Skimming of Test-Case Execution Bandwidth on a Balanced Cloud Service Through An Worker Role With Logistic Regression Determination

K. Vivekanandan[1], G.Keerthi Lakshmi[2]

Abstract

Cloud environments pose new challenges to the traditional testing techniques by means of architectural challenges, effective utilization of bandwidth available and security. Effective bandwidth utilization is regarded as the most important challenge to the testers while designing the test cases that are meant to run on the live environment since unconcentrated test case design yields in more compute hours and plays a role in QoS reduction to the real customers for cross bandwidth utilization. Hence an attempt has been made in this paper to skim the execution bandwidth utilized on a cloud service that is running across multiple servers across the globe by imparting the logistic regression method. The tests are run against the cloud services hosted on windows azure cloud platform and bandwidth utilized with regards to compute hours are measured against the bandwidth used on normal case without the worker role involving in the bandwidth reduction and about 52% reduction overall is achieved with 93% of confidence level.

Keywords: — Software Testing, Test case optimization, Cloud testing, Cloud cost Model, Benefit Analysis

I. Introduction

With the advent of billing for pay per use cloud services, compute hours has gained enough significance in software testing. Test-suites having hefty test cases that utilize a fraction of compute hours in every month for production testing involves a significant share of monetary loss to the hosting organization.

On the other hand, the compromise in the number test cases written to minimize the bandwidth shall involve a huge gap in terms of QoS and would let the bugs in the code go undiscovered till a customer is affected. Hence steps are to be taken in writing test cases that not only is complete in terms of functionality but also utilize less bandwidth in reality.

A. Billing on Cloud Hosting

Billing on a cloud is based on four factors: Computing, Storage, Storage Transactions, and Bandwidth. The computing factor is priced on a per hour basis.

A compute hour is basically the duration at which a role instance runs. A role instance represents either a Web Role or a Worker Role. The following is the definition on the roles –

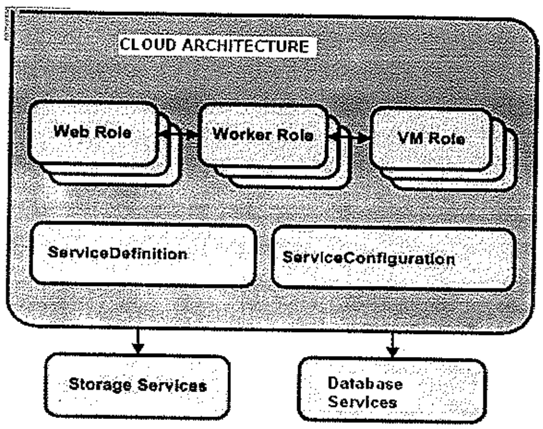

Fig 1 The Cloud Architecture & Roles

1 Reader, Department of School of Management, Bharathiar University, Coimbatore - 641 046, Tamil Nadu, India, ***vivekbsmed@gmail.com***

2 Research Scholar, Department of Computer Science, Bharathiar University, Coimbatore - 641 046, Tamil Nadu, India, ***keerthi.gopal@gmail.com***

Web role – A web role is a role that is customized for web application programming as supported by webserver and programming language platform that can work on the webserver. The benefit of using this type of role is that the webserver setup is done for the customer already. This role is best used for providing a web based frontend for the hosted service. It is not suited for long running processes.

Worker role – A worker role is a role that is useful for generalized development, and may perform background processing for a web role. When there is a need for a background process that performs long running or intermittent tasks, worker roles are most suitable.

VM role – A VM role is a special type of role that enables you to define the configuration and updates of the operating system for the virtual machine. While a web role and a worker role run in a virtual machine, the VM role is the virtual machine, which gives the full control of operations. When there are long and complicated installations in the operating system or special setup issues, VM roles are best suited.

Once deployed, each of these roles can have one or more instances running. This approach gives the customers the flexibility to quickly scale their application to use multiple instances if necessary.

As an example, imagine having a worker role that is responsible for processing electronically submitted orders. If this worker role was deployed for use in a smaller business, a single instance may be appropriate. If you were to keep this instance running for a single day, your compute cost would be calculated as:

1 (instance) x 24 (hours in a day) x .12 (cents per compute hour) = $2.88

II The Existing System

The existing system is considered as a fictional photo album sharing website that shall does the following process:

1. A registered user shall upload the photos that are to be shared to his friends on the social networking.
2. The site gets the photos and shows the preview of the photos once uploaded.
3. The user views the photos, tags or labels them.
4. Step 1-3 is repeated until the user signs out of the system.

