The International Journal of Engineering and Science (IJES)
|| Volume || 11 || Issue || 2 || Series I || Pages || PP 01-07 || 2022 ||
ISSN (e): 2319-1813 ISSN (p): 20-24-1805

A Framework for Stress-Testing of Cloud-basedApplications

EchezonaStephenson C.[1],Okereke George E.[2],NwagwuHonour C.[3]
Senior Lecturer Department of Computer Science, University of Nigeria, Nsukka, Enugu State, Nigeria[1]
Senior Lecturer Department of Computer Science, University of Nigeria, Nsukka, Enugu State, Nigeria[2]
Lecturer I,Department of Computer Science, University of Nigeria, Nsukka, Enugu State, Nigeria[3]

----------*ABSTRACT*----------
Some cloud-based systems albeit internet-based systems suffer delays and disruptions in processing when in full use or large number of users are online. Sometimes the application may not readily respond to requests from the users (cf. denial of service – DOS) or some boxes may not present properly leading to the usual frustrations when it occurred. This lethargic behavior may be as a result of unavailability of facilitiesto carry the load. Such facilities like bandwidth and latency if not properly checked may lead to such disruptions of service. This paper therefore, tries to predict through testingby building a mathematical relations to predict the behavior of cloud-based systems during development and final production but before deployment, to gauge the behaviour of the system when an extreme number of usersare online and predict when the system is expected to demand additional resources. This paper was a result of original research for a higher degree conducted by the authors.
Keywords*:Domain (Cloud Computing), Cloud-based systemsResponse time, Black-box testing, CloudAnalyst, Workload, Behavioural testing.*

Date of Submission: 05-02-2022Date of Acceptance: 18-02-2022

I. INTRODUCTION

Naturally, software developers perform testing of their software either by themselves or by outsourced testers. They employ many software testing strategies, such as, White box testing by inspecting the codes to make sure that boundary values and unclear codes are not missed [1], as well as, Black box testing or behavioural testing where the software is more-or-less validated (that is, ensure that the software is doing what it expected to do) [1],[2]. Both testing strategies are adjudged relevant. The test plans available in black box testing include equivalent partition, boundary value analysis, stress testing and error guessing [2],[3].

Equivalent Partitioning involves dividing all possible inputs into classes (partitions), such that there are a finite number of input equivalent classes. In equivalent partitioning you may reasonably assume that:

(i) The program behaves analogously for inputs in the same class.
(ii) A test with a representative value from a class is sufficient.
(iii) If representative detects fault then other class members will detect the same fault. [1],[2],[3]

Boundary Value Analysis is based onheuristic experience:

- Testing boundary conditions of equivalence classes is more effective, ie., values directly on, above and beneath edges of equivalent classesare tested [1],[2],[3].

Example strategy as extension of equivalence partitioning

- Choose (k) arbitrary values in each equivalent class
- Choose values exactly at the lower and upper boundaries of equivalent classes.
- Choose values immediately before and above each boundary (if applicable) [3].

Stress testing tests extreme number of users online in a system concurrently thereby giving room to test delays as they struggle for limited resources [4],[5]. Because at development stage it is not possible to test extreme number of users, mathematical functions were developed to estimate these factors.

It is discovered that most software developers do not go the extra mile, to determine the number of users expected in their new system and the capacity of resources that the system needs to function effectively. This leads to the usual delays and denial of service syndrome in most applications online. Hence, such systems may experience slowing down when extreme number of users are online.

Existing systems apply a considerable amount of load to the system under testing or its components to gauge how the system behaves, [6]. If for instance the CPU is under testing, extreme workload is applied to check how the CPU copes. Whether it fails gracefully, or simply results in loss of data, etc. Similarly, in the existing system, testing for memory sees extreme software loaded in memory. Some of the tools used to load the CPU component and test its other parts are: Prime 95,SiSoft 2012, or Pass mark Burnin, etc., [7].

This defacto stress-testing method is very cumbersome as you have to obtain enough loads to reach breaking point of your application. This intuitive framework developed will enable faster testing and generate results that are particular to the software under testing. It promises reduction in cost of testing.

