PROJECT REPORT

ENGINEERING DISTRIBUTED OBJECTS FOR CLOUD COMPUTING

CS-441

SUBMITTED BY:
SANKET GAURAV(UIN:667108958)
ARYADIP SARKAR(UIN:660701709)
PRADEEP GUDANAMATH JAGDISH(UIN:672433634

Aim of the Project:

To discover provisioning strategies that most effectively alleviate decreasing throughputs of applications deployed on cloud

AUT: Logicaldoc: Document Management System (http://www.logicaldoc.com/en.html)

Lines of code: 131888
Method: 8692
Packages: 103

Software Used:

Apache Tomcat
JDK
ANT
Maven
Logicaldoc
Azure web toolkit

Application Description:

LogicalDOC[1] is an enterprise electronic document management system which is extremely useful for both small and large companies. It is a valuable tool in knowledge management processing that provides a more flexible and lower-cost alternative to other proprietary applications.

A company's use of knowledge management systems facilitates the intelligent and efficient management of its resources.This leads to significant productivity increases. These systems allow universal access to the information and knowledge that is generated within the organization.

It is easy to use LogicalDOC to manage your company's documents. One of our major goals is the usability case study. LogicalDOC is a document management application with a web user interface that permits the following operations: sharing, setting security roles, auditing, and finding enterprise documents and registers. LogicalDOC allows users to easily collaborate and communicate.

Approach Used:

1. First of all source code of logicaldoc was downloaded and according to the steps described in installation document, the logicaldoc web application was deployed on our local machine.
2. Then we ran setup of logicaldoc and choose embedded database option which creates the database embedded inside the application.

3. We explored the logicaldoc web application i,e. familiarising and understanding functioning of the application.
4. 8 functions were chosen which can be frequently used over the logicaldoc web application.
5. With Jmeter we tested all those functions of the application locally over the throughput and response time metrics. We tested all 8 functions separately because there would bet times with the logicaldoc application that only majority of the user would perform same function for example in the evening when everyone is signing off from the office most of the people will be uploading the documents. So that particular function is used most in evening and same is case in morning when everyone comes to the office they would download the documents to work on.
6. The application was deployed on MS-Azure cloud through all 3 methods listed in installation script:
    - Eclipse
    - Webapp
    - Virtual Machine
7. We gone forward with deploying over virtual machine application as cloud service over the azure as we faced few problems while deploying through eclipse and webapp. So, we deployed our logicaldoc web application as IaaS (Infrastructure as a Service) as company should have control over the application.
8. Then we tested all those 8 functions individually to get the bottleneck on each function of the application. All the test cases are mentioned in the test case document results were reported below.
9. After finding the bottleneck on certain number of users for which CPU usage goes very high like 95%, we decide the provisioning strategy.
10. Then we wrote provisioning strategy in azure to balance the load at stress situations.
11. We varied our provisioning strategy and seen the performance i,e. throughput of the system and error rate has decreased or not.
12. We wrote provisioning strategy for different times of the day as our application will be mostly used in company scenarios.

Jmeter Testing Plan:

We created test cases in Jmeter according to the test case document.
Description of each test cases

1) **Adding a document** : Uploading a document on LogicalDoc from user's local machine

2) **Editing a document** : Editing a document on LogicalDoc from user's local machine using 'Checkout' and 'Checkin'

3) **Download a document** :Uploading a document on LogicalDoc from user's local machine

4) **Delete a document** : Deleting a document on LogicalDoc from user's local machine

5) **Locking a document** : Locking a document on LogicalDoc from user's local machine so to protect the document so that no other user can make changes to it

6) **Unlock a document** : Unlocking a document on LogicalDoc from user's local machine so to protect the document so that no other user can make changes to it

7) **Copying a document** :Copying a document from one folder to another in LogicalDoc application in local machine

8) **Moving a document** : Moving a document from one folder to another in LogicalDoc application in local machine

9) **Search** : To search a word in all the documents present in the repository

All the test steps and its description are present in cloud_Testcases.

