operators that is arithmetic, relational, assignment, logical, bitwise, membership and identity operators.

Python control flow statements support if, while and for loops which can be used as building blocks for different code segment executions. Python has advanced data structures where the data or objects can be placed in list, tuples, dictionaries and set. The small chunks of data in python which is used for a specific task is notified as a function where python consists of built in functions, recursive functions and fruitful functions. Python encompasses with many modules and packages which can supports for distributed computing environment. A test driven feature exhibits the python as a different modern programming language which supports GUI concept. [80]

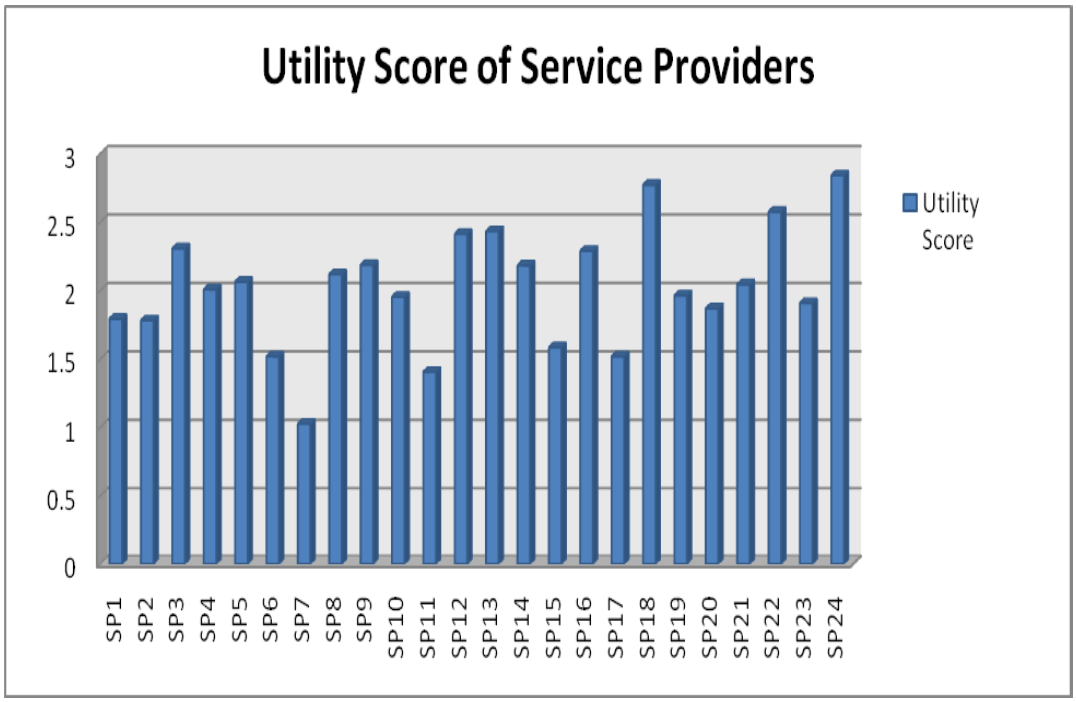
7.3 Experimental Results - Comparison

7.3.1 Tables & Graphs

SaaS_ID	Availability	Reliability	Cost($)	Response Time(ms)	Utility Score	GRADE ASSESSMENT
SP1	0.875	0.04761905	0.719402985	0.2	1.794403	18
SP2	0.64772727	0.07142857	0.062686567	0.1	1.7818424	19
SP3	0.27272727	0.28571429	0.949253731	0.8	2.3076953	6
SP4	0.875	0.33333333	0.949253731	0.8	2.0083333	13
SP5	0.54545455	0.0952381	0.823880597	0.6	2.0645732	11
SP6	0.59090909	0.19047619	0.537313433	0.2	1.5186987	21
SP7	0.31818182	0.5	0.208955224	0.8	1.027137	24
SP8	0.07954545	0.5952381	0.444776119	0.1	2.1195597	10
SP9	0.95454545	0.95238095	0.280597015	0.8	2.1875234	8
SP10	0.53409091	0.14285714	0.47761194	0.8	1.95456	15
SP11	0.38636364	0.5	0.519402985	0.8	1.4057666	23
SP12	0.79545455	0.61904762	0.062686567	0.8	2.4145022	5
SP13	0.01136364	0.88095238	0.752238806	0.8	2.4331912	4
SP14	0.14772727	0.78571429	0.850746269	0.4	2.1841878	9
SP15	0.01136364	0.71428571	0.459701493	0.4	1.5853508	20
SP16	0.42045455	0.54761905	0.52238806	0.8	2.2904617	7
SP17	0.46590909	0.4047619	0.247761194	0.4	1.5184322	22
SP18	1	0.28571429	0.486567164	0.1	2.7722814	2
SP19	0.375	0.5952381	0.194029851	0.8	1.9642679	14
SP20	0.36363636	0.52380952	0.582089552	0.4	1.8695354	17
SP21	0.86363636	0.16666667	0.614925373	0.4	2.0452284	12
SP22	0.54545455	0.64285714	0.388059701	0.1	2.5763714	3
SP23	0.67045455	0.04761905	0.591044776	0.6	1.9091184	16
SP24	1	1	0.441791045	0.4	2.841791	1