To use this system, the tester only need to generate an initial representative points, convert these points to an equation, then a software that generates other points outside the initial points are coded and the result of the software is charted using say, MS Excel to clearly interpret these results.[8], wrote that a number of cloud platforms and services have been developed for data intensive computing. Open Cloud Testbed (OCT) was designed and implemented to create standard for performance of these systems. Also, [9] wrote that Open Cirrus is a cloud-computing testbed that federates distributed data centres. It aims to spur innovation in systems and applications research and catalyze development of an open source service stack for the cloud. [10] wrote that apart from being less demanding on hardware and software investment, mathematical modeling and analysis may also be attractive for providing an understanding of the interdependencies involved in cloud computing. It is particularly suitable for identifying optimal values and equilibria and predicting behaviour.[11], recommended 5 test users in a usability study. He argued that this will let you find almost as many usability problems as you would find using many more test participants. 5 users get close to user testing maximum benefit cost ratio.

II. MATERIALS AND METHODS

System is tested firstly to remove bugs and secondly to comply with requirement specifications. It has been stated that the well-known testing methods that achieve the two objectives are blackboard and whiteboard testing. The developer of the system can easily perform white box testing with the knowledge of the underlying codes. While a number of tools are employed to test the behaviour of the system, such as, Jmeter, Pylon, [12],

A.Stress Testing

i)Testobjectives:The listed are the specific objectives for stress testing, according to [13].

1. To identify the ways the system can possibly fail catastrophically in production?
2. To provide information to the team to possibly build defenses against catastrophic failures?
3. To identify how the application behaves when system resources such as memory, disk space, network bandwidth, or processor cycles are depleted?
4. To ensure that functionality does not break under stress? Example, orders are not inserted in the database, the application is not returning the complete product information in searches, form controls are not being populated properly, redirects to custom error pages are occurring during the stress testing, and so on.

A Scenario for this test is the protracted delay in response to user requests, example download or upload of say courseware. One of the stress related problems is application component refusing to respond.

Next, the workload applied to a system to be tested will stress the system beyond threshold level and allow for the observation of the stress consequences that may arise. The problem is to identify the load at which the system starts to show signs of stress. Ideally, to achieve this, it is recommended that actual load be applied systematically in increasing proportions until the system begins to behave abnormally. However, this style of testing may not be possible as the tester will not have the required data at testing stage. Hence, a simulated workload becomes inevitable. The framework presented uses mathematical relations to project the peak/stress point of a system.

The metrics considered for this testing have been narrowed down to mostly production/business which are, [1], [2], [3]:

- Transactions/sec
- Transactions succeeded
- Transactions failed
- Requests succeeded
- Requests failed
- Contentions per second
- Deadlocks
- Thread allocation
- Transactions times

A test case is how long it takes 6,000 users to upload/download a courseware or publication of say 25Mb. The worst case should be beyond 5 seconds.

a. Test Plan background

A cloud based resources of higher institutions of learning in Nigeria (ERIS), was created and deployed in the cloud (as part of a Doctoral research which basically uploads and downloads courseware to/from the cloud) to be used as a test bed. Major operations such as upload and download times are of interest to the users.

The experiment expects numerous users to access the cloud artifact. The major operations expected are downloading and uploading contents to and from the artifact. Because certain network considerations are not at the disposal to the tester, testing of the specific elements such as bandwidth, latency, and hoping of packets are subsumed in the testing of the response time of the requests.

B. EXPERIMENT

The resources from the n participating institutions are closed under update and retrieval operations.For the 3 resources used as sample autonomous resources existing in a federation, researcher tested each resource separately by accessing its contents independently and later collectively to ensure compliance to boundary problem.