The site employs a SQL server database over cloud (SQL Azure) for storing the registration details and a storage service (blob storage) for storing the uploaded photos by the registered users.

On an average, a photo spans 1 MB and there are 80 registered users with the average photo/users as 2000 (includes both transfer in and transfer out). This amounts to 2000* 1 = 2 GB / per user (approx) and a total of 2000* 80 = 160 GB and the closest hosting plan yields the following bill of approx 37.59 dollars monthly. (Refer Fig 1).

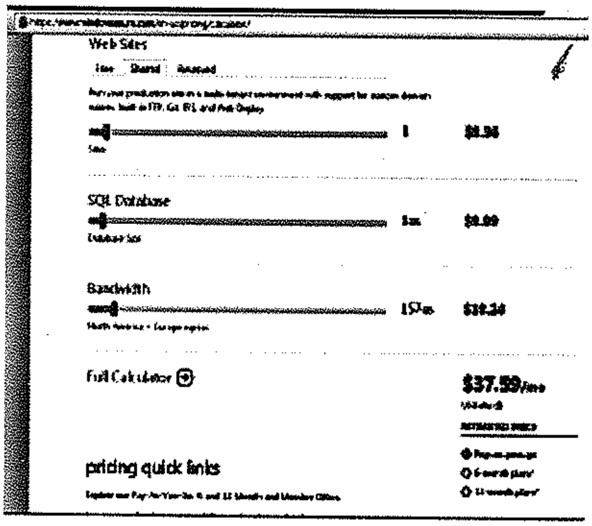

Fig 2

A typical Estimate billing on a web Role

Every month, the hosting application is upgraded with features and performing a full cycle testing at the production yields the following metric for 4000 photos mounting to 40 GB which involves a dollar cost of 9.39$.

This is approximately 25% of the total bandwidth utilization and would increase proportionately as the users or the albums uploaded climbs up.

III Problems in the Existing Approch

Though the existing system is stable, well architectured and pricing is less in terms of bandwidth utilization of pay per go, the following are the foreseen problems in terms of usage:

1. 25% of the overall compute cost spending of testing efforts may not be futile.
2. When more servers are spinned over for failover or geo-clustering management, the pricing to be spent on testing shall increase up exponentially which is obviously not the best way to jeopardize.
3. When the slab-pricing formula is introduced by the hosting solution provider, the prices shall increase in further over exponential soughting alternate ways to reduce the usage bandwidth.

IV. The Proposed Solution

The idea is to use a worker role of the cloud service which shall yield out a mock or compressed size of the image and store in the same blob of the cloud storage.

When a request to serve the image is encountered by the webserver on the cloud, the server shall determine if the request is real or for testing by adopting a method of logistic regression and serve the compressed image if the request is found to be from the test bed, thus saving on the bandwidth utilization and reduction of the compute hour charges.

A. Cloud Services

When an application is created and run in cloud Azure, the code and configuration together are called a cloud service. By creating a cloud service, a multi-tier application in reality can be deployed over the cloud, defining multiple roles to distribute processing and allow flexible scaling of the application.

A cloud service consists of one or more web roles and/or worker roles, each with its own application files and configuration. Web roles provide a dedicated web server that can be used for hosting the web front-end of the cloud service. Application code hosted within worker roles can run tasks in the background that are asynchronous, long-running, or perpetual.

Storage services provide storage in the cloud, which includes Blob services for storing text and binary data, Table services for structured storage that can be queried, and Queue services for reliable and persistent messaging between services.

B. Worker Roles To The Rewscue

A worker role runs in the background to provide services or execute time related tasks like a service process. worker roles are created to read work items posted to a queue by the web role front-end.

To process the work item, the worker role extracts information about a photo entry from the message and then retrieves the corresponding entity from table storage. It then fetches the associated image from blob storage and creates its thumbnail, which it also stores as a blob. Finally, to complete the processing, it updates the URL of the generated thumbnail blob in the guest book entry, which shall be used when a request is reached from a test-bed.

At the time of writing this paper, the following is the pricing available on a hosting of worker role.