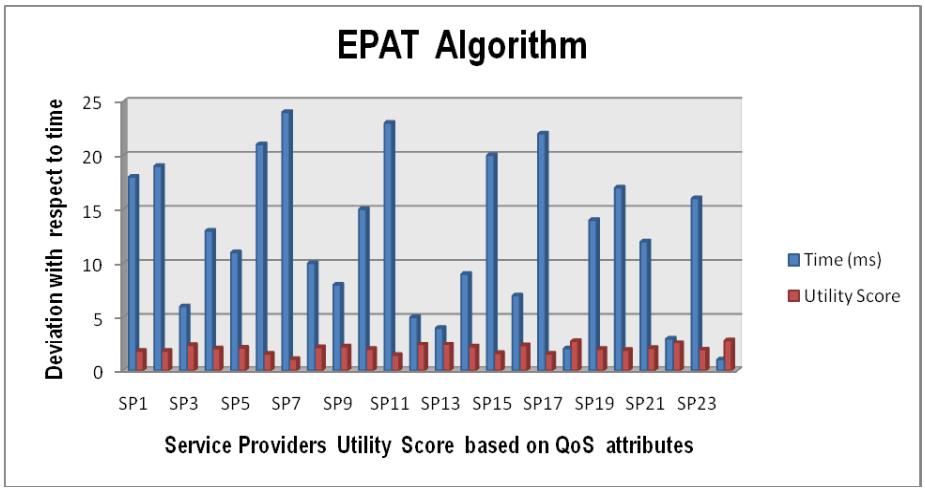
Table 7.3.1.1 Potential Service Providers for Economic QoS Assessment

From the above Table 3.1.1.1 Potential service providers of their quality of service offerings was evaluated to disclose the economic potential service provider by the audit trail assessment. As Availability and Reliability are utility driven attributes and Cost and Response Time are cost driven attributes computed separated by their specific formulas which are showcase in chapter 4. Finally the list of service providers with their QoS offerings are computed to find the Utility Score and Grade Assessment was complimented



Graph 7.3.1.1 Utility Score of Service Providers – EPAT algorithm
X-axis: List of Service Providers with their QoS Service Offerings
Y-axis: Utility Score

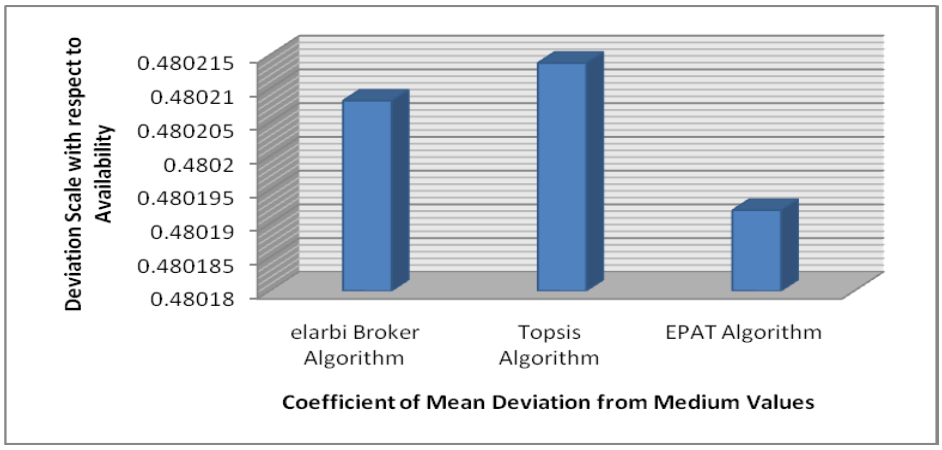
The above graph discloses the List of Service Providers with their QoS Service Offerings in X-axis and Utility Score in Y-axis. The Utility Score is calculated by categorization of attributes as Utility based and Cost based. In finding of the Utility score every attribute of each service provider was aggregated finally to evaluate the potentiality of that service provider related to economics.