Queries that divulge to single user in the federated systemis performed, and the time the result was obtained was recorded. Repeat for queries that access/obtain information for two users, record the time and once more repeat for queries that access/obtain results for three users, repeat and record this time intervals for up to 5 users. (Ideally, these tests are performed many timessay (5 times each) and averages recorded to eliminate chance events.). See table I. Then, use numerical method technique to obtain equations governing these 5 points, apply extrapolation technique to determine the sixth time interval. Repeat sequentially for 5 through n, where n is up to 10,000 (10 thousand). Plot the time differences against the users. This will give an indication of the stress level behaviour if very many users are online the ERIS at the same time.

i)Test Results: Table Icontain the test results obtained from ERIS for:

- Upload Time (Ti). This is the time (in microseconds) taken for 1 to 5 users to upload data to the cloud simultaneously.
- Retrieval/Download time (Ri). This is the time (in micro seconds) for retrieval/download of data for 1 to 5 users simultaneously using ERIS.

The upload (Ti) and the download (Ri) times refers to exact time between request and delivery as returned by the test application.

Table I: Data obtained from running the ERIS software

Ni/Ri/Ti

Ni	***1***	***2***	***3***	***4***	***5***
Ri	***10+0.03569006E-6***	***10+0.03569006E-6***	***10+0.03569007E-6***	***10+0.03569008E-6***	***10+0.03569008E-6***
Ti	***10+0.02716183E-6***	***10+0.02716183E-6***	***10+0.02716184E-6***	***10+0.02716185E-6***	***10+0.02716185E-6***

Table I describes the raw data (upload and download times) using three institutions' resources in the cloud with minimal users accessing the artifact. The behaviour of the system with more than three institutions joining ERIS tend to produce more traffic which by converting the relationship in table I to equation using Lagrange, [9], yield the following equations.Note that a user's query accessing the artifact in the cloud will undergo these three processes: Waiting time, Connection time and Session time. The waiting time and the connection time were found to be relatively constant (10micro-seconds) while the session time varied. The addition of the waiting time, the connection time and the session time gives the total time for upload or download. This paper therefore only factored the session time to an equation that can help gauge the behaviour of the system.

$F(r) = P(r)$

Where $P(r) = \sum l_i(r)\, f(x_i), \quad i = 0 \ldots n$

When:

$$i = 0 \rightarrow l_0 = \frac{(r-r_1)(r-r_2)(r-r_3)(r-r_4)}{(r_0-r_1)(r_0-r_2)(r_0-r_3)(r_0-r_4)}$$

$$i = 1 \rightarrow l_1 = \frac{(r-r_0)(r-r_2)(r-r_3)(r-r_4)}{(r_1-r_0)(r_1-r_2)(r_1-r_3)(r_1-r_4)}$$

$$i = 2 \rightarrow l_2 = \frac{(r-r_0)(r-r_1)(r-r_3)(r-r_4)}{(r_2-r_0)(r_2-r_1)(r_2-r_3)(r_2-r_4)}$$

$$i = 3 \rightarrow l_3 = \frac{(r-r_0)(r-r_1)(r-r_2)(r-r_4)}{(r_3-r_0)(r_3-r_1)(r_3-r_2)(r_3-r_4)}$$

$$i = 4 \rightarrow l_4 = \frac{(r-r_0)(r-r_1)(r-r_2)(r-r_3)}{(r_4-r_0)(r_4-r_1)(r_4-r_2)(r_4-r_3)}$$

Substituting for ri in table I and multiplying out with $f(x_i)$ given in the table resulted to equation (1). Repeating similar operations for ti yielded equation 2. Then applying Rechardson's extrapolation technique, [14], when implemented in a program using the technique of table II and plotting using MS Excel, gave rise to figures 1 and 2.

Equation 1 for Ri using Lagrange's method

$0.0029747(r^4)- 0.00327149(r^3) + 0.1754927(r^2) – 1.3074241(r) +0.1962884$ . . … .. 1

Equation 2 for Ti using Lagrange's method

$0.002263(t^4) – 0.0300808(t^3) + 0.253736(t^2) – 0.349689(t) + 0.0770296$ 2

Where Ristands for Retrieve/Download time.