Bottleneck :

Following figures shows the screenshot of azure and jmeter of our tested scripts.

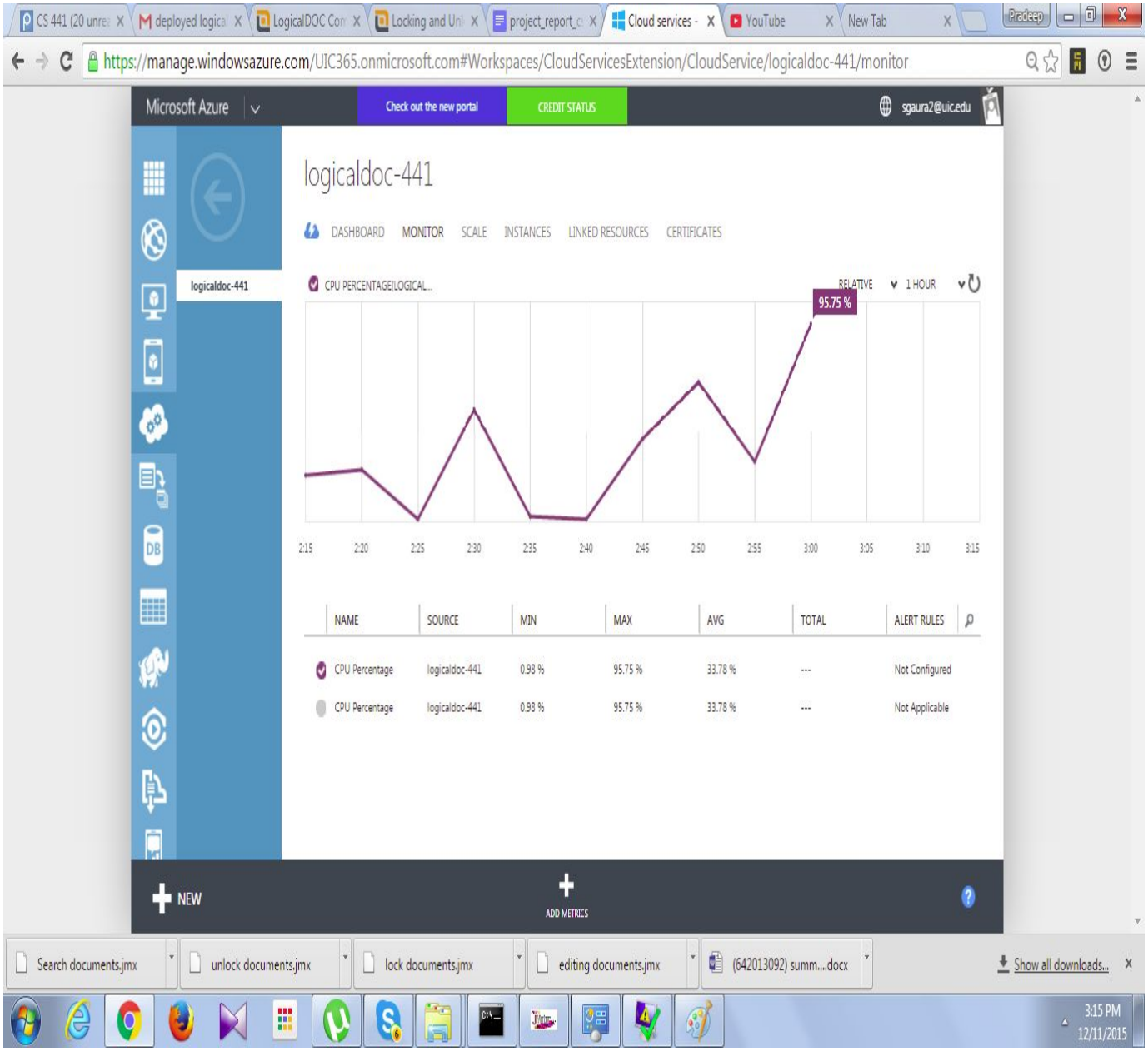

Fig 1.Deleting a document