Graph 7.3.1.2 EPAT Algorithm Utility Score and Time Consumed of Potential Service Providers

X-axis: List of Service Providers with their Utility Score and Time Consumed

Y-axis: Deviation with respect to time



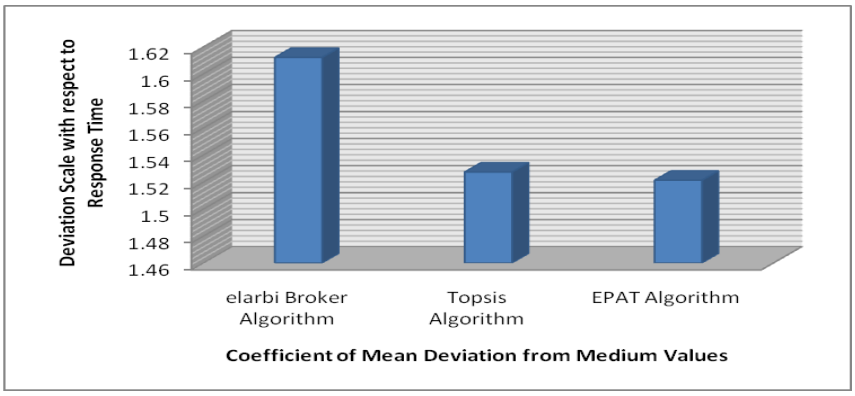
Graph 7.3.1.3 Coefficient of Mean Deviation from Medium Values for EPAT Algorithm attribute

X-axis: Different algorithm showcase distortions of data in finding QoS attribute

Y-axis: Deviation with respect to Availability

The above graph 7.3.1.2 show case potential service providers with Utility Score with time bounds on X-axis and Time Scale deviation on Y-axis with clear comparison.

The above graph 7.3.1.3 and below graph 7.3.1.4 show case disortion in data at finding portential service provider. In X-axis elarbi Broker Algorithm, Topsis Algorithm and EPAT Algorithm was strected and Deviation in disortion in data for Availability attribute Graph 7.3.1.3 and Response Time Graph 7.3.1.4 with respect to algorithms are depicted clearly for comparisons.



Graph 7.3.1.4 Coefficient of Mean Deviation from Medium Values for EPAT Algorithm attribute

X-axis: Different algorithm showcase distortions of data in finding QoS attribute

Y-axis: Deviation with respect to Response Time

SaaS_ID	Availability	Computed Sensitivity Value on Availability	SLA Value on Availability	Auditor Assessment Value on Availability
SP1	0.99988	0.999760014	0.989762414	0.990181178
SP2	0.99968	0.999360102	0.989366501	0.989785098
SP3	0.99935	0.998700423	0.988713418	0.989131739
SP4	0.99988	0.999760014	0.989762414	0.990181178
SP5	0.99959	0.999180168	0.989188366	0.989606888
SP6	0.99963	0.999260137	0.989267536	0.98968609
SP7	0.99939	0.998780372	0.988792568	0.989210922
SP8	0.99918	0.998360672	0.988377066	0.988795244
SP9	0.99995	0.999900003	0.989901002	0.990319825
SP10	0.99958	0.999160176	0.989168575	0.989587088
SP11	0.99945	0.998900303	0.988911299	0.989329703
SP12	0.99981	0.999620036	0.989623836	0.990042541
SP13	0.99911	0.998220792	0.988238584	0.988656704
SP14	0.99924	0.998480578	0.988495772	0.988914
SP15	0.99912	0.998240774	0.988258367	0.988676494
SP16	0.99948	0.99896027	0.988970668	0.989389097
SP17	0.99952	0.99904023	0.989049828	0.989468291
SP18	0.99999	0.99998	0.9899802	0.990399056
SP19	0.99944	0.998880314	0.98889151	0.989309906
SP20	0.99943	0.998860325	0.988871722	0.989290109
SP21	0.99987	0.999740017	0.989742617	0.990161372
SP22	0.99959	0.999180168	0.989188366	0.989606888
SP23	0.9997	0.99940009	0.989406089	0.989824702
SP24	0.99999	0.99998	0.9899802	0.990399056

Table 7.3.1.2 Audit Trail Assessment on Policy Parameter-Availability

The above Table 7.3.1.2 discloses the informatin about the Audit Trail Assessment on Policy Parameter – Availablity. As the auditor assessment is requested by th both parties consumer and provider, initially SLA policy value is assumed as 0.99 by them. The consumer urge is notified as sensitivity on particular attribute availablity is computed. Basing on sensitivity SLA value variation of Availablity is computed and finally auditor assessment value on that particular attribute was computed and tabulated for each service provider which has to be compared with Auditor assessment value of Policy Attribute. The below Table 7.3.1.3 also show case the same for Response Time.