Ti stands for Upload time

With these two equations, Richardson's Extrapolation technique, given as equation 3

$Nj(k) = (4^{j-1} N_{j-1}(h/2) – N_{j-1}(h)) / (4^{j-1} – 1)$ 3

for j = 1,2,3, . . ., m

Where $N_1(h) = ½*h[f(x0+h) – f(x0 - h)]$

$N_1(h/2) = 1/4*h[(f(x0 +h) – f(x0 –h)]$

$N_1(h/4)=1/8*h[(f(x0+h) – f(x0 – h)]$

Suppose for a hypothetical case of x0 = 2, h = 0.2 and $f(x) = xe^x$

Table II shows the steps of Richardson's extrapolation.

Table II: Steps of Richardson's extrapolation

$N_1(0.2) = 22.414160$

$N_2(0.2)=(4N_1(0.1)–N_1(0.2))/3=22.166995$

$N_1(0.1) = 22.228786$ **$N_3(0.2)=(16N_2(0.1)-N_2(0.2))/15=22.167168$**

$N_2(0.1)= 4N_1(0.1)-N_1(0.2))/3=22.167157$

$N_1(0.05)=22.182564$

The sample algorithm to compute the Rechardson's extrapolation model is given in algorithm 1. It is to be noted that only lines 5 and 6 of the algorithm need to change for derived equations of different application under testing.

```
1       To approximate the next point N5+k outside the known points (Ni for i =1, . . . , 5)
2       INPUT: Known end points N1, N2, N3, N4, N5
3       OUTPUT: Approximate Ri, Ti for upload and download times respectively
4       Step1   set  h = 0.2, Xo  =  200 up to 10000
5               F(Ri)  =  0.0029747(r^4)- 0.00327149(r^3) + 0.1754927(r^2) – 1.3074241(r) +0.1962884
6               F(Ti)  =  0.002263(t^4) – 0.0300808(t^3) + 0.253736(t^2) – 0.349689(t) + 0.0770296
7       Step 2  For i 200 to 10000 step 200
8               Set y[i]  = [ funct( f1(Ri)) – funct(f2(Ri)) ]
9               Set  y2[i] =  [ funct(f3(Ri)) – funct(f4(Ti)) ]
10      Step 3  for t = 2  to 5
11              Begin
12                      Set f = 2
13                      For j =  2, m = t;  m<= 5
14                      Begin
15                              Set yk  =  (4^(t-1) * y[ j –f + 2]  -  y[ f - j + 1]) / ( 4^(t-1) – 1)
16                              Set y2k = (4^(t-1) * y2[ j –f + 2]  -  y2[ f - j + 1]) / ( 4^(t-1) – 1)
17                      End (* for  j loop *)
18              End  (* for t loop *)
19      Step 4 Output Yk and Y2k   //yk represents download time Ri
20      End  (* for i loop *)              //y2k represents upload time  Ti
21      STOP
```

Algorithm 1: Algorithm of Richardson's Extrapolation technique for the evaluation of the download and upload times using ERIS cloud application.

III. RESULTS

Figures 1 and 2 are Excel charts of the data generated from the algorithm I, depicting the behaviour of the system when it goes live/when stressed. This is tested between x0 = 200 and 10,000 (difference of 200) users online at same time and h = 0.2. Figure 1 shows that the time difference is not linear. There are periods when appreciable changes occur, implying that it is at those points that the system demands expansion. This is therefore the point when decisions as to possible need to provision more resources to the system are made.

The code for the implementation of extrapolation was written in Java programming language. Considering the result of the test experiment, at key points between 200 users and 10,000 users, the differences in time are within acceptable significance, which means that such systems will behave well even when stressed at 6,200 users online simultaneously.

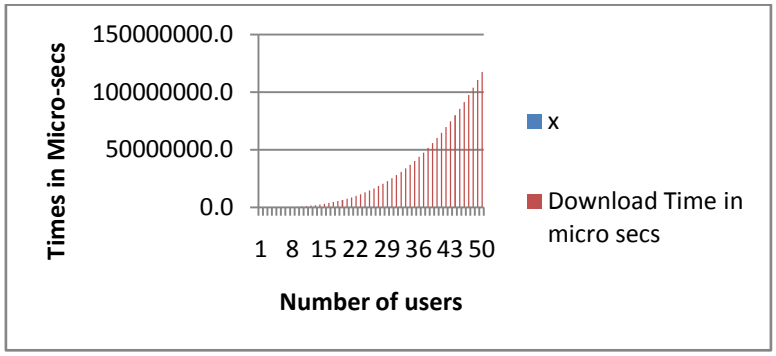

Fig 1: Bar chart of changes in Ti with more users (upload)