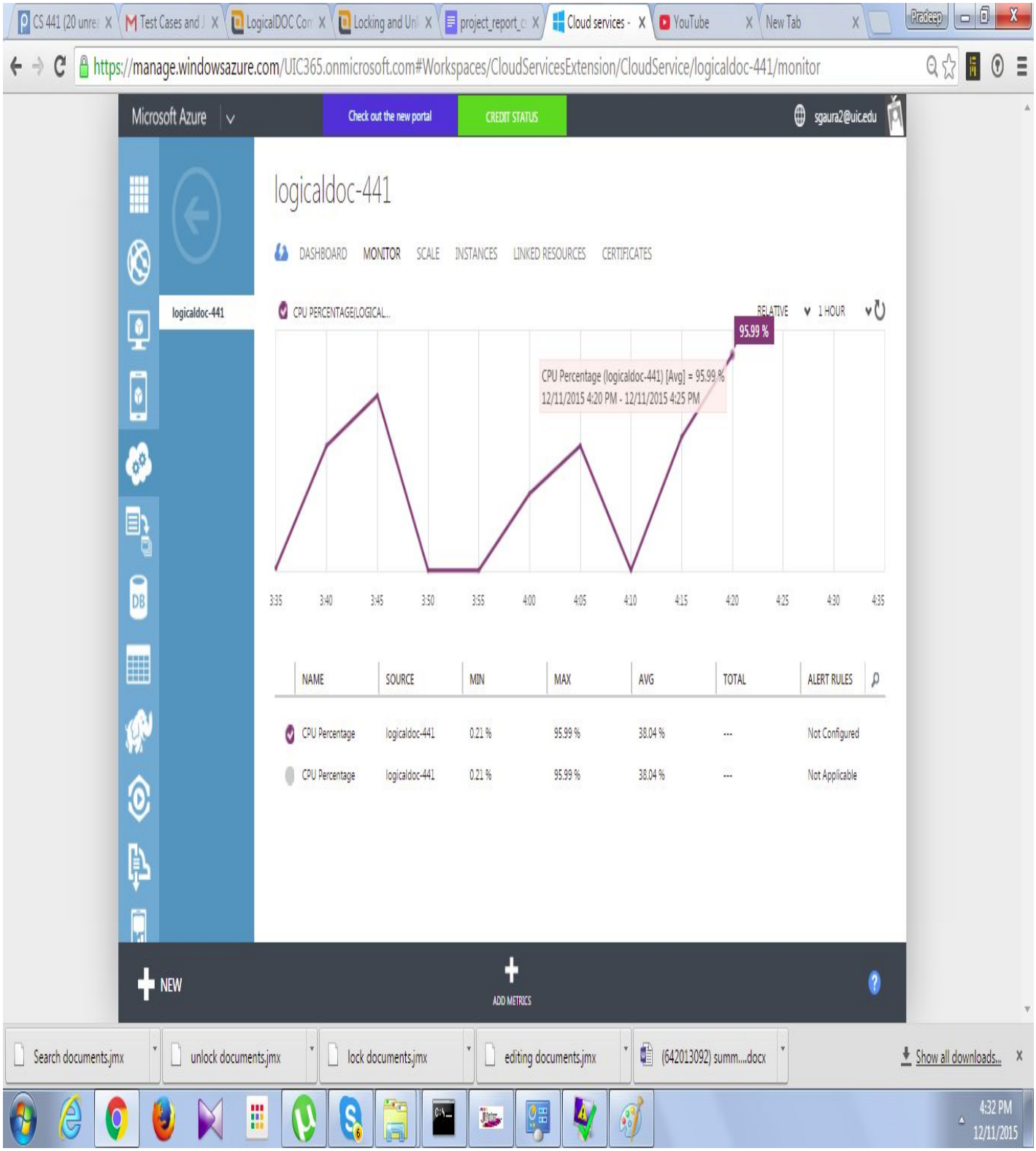

Fig 1.Unlocking

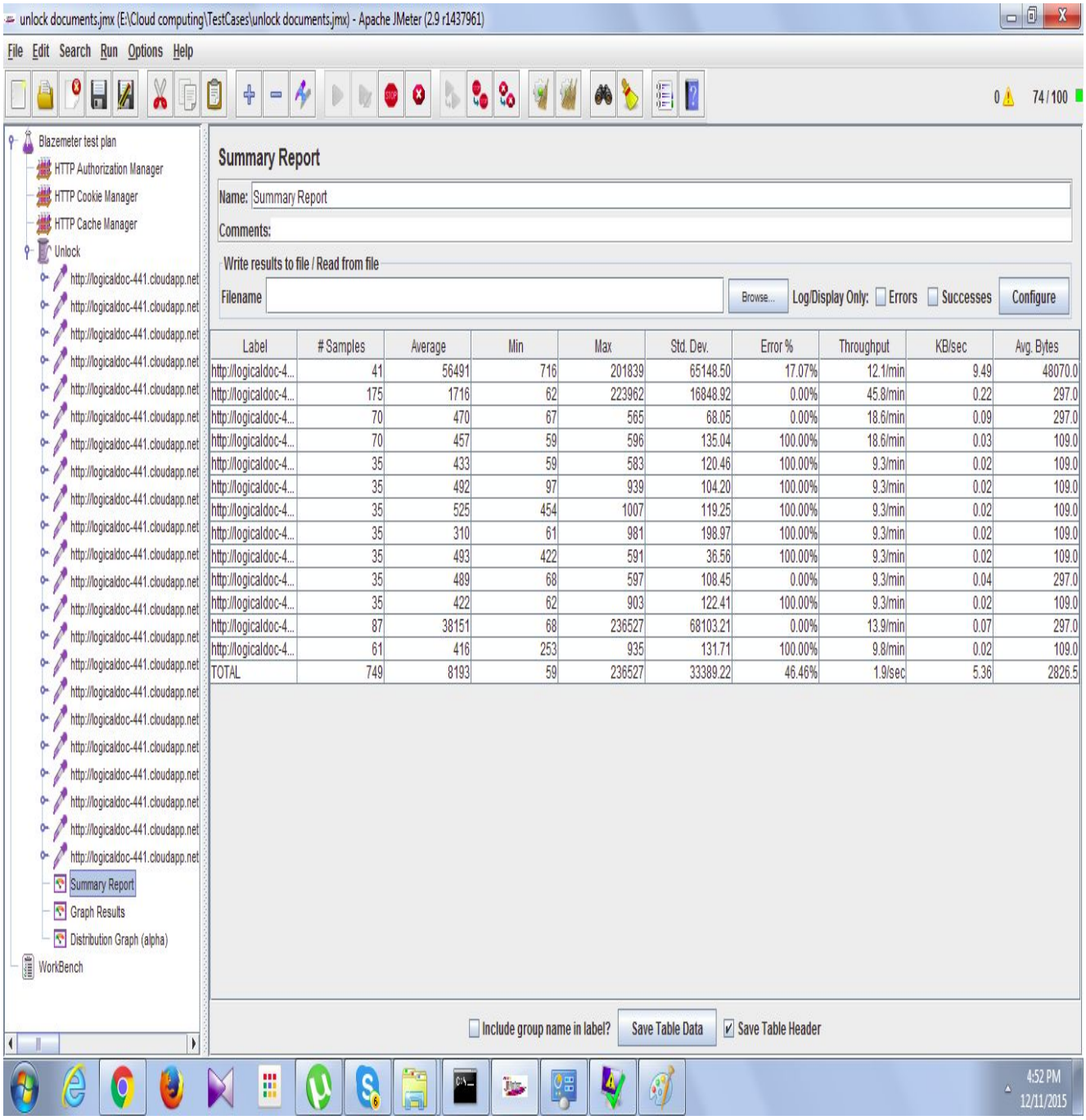