SaaS_ID	Response Time(ms)	Computed Sensitivity Value on Response Time(ms)	SLA Value on Response Time(ms)	Auditor Assessment Value on Response Time(ms)
SP1	0.6	0.36	0.072	0.166153846
SP2	0.2	0.04	0.008	0.018461538
SP3	0.3	0.09	0.018	0.041538462
SP4	0.3	0.09	0.018	0.041538462
SP5	0.4	0.16	0.032	0.073846154
SP6	0.6	0.36	0.072	0.166153846
SP7	0.7	0.49	0.098	0.226153846
SP8	0.2	0.04	0.008	0.018461538
SP9	0.7	0.49	0.098	0.226153846
SP10	0.3	0.09	0.018	0.041538462
SP11	0.7	0.49	0.098	0.226153846
SP12	0.7	0.49	0.098	0.226153846
SP13	0.3	0.09	0.018	0.041538462
SP14	0.5	0.25	0.05	0.115384615
SP15	0.5	0.25	0.05	0.115384615
SP16	0.3	0.09	0.018	0.041538462
SP17	0.5	0.25	0.05	0.115384615
SP18	0.2	0.04	0.008	0.018461538
SP19	0.3	0.09	0.018	0.041538462
SP20	0.5	0.25	0.05	0.115384615
SP21	0.5	0.25	0.05	0.115384615
SP22	0.2	0.04	0.008	0.018461538
SP23	0.4	0.16	0.032	0.073846154
SP24	0.5	0.25	0.05	0.115384615

Table 7.3.1.3 Audit Trail Assessment on Policy Parameter-Response Time

SLA Pre-defined Policy Value for Utility driven attribute Availability of 0.99	
Average value of Utility driven attribute Availability	0.999577083
Auditor Assessment Value for Utility driven attribute Availability	0.998697083

Table 7.3.1.4 Computed Auditor Assessment values for Availability

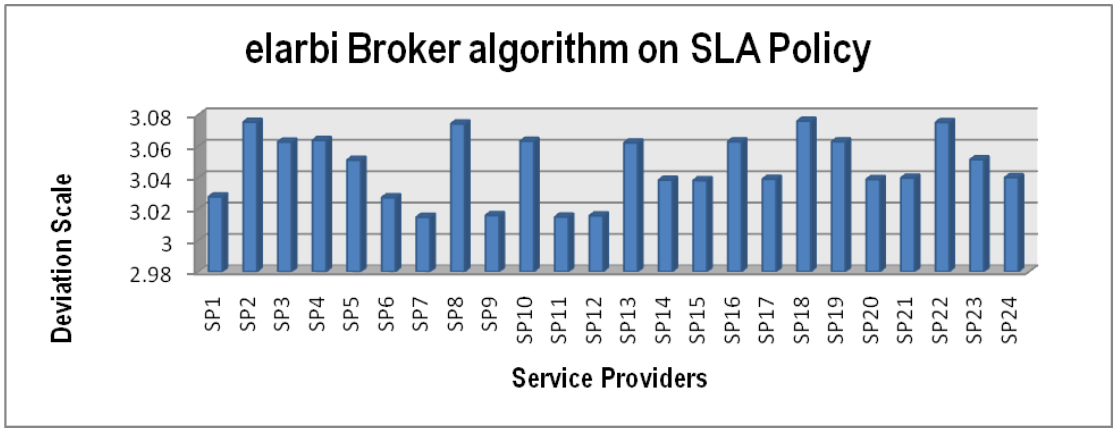
SLA Pre-defined Policy Value for Cost driven attribute Response Time(ms) of 0.2	
Average value of Cost driven attribute Response Time(ms)	0.433333333
Auditor Assessment Value for Cost driven attribute Response Time(ms)	0.216666667