Fig.1 shows a bar chart illustrating the number of users on the horizontal axis against the estimated times to finish an upload at the vertical axis. The horizontal axis figures are in multiples of 200 users. The bar chart indicates that between 200 and 6200 users; the behaviour is uniformly tolerable, suggesting that no special added resource requirement is needed. But beyond 6200 there is a significant increase in demand of resources (eg, bandwidth, network, etc.).

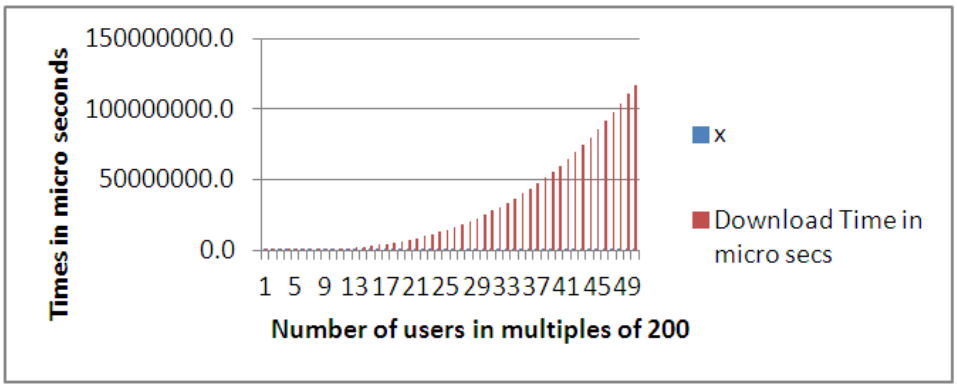

Fig. 2: Bar chart of changes in Number of users against time (download).

Almost the same behavior is exemplified as number of users gets to and beyond 5800.

IV. DISCUSSION

Fig.2 reflects the same behaviour as fig.1 for retrieval/download times. The two graphs may look similar considering that both upload and download times increases as the number users increase but not at the same rate as a closer observation will reveal. At 50,000,000 microseconds, upload can accommodate approximately 6200 users (ie., 31 * 200), see figure 1. While download is approximately at 5,800 users (ie., 29 * 200), see figure 2.

Based on the uniform behaviours of this experiment, one can estimate the average infrastructure requirements such as bandwidth thus:

Suppose on average 25 kilobytes of data is involved on every courseware/publications upload/download per second and that the system will maintain an estimated one thousand (1000) users per second nation-wide/worldwide at the same time. Thiswill give an estimated 25000kb per second or approximately 25,000,000 bytes per second. This gives 200,000,000 bits per second or approximately 200Mbits per second bandwidth requirement.

This paper reveals a novel system of predicting the behaviour of a cloud/internet system when an extreme number of users are online and help gauge when the system may require providing more resources to continue to run well. This prediction is quickly elicited before deployment.

DECLARATIONS

List of Abbreviations

Not applicable

Availability of data and materials

Yes, see attached table.

Competing interests

Not applicable

Funding

Not applicable

Authors Contributions

The first author initiated the idea and developed the introduction and the mathematical application during his doctoral research. The second author situated the work with respect to the current works. The threeauthorsdid used the data to develop Excel charts (Fig. 1 and Fig. 2). The fourth author checked and developed the reference.

The four authors together developed the algorithm and Java language implementation.

Acknowledgements

Not applicable

Authors Information

Echezona S. C. is a Senior Lecturer, he had his Ph.D. In Cloud Computing in April, 2017.

Okereke G.E. is a Senior Lecturer, he had his doctorate in Computer Engineering.

Nwagwu H. C. is a lecturer I, he had his doctorate degree in Hallam Sheffield University, UK.