Table 1 Cloud- Worker Role pricing Model

CLOUD SERVICES INSTANCE SIZE	CPU CORES	CPU SPEED	MEMORY	INSTANCE STORAGE	I/O PERFORMANCE	COST/HOUR
Extra Small	Shared	1.0 GHz	768 MB	20 GB	Low	$0.02
Small	1	1.6 GHz	1.75 GB	225 GB	Moderate	$0.12
Medium	2	1.6 GHz	3.5 GB	490 GB	High	$0.24
Large	4	1.6 GHz	7 GB	1,000 GB	High	$0.48
Extra Large	8	1.6 GHz	14 GB	2,040 GB	High	$0.96

The code in table 2ensure the compression of the image by sacrificing the QoS for the services rendered to the test-bed when a worker process is run.

Table 2 – Logic for Worker Process Image Compression

```
public class WorkerRole : RoleEntryPoint
{
  ...
  public override void Run()
  {
    Trace.TraceInformation("Listening for queue
messages...");

    while (true)
    {
      try
      {
        // retrieve a new message from the queue
        CloudQueueMessage msg = this.queue.GetMessage();
        if (msg != null)
        {
          // parse message retrieved from queue
          var messageParts = msg.AsString.Split(new char[]
{ ',' });
          var imageBlobUri = messageParts[0];
          var partitionKey = messageParts[1];
          var rowkey = messageParts[2];
          Trace.TraceInformation("Processing image in blob
'{0}'.", imageBlobUri);

          string thumbnailBlobUri =
System.Text.RegularExpressions.Regex.Replace(imageBlobUri,
"([^\\.]+)(\\.[^\\.]+)?$", "$1-thumb$2");

          CloudBlob inputBlob =
this.container.GetBlobReference(imageBlobUri);
          CloudBlob outputBlob =
this.container.GetBlobReference(thumbnailBlobUri);

          using (BlobStream input = inputBlob.OpenRead())
          using (BlobStream output =
outputBlob.OpenWrite())
          {
            this.ProcessImage(input, output);

            // commit the blob and set its properties
            output.Commit();
            outputBlob.Properties.ContentType =
"image/jpeg";
            outputBlob.SetProperties();

            // update the entry in table storage to point
to the thumbnail
            GuestBookDataSource ds = new
GuestBookDataSource();
            ds.UpdateImageThumbnail(partitionKey, rowkey,
thumbnailBlobUri);

            // remove message from queue
            this.queue.DeleteMessage(msg);

            Trace.TraceInformation("Generated compressed
image in blob '{0}'.", thumbnailBlobUri);
          }
        }
        else
        {
          System.Threading.Thread.Sleep(1000);
        }
      }
      catch (StorageClientException e)
      {
        Trace.TraceError("Exception when processing queue
item. Message: '{0}'", e.Message);
        System.Threading.Thread.Sleep(5000);
      }
    }
  }
  ...
}
```

5. Immedieate Benefits

Considering the above case on the worker role, the cost and the processing bandwidth for the scenario of photosharing, it would take 50% (as the image is scaled to a ratio of 1: 2) of the storage bandwidth to store the lossy compression images that would yield to 10 GB, which requires an extra small cloud service that has a processing cost of 0.02 / hour, which equates to 14.4 dollars, that is greater than the current cost of 39.7 * 0.25 = 9.925. However, when a load balancing is opted with the introduction of just a new server for fail over, the pricing drastically shall show improvement as envisioned in the table 3:

Table 3 Pricing Comparison of Testing Efforts with new and proposed System

S.No	# Servers	Proposed	Existing	% Savings
1	1 (Monolithic)	14.4	9.925	-31.07% (↓)
2	2	14.4	19.85	27.45% (↑)
3	3	14.4	29.775	51.63% (↑)
4	4	14.4	39.7	63.72% (↑)

As the worker process is stored independent of geography and having a monolithic cloud defeats the purpose of the cloud architecture, the benefits shall start showing up with just a couple of redundant cloud services.

VI. The Classification

The implementation however pose a serious risk, there has to be a classification of the request that distinguishes if the request is from a real customer for viewing of the photo gallery or from a test bed that is to take away the bandwidth.

Most probably, the provision for this classification is made in code level to determine a real or a fake request. If real request, the storage blob shall serve the actual image and if from the test-bed, a compressed & thumb nailed image shall be served that was created by the worker process.