Table 7.3.1.5 Computed Auditor Assessment values for Response Time

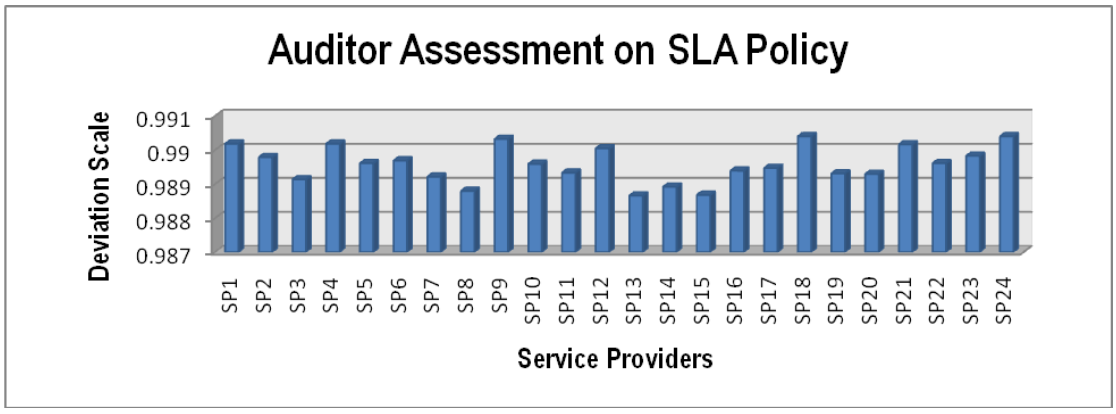
SaaS_ID	elarbi Broker algorithm on SLA Policy	Auditor Assessment on SLA Policy
SP1	3.027706005	0.990181178
SP2	3.075286013	0.989785098
SP3	3.062593027	0.989131739
SP4	3.063706005	0.990181178
SP5	3.051097017	0.989606888
SP6	3.027181016	0.98968609
SP7	3.014677026	0.989210922
SP8	3.074236034	0.988795244
SP9	3.015853002	0.990319825
SP10	3.063076018	0.989587088
SP11	3.014803023	0.989329703
SP12	3.015559008	0.990042541
SP13	3.062089037	0.988656704
SP14	3.038362032	0.988914
SP15	3.038110037	0.988676494
SP16	3.062866022	0.989389097
SP17	3.03895002	0.989468291
SP18	3.075937	0.990399056
SP19	3.062782024	0.989309906
SP20	3.038761024	0.989290109
SP21	3.039685005	0.990161372
SP22	3.075097017	0.989606888
SP23	3.051328013	0.989824702
SP24	3.039937	0.990399056

Table 7.3.1.6 Policy Assessment Comparison between Broker and Auditor

From the Table 7.3.1.4 and Table 7.3.1.5 show case the computed values of auditor with regard to policy parameter attributes Availability and Response Time. Where the assumed pre defined SLA values are also considered. These Tables form a benchmark value about the policy parameters and compared to the nearer values as best from the computed Tables 7.3.1.2 and Table 7.3.1.3



Graph 7.3.1.5 elarbi Broker algorithm on SLA Policy
X-axis: Service Providers Variations in Policy Parameter
Y-axis: Deviation with respect to Policy

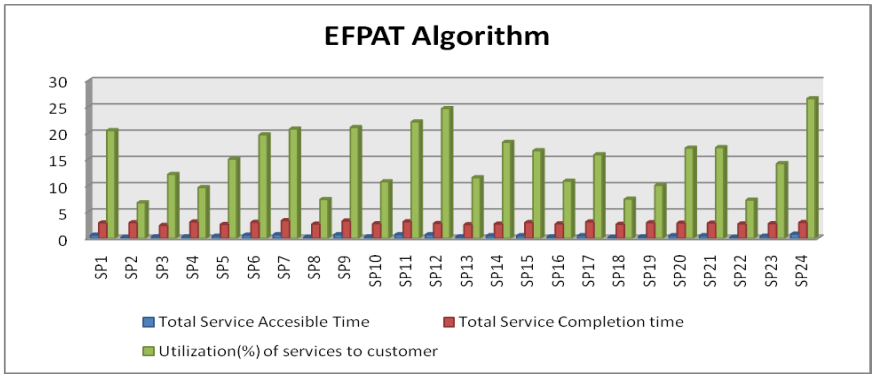


Graph 7.3.1.6 Auditor Assessment on SLA Policy – PPAT algorithm
X-axis: Service Providers Variations in Policy Parameter
Y-axis: Deviation with respect to Policy