REFERENCES

[1]. Thomas Ostrand, (2018), Software Engineering online library,Wiley.com, John Wiley and sons Inc. pp.28, 2018.

[2]. Heshan Du, SuchithAnand, Natasha Alechina, Jeremy Morley, Glen Hart, Didier Leabovic, Mich Jackson, and Mark Ware, (2012), Geospatial Information Integration for Authoritative and Crowed Sourced Road Vector Data, Transaction in G13, 16,4 (457), 2012.

[3]. DadeauF.Giorgetti A, Bouguet F, and Enderlin I, (2018), Contrat based Testing for PHP with prapet, Journal of Systems and Software, 136; C, (210). Online publication date: 1st February, 2018.

[4]. Karpov Y, Karpov L, and Smetanin Y, (2018), Adaptation of General concepts of Software Testing to Neural Networks, Programming and Computing Software, 44:5, Online publication date 1-Sep-2018

[5]. Kazmi R, Jawawi D, Mohamad R and Gani I, (2017), Effective Regression Test Case Selection, ACM Computing Surveys, 50:2 (2), Online publication date: 19- Jun-2017

[6]. Nelson Wayne B., (2004), Accelarated Testing – Statistical Models, Test Plans and Data analysis, John Wiley and Sons, New York ISBN 0-471-69736-2

[7]. Juan Jose Guerrero III – ASUS 92012-03-29), "Intel X 79 Mother board Overlocking Guide" benchmarkreviews.com, Reviewed 2 February, 2013

[8]. R. L. Grossman, Y. Gu, M. Sabala, C. Bennet, J. Seidman, and J. Mambretti, "The Open Cloud Testbed: a wide area testbed for cloud computing utilizing high performance network services," in *Gridnets 2009*, Sep. 2009, pp. 89–97.

[9]. Roy Cambell, Indranil Gupta, Micheal T. Heath, Steven Y. Ko, ., "Open Cirrus Cloud Computing testbed: federated data centers for open source systems and services research," in *USENIX Hotcloud '09*, June 2009, pp. 1–5.

[10]. G. Sakellari and G. Loukas, "A survey of mathematical models, simulationapproaches and testbeds used for research in cloud computing,"*Simulation Modelling Practice and Theory*, vol. 39, no. 0, pp. 92–103,Dec. 2013.

[11]. Jakob Nielsen, "How Many Test Users in a Usability Study?" https://www.nngroup.com/articles/how-many-test-users, on June 4, 2012

[12]. http://stackoverflow.com/questions/7492/performing-a-stress-test-on-web-application, Performing a stress test on web applications, pp. 1.

[13]. J.D. Meier, Carlos Farre, PrashantBansode, Scott Barber, and Dennis Rea, "Performance Testing Guidance for Web Applications", Microsoft Corporation, September 2007, pp. 1-2.

[14]. Burden L. R., Faires J. D, (1988), Numerical Analysis , *PWS-KENT Publishing Company*, Boston, Foutth Edition, pp 157-160.

FIGURE TITLES AND LEGENDS

1. Fig 1: Bar chart of changes in Ti with more users (upload)

Fig. 1 Shows the bar chart illustrating the number of users on the horizontal axis against the estimated time to finish an upload at the vertical axis. The horizontal axis figures are in multiples of 200 users. The bar chart indicates that between 200 and 6200 user, the behaviour is uniformly tolerable suggesting that no added

resource requirement is needed. But beyond 6200 there is a significant increase in demand of resources (eg. bandwidth,etc.).

2. Fig. 2: Bar chart of changes in Number of users against time (download).

Fig. 2 reflects the same behaviour as Fig. 1 for retrieval/download times. The two graphs may look similar considering that both upload and download times increases as the number of users increase but not at the same rate as a closer observation will reveal. At 20,000,000 microseconds, upload can accommodate approximately 6,200 users (ie., 31 * 200). While download is approximately 5, 800 users (ie., 29 * 200) for the same 20,000,000 microseconds.

EchezonaStephenson C, et. al. "A Framework for Stress-Testing of Cloud-based Applications."*The International Journal of Engineering and Science (IJES)*, 11(2), (2022): pp.