But if no provision is made at the code level to handle the request, the web process shall enact a smarter way to identify if the request is from web or the test bed through logistic regression method.

As per definition from Wikipedia. "**Logistic regression** is a type of regression analysis used for predicting the outcome of a categorical (a variable that can take on a limited number of categories) criterion variable based on one or more predictor variables. The probabilities describing the possible outcome of a single trial are modeled, as a function of explanatory variables, using a logistic function."

Logistic regression is considered to be a machine-learning artificial intelligence statistical technique that can be used to make predictions on data where the dependent variable to be predicted takes a value of 0 or 1.

$$\pi(x) = \frac{e^{(\beta_0+\beta_1 x)}}{e^{(\beta_0+\beta_1 x)}+1} = \frac{1}{e^{-(\beta_0+\beta_1 x)}+1},$$

and

$$g(x) = \ln \frac{\pi(x)}{1-\pi(x)} = \beta_0 + \beta_1 x,$$

And,

$$\frac{\pi(x)}{1-\pi(x)} = e^{(\beta_0+\beta_1 x)}.$$

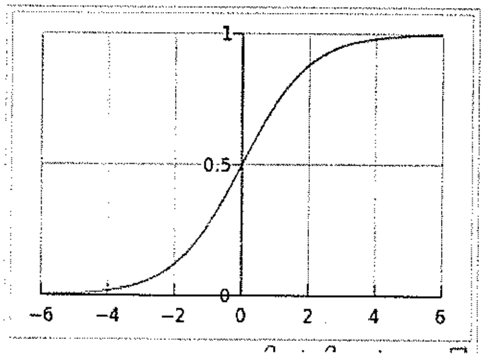

Fig 2 The logistic function, with RHS on the horizontal axis and LHS on the vertical axis – Sigmoid Function

In our case, the prediction would be that the request from machine is from test bed or not.

Logistic regression assumes that problem data fits an equation that has the form $p = 1.0 / (1.0 + e-z)$ where $z = b0 + (b1)(x1) + (b2)(x2) + \ldots + (bn)(xn)$. The x variables are the predictors and the b values are constants that must be determined.

In our case, the variables x1=Referrer URL, x2=compute efficiency and x3=user name white list of testers.

The function $1.0 / (1.0 + e-z)$ is the sigmoid function that is the domain of possible values for z is all real numbers. When using logistic regression, the beta values for the LR equation is to be arrived at. In most situations, that can be done with known results and use one of several techniques to find the values of beta that best fit the data.

After determining the values of beta, it can be used to make predictions on new data. One of the most common techniques for finding the beta values for a logistic regression equation is called iteratively reweighted least squares (IRLS).

IRLS starts with an estimate of the beta values and then iteratively computes a new, better set of betas until some stopping condition is met. Newton-Raphson method is to be used to determine a new, better set of beta values, which involves finding the calculus derivative of sigmoid function.

The program has stop criteria for the result (set as 25), ephsilon(0.01) and Jumpfactor (1000) and shall leverage the webserver log file for computing the statistics once the worker process is loaded into the machine.

When the request comes, the web server updates the logs and computes the algorithm to identify if it is from test-bed or not. If not, it shall supply the album in the original dimensions as shown here:

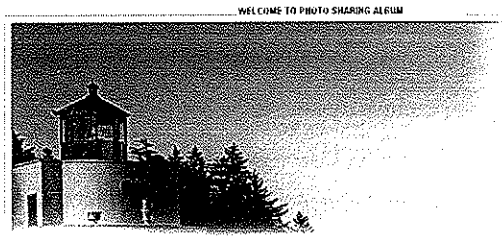

Fig 4 Album served at its original Size

On the contrary, if it is from the test bed, the size shall be a lossy compressed one, as shown below:

Fig 5 Lossy compressed and resized image served for test bed

An overall test over 10000 sample yielded a result of 9300 postive results, with 93% confidence level of classification with the historic data.

VII. Scope For Further Work

Though newton raphson is faster, but it is not accurate as it yielded a success rate of 93% which is not within the sixth sigma specifications. The system falls back to the conventional system in the case if the confidence levels are lesser than the sigma levels.

More powerful classification algorithms like particle swarm optimization could be used, to instill six sigma confidences.