The above Graph 7.3.1.5 and Graph 7.3.1.6 clearly picturized the comparison of elarbi Broker policy assessment and Auditor Assessment through PPAT algorithm about their undertaken policy with respect to effectiveness of time instance

SaaS_ID	Total Service Accesible Time	Total Service Completion time	AVG No. of Services by the provider	Utilization(%) of services to customer
SP1	0.6	2.92138	4.86896667	20.53823878
SP2	0.2	2.96121	14.80605	6.753995833
SP3	0.3	2.46697	8.22323333	12.16066673
SP4	0.3	3.10352	10.3450667	9.666443264
SP5	0.4	2.65113	6.627825	15.08790591
SP6	0.6	3.04321	5.07201667	19.71602354
SP7	0.7	3.3631	4.80442857	20.81412982
SP8	0.2	2.70493	13.52465	7.393906682
SP9	0.7	3.31585	4.73692857	21.11072576
SP10	0.3	2.78314	9.27713333	10.77919185
SP11	0.7	3.15516	4.50737143	22.18587964
SP12	0.7	2.83357	4.04795714	24.70381886
SP13	0.3	2.59898	8.66326667	11.54298994
SP14	0.5	2.73307	5.46614	18.29444544
SP15	0.5	2.99492	5.98984	16.69493676
SP16	0.3	2.75321	9.17736667	10.89637187
SP17	0.5	3.13719	6.27438	15.93782971
SP18	0.2	2.67761	13.38805	7.469347665
SP19	0.3	2.97319	9.91063333	10.09017251
SP20	0.5	2.91315	5.8263	17.16355148
SP21	0.5	2.89144	5.78288	17.29242177
SP22	0.2	2.74336	13.7168	7.290330106
SP23	0.4	2.80722	7.01805	14.24897229
SP24	0.8	3.00791	3.7598875	26.59654046

Table 7.3.1.7 Consumer Utilization of Provider Services with Time Bounds



Graph 7.3.1.7 EFPAT Algorithm on Utilization of Services

X-axis: Service Providers Time Variations on Service Utilization

Y-axis: Deviation with respect to Time

The above Table 7.3.1.7 show case the novel approach of efficiency computation on the providers services utilization by the consumer. This table show case the computed values of Total Service Acceble time of provider given to the consumer and Total Service Completion time given to the consumer. Finally Utilization of Consumer was disclosed.

7.3.2 Sample Source Code of EPAT, PPAT, EFPAT Algorithms

We simulate our research results in Python language, and we use Jupyter Notebook which can be avail as a open source application in web platform. This Jupyter Notebook used to create code instantaneously, it consists of built in libraries that support for visualizations and sharing of documents are also feasible and supported by colab.

Sample Code for EPAT algorithm

```
In [ ]: import pandas as pd from matplotlib import pyplot as plt import

In [ ]: 'drive/MyDrive/PHD/x.xlsx' as
numpy np In [ ]: from google.colab import drive drive.mount('/content/drive') In [ ]: excel_file = tocode = pd.read_excel(excel_file) # opening Excel File tocode

In [ ]: avail_maxi = tocode["Availability"].max() avail_mini = tocode["Availability"].min() avail_diff = round(avail_maxi - avail_mini response_diff))(str)) ,5print("Differrence Value of ResponseTime(ms) is : "+)
#Caliculating Maximum,Minimum,Diffrence for Availability


In [ ]: #Use this if you want to print("Maximum Value of Availability is : "+str(avail_maxi)) print("Minimum Value of Availability is : "+str(avail_mini)) print("Differrence Value of Availability is : "+str(avail_diff)) #Printing Maximum,Minimum,Diffrence for Availability

In [ ]: relia_maxi = tocode["Reliability"].max() relia_mini = tocode["Reliability"].min() relia_diff = round(relia_maxi - relia_mini,5) #Caliculating Maximum,Minimum,Diffrence for Reliability

In [ ]: print("Maximum Value of Reliability is : "+str(relia_maxi)) print("Minimum Value of Reliability is : "+str(relia_mini)) print("Differrence Value of Reliability is : "+str(relia_diff))

In [ ]: cost_maxi = tocode["Cost($)"].max() cost_mini = tocode["Cost($)"].min() cost_diff = round(cost_maxi - cost_mini,5) #Caliculating Maximum,Minimum,Diffrence for Cost($)
```

```
In [ ]: print("Maximum Value of Cost($) is : "+str(cost_maxi)) print("Minimum Value of Cost($) is : "+str(cost_mini)) print("Differrence Value of Cost($) is : "+str(cost_diff))

In [ ]: response_maxi = tocode["Response Time(ms)"].max() response_mini = tocode["Response Time(ms)"].min() response_diff = round(response_maxi - response_mini,1) #Caliculating Maximum,Minimum,Diffrence for ResponseTime(ms)

In [ ]: print("Maximum Value of ResponseTime(ms) is : "+str(response_maxi)) print("Minimum Value of ResponseTime(ms) is : "+str(response_mini

In [ ]: avail_maxi = tocode["Availability"].max() avail_mini = tocode["Availability"].min() avail_diff = round(avail_maxi - avail_mini ,5)
#Caliculating Maximum,Minimum,Diffrence for Availability

In [ ]: #Use this if you want to print("Maximum Value of Availability is : "+str(avail_maxi)) print("Minimum Value of Availability is : "+str(avail_mini)) print("Differrence Value of Availability is : "+str(avail_diff)) #Printing Maximum,Minimum,Diffrence for Availability

In [ ]: relia_maxi = tocode["Reliability"].max() relia_mini = tocode["Reliability"].min() relia_diff = round(relia_maxi - relia_mini,5) #Caliculating Maximum,Minimum,Diffrence for Reliability

In [ ]: print("Maximum Value of Reliability is : "+str(relia_maxi)) print("Minimum Value of Reliability is : "+str(relia_mini)) print("Differrence Value of Reliability is : "+str(relia_diff))
In [ ]: cost_maxi = tocode["Cost($)"].max() cost_mini = tocode["Cost($)"].min() cost_diff = round(cost_maxi - cost_mini,5) #Caliculating Maximum,Minimum,Diffrence for Cost($)
In [ ]: print("Maximum Value of Cost($) is : "+str(cost_maxi)) print("Minimum Value of Cost($) is : "+str(cost_mini)) print("Differrence Value of Cost($) is : "+str(cost_diff))
In [ ]: response_maxi = tocode["Response Time(ms)"].max() response_mini = tocode["Response Time(ms)"].min() response_diff = round(response_maxi - response_mini,1) #Caliculating Maximum,Minimum,Diffrence for ResponseTime(ms)

In [ ]: print("Maximum Value of ResponseTime(ms) is : "+str(response_maxi)) print("Minimum Value of ResponseTime(ms) is : "+str(response_mini response_diff)) (str)) print("Differrence Value of ResponseTime(ms) is : "+

In [ ]: tocode["Obtained Reliability"] = (tocode["Reliability"] - relia_mini)/relia_diff
In [ ]: tocode["Obtained Cost"] = (cost_maxi - tocode["Cost($)"])/cost_diff

In [ ]: tocode["Obtained ResponseTime"] = (response_maxi - tocode["ResponseTime(ms)"])/response_diff

In [ ]: tocode["Utility Score"] = tocode["Availability"] + tocode["Reliabil ity"] + tocode["Cost($)"] + tocode["Response Time(ms)"]

In [ ]: tocode["Grade Assessment"] = tocode["Utility Score"].rank(ascending=0)

In [ ]: tocode["Utility Score"] = tocode["Obtained Availability"] + tocode[
"Obtained ResponseTime"]+tocode["Obtained Reliability"]+tocode["Obtained Cost"]tocode
```