VIII. Conclusion

Cloud computing is inevitable in future, having a strong line of attack of testing solutions is absolute necessity to play a role in developing a test architecture for the future to yield less compute hours and instill confidence on the testing process.

Logical Regression method is the fastest way to solve classification problems and is adopted here in a versatile way to handle requests.

The solution adopted here shall work its best on all non monolithic cloud implementations. Logical regression shall be adopted if no provision is made in the request handling to determine a real and fake request.

REFERENCES :

[[1] L. Northrop, P. Feiler, R. P. Gabriel, J. Goodenough, R. Linger, T. Longstaff, R. Kazman, M. Klein, D. Schmidt, K. Sullivan, and K. Wallnau, Ultra-Large-Scale Systems: The Software Challenge of the Future. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2006.

[2] Yuriy Brun , Giovanna Marzo Serugendo , Cristina Gacek , Holger Giese , Holger Kienle , Marin Litoiu , Hausi Müller , Mauro Pezzè , Mary Shaw, Engineering Self-Adaptive Systems through Feedback Loops, Software Engineering for Self-Adaptive Systems, Springer-Verlag, Berlin, Heidelberg, 2009 [doi>10.1007/978-3-642-02161-9_3]

[3] Jeffrey O. Kephart , David M. Chess, The Vision of Autonomic Computing, Computer, v.36 n.1, p.41-50, January 2003 [doi>10.1109/MC.2003.1160055]

[4] James E. Hanson , Ian Whalley , David M. Chess , Jeffrey O. Kephart, An Architectural Approach to Autonomic Computing, Proceedings of the First International Conference on Autonomic Computing, p.2-9, May 17-18, 2004

[5] C. Murray Woodside , Marin Litoiu, Performance Model Estimation and Tracking Using Optimal Filters, IEEE Transactions on Software Engineering, v.34 n.3, p.391-406, May 2008 [doi>10.1109/TSE.2008.30]

[6] Yixin Diao , Xiaolei Hu , Asser Tantawi , Haishan Wu, An adaptive feedback controller for SIP server memory overload protection, Proceedings of the 6th international conference on Autonomic computing, June 15-19, 2009, Barcelona, Spain [doi>10.1145/1555228.1555234]

[7] Jim (Zhanwen) Li , John Chinneck , Murray Woodside , Marin Litoiu, Fast scalable optimization to configure service systems having cost and quality of service constraints, Proceedings of the 6th international conference on Autonomic computing, June 15-19, 2009, Barcelona, Spain [doi>10.1145/1555228.1555268]

[8] Jim Li , John Chinneck , Murray Woodside , Marin Litoiu , Gabriel Iszlai, Performance model driven QoS guarantees and optimization in clouds, Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, p.15-22, May 23-23, 2009 [doi>10.1109/CLOUD.2009.5071528]

[9] Jim Zhanwen Li , John Chinneck , Murray Woodside , Marin Litoiu, Deployment of Services in a Cloud Subject to Memory and License Constraints, Proceedings of the 2009 IEEE International Conference on Cloud Computing, p.33-40, September 21-25, 2009 [doi>10.1109/CLOUD.2009.62]

[10] Ye Hu , Johnny Wong , Gabriel Iszlai , Marin Litoiu, Resource provisioning for cloud computing, Proceedings of the 2009 Conference of the Center for Advanced Studies on Collaborative Research, November 02-05, 2009, Ontario, Canada [doi>10.1145/1723028.1723041]

[11] https://www.windowsazure.com/

[12] http://en.wikipedia.org/wiki/Logistic_regression.

Author's Biography

Keerthilakshmi is working as a senior IT consultant at Cognizant technology solutions, Bangalore. She has nearly a decade of vast IT experience in multiple facets of Information technology and its management including healthcare, infrastructure, agriculture etc.., She has developed and architected testing solutions for various global clients including Abott, Johnson & Johnson, Monsanto etc.., Her areas of interests include Test-case optimization, Designing efficient storage systems for Software testing etc..,

Dr.K.Vivekanandan received the Ph.D. degree in Computer Science from Bharathiar University, India. He is currently Professor in Bharathiar University, India. He has a total teaching experience of 23 years. He has published 26 papers in international and national journals. He has produced 10 Ph.D candidates and 11 M.Phil scholars in computer science. His research interests include Data Mining and knowledge discovery, and Management Information Systems.