Sample Code for PPAT algorithm

```
In [ ]: import pandas as pd import numpy from matplotlib import pyplot as plt import numpy as np

In [ ]: from google.colab import drive drive.mount('/content/drive')
In [ ]: excel_file = 'drive/MyDrive/PHD/y.xlsx' tocode = pd.read_excel(excel_file) # opening Excel File tocode
In [ ]: Average_Availability_SaaS_ID = tocode["Availability"].mean(axis=0) #Average_Availability_SaaS_ID # calculating Average of Availability of SaaSID
In [ ]: tocode['SP(Availability)Sensitivity ^^2'] = tocode['Availability']** 2 #calculating SP(Availability)Sensitivity ^^2
In [ ]: tocode['SLA * SP(Availability)Sensitivity ^^2'] = tocode['SP(Availab ility)Sensitivity ^^2']*0.99 #calculating SLA * SP(Availability)Sensitivity ^^2

In [ ]: tocode['SLA* SP(Availability)Sensitivity ^^2 / Average Availability of SaaS_ID'] = tocode['SLA * SP(Availability)Sensitivity ^^2']/Aver age_Availability_SaaS_ID #calculating SLA* SP(Availability)Sensitivity ^^2 / Average Availabi lity of SaaS_ID

In [ ]: tocode['Percentage of Availability'] = numpy.floor(((tocode['SLA* SP (Availability)Sensitivity ^^2 / Average Availability of SaaS_ID']/0. 99)*100)) #calculating Percentage for Availability table

In [ ]: tocode['SP(Response Time) Sensitivity ^^2'] = tocode['Response Time (ms)']**2 #calculating SP(Response Time) Sensitivity ^^2

In [ ]: Average_Response_Time_of_SaaS_ID = tocode["Response Time(ms)"].mean( axis=0) #calculating Average_Response_Time_of_SaaS_ID In [ ]: tocode['SLA* SP(Response Time) Sensitivity ^^2'] = tocode['SP(Respo nse Time) Sensitivity ^^2']*0.2 #calculating SLA* SP(Response Time) Sensitivity ^^2

In [ ]: tocode['SLA* SP(Response Time) Sensitivity ^^2 / Average Response T ime of SaaS_ID'] = tocode['SLA* SP(Response Time) Sensitivity ^^2'] /Average_Response_Time_of_SaaS_ID #calculating SLA* SP(Response Time) Sensitivity ^^2 / Average Respo nse Time of SaaS_ID

In [ ]: response_table = tocode[["SaaS_ID","Response Time(ms)","SP(Response Time) Sensitivity ^^2","SLA* SP(Response Time) Sensitivity ^^2", "SLA* SP(Response Time) Sensitivity ^^2 / Average Response Time of SaaS_ID","Percentage of Response time"]] #Total response Table Calculated and printed response_table

In [ ]: tocode['SP(Cost) Sensitivity ^^2'] = tocode['Cost($)']**2 #calculating SP(Cost) Sensitivity ^^2

In [ ]: Average_Cost_of_SaaS_ID = tocode["Cost($)"].mean(axis=0) #calculating Average_Cost_of_SaaS_ID

In [ ]: tocode['SLA* SP(Cost) Sensitivity ^^2'] = tocode['SP(Cost) Sensiti vity ^^2']*0.5 #calculating SLA* SP(Response Time) Sensitivity ^^2 In [ ]: tocode['SLA* SP(Cost) Sensitivity ^^2 / Average Cost of SaaS_ID'] = tocode['SLA* SP(Cost) Sensitivity ^^2']/Average_Cost_of_SaaS_ID #calculating SLA* SP(Response Time) Sensitivity ^^2 / Average Respo nse Time of SaaS_ID In [ ]: Cost_table = tocode[["SaaS_ID","Cost($)","SP(Cost) Sensitivity ^^2" ,"SLA* SP(Cost)
```

Sample Code for EFPAT algorithm

```
In [ ]: import pandas as pd from matplotlib import pyplot as plt import numpy as np

In [ ]: from google.colab import drive drive.mount('/content/drive')

In [ ]: excel_file = 'drive/MyDrive/PHD/z.xlsx' tocode = pd.read_excel(excel_file) # opening Excel File
In [ ]: tocode
In [ ]: tocode["Total Service Accesible Time"] = (tocode["Service Accessible time for Service A"] + tocode["Service Accessible time for Service B"] ) # Calculating the Total Service Accesible Time
In [ ]: tocode["Total Service Completion time"] = (tocode["Service Completio n Time for Service - A"] + tocode["Service Completion Time for Servi ce - B"])
In [ ]: tocode["AVG No. of Services by the provider"] = tocode["Total Servic e Completion time"]/tocode["Total Service Accesible Time"]

In [ ]: tocode["Utilization(%) of services to customer"] = (1/tocode["AVG N o. of Services by the provider"])*100

In [ ]: Final_Table = tocode[["SaaS_ID","Service Accessible time for Service A","Service Accessible time for Service B","Total Service Accesible Time","Service Completion Time for Service - A","Service Completion Time for Service - B","Total Service Completion time","AVG No. of Se rvices by the provider","Utilization(%) of services to customer"]]
Final_Table
```

7.4 Chapter Summary

The showcasing of results carried in this chapter demonstrated in a systematic manner. The introduction approach of Python language was lucidly explained. The outcomes of our algorithms EPAT with detail computations are tabulated and comparison graphs are structured with explanation. The outcomes of PPAT and EFPAT algorithms with comparison graphs depicted and explanation text was presented. A sample code of these three algorithm show case as a reference to our simulation process.