Fig3:Upload jmeter report

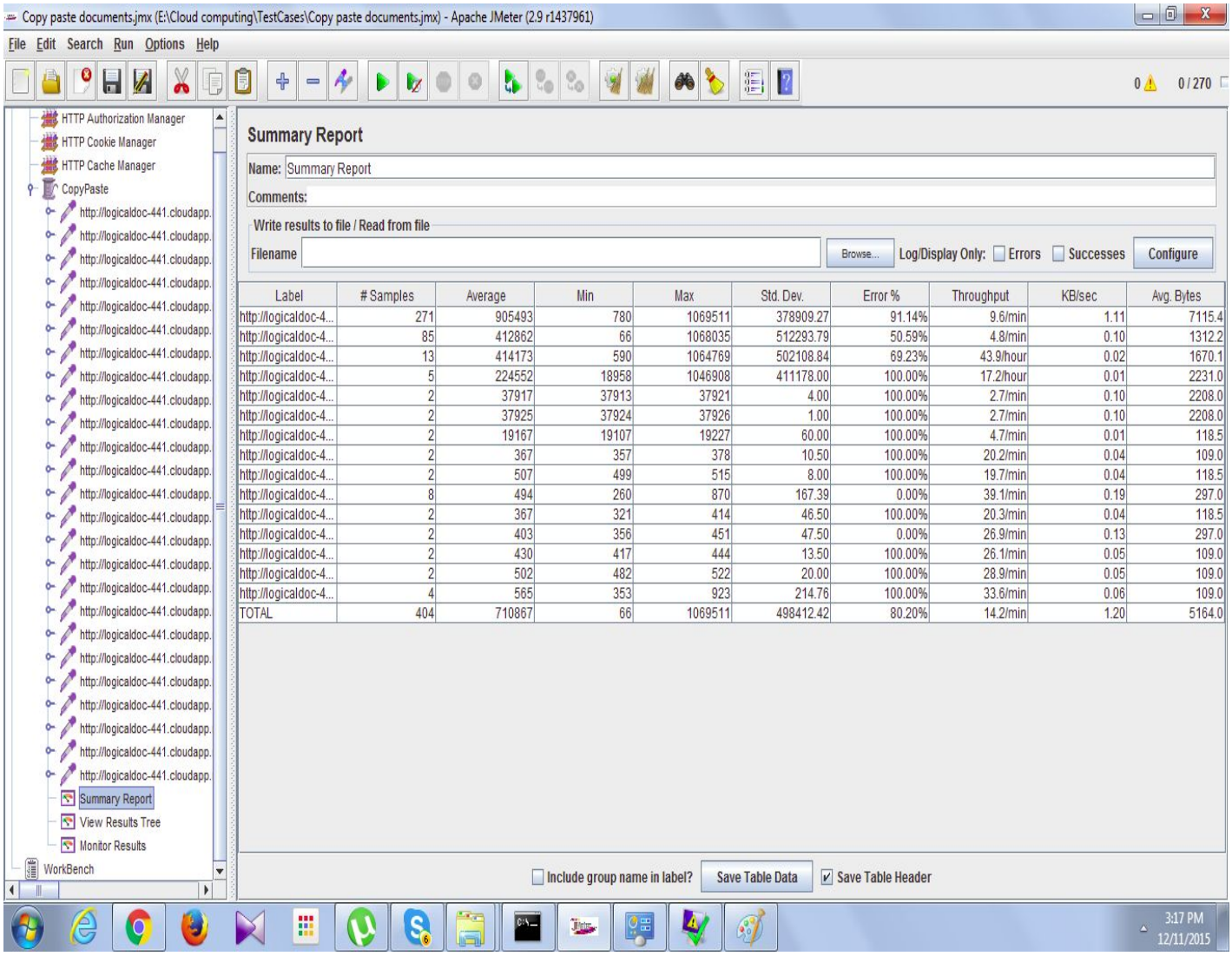

Fig4. Copy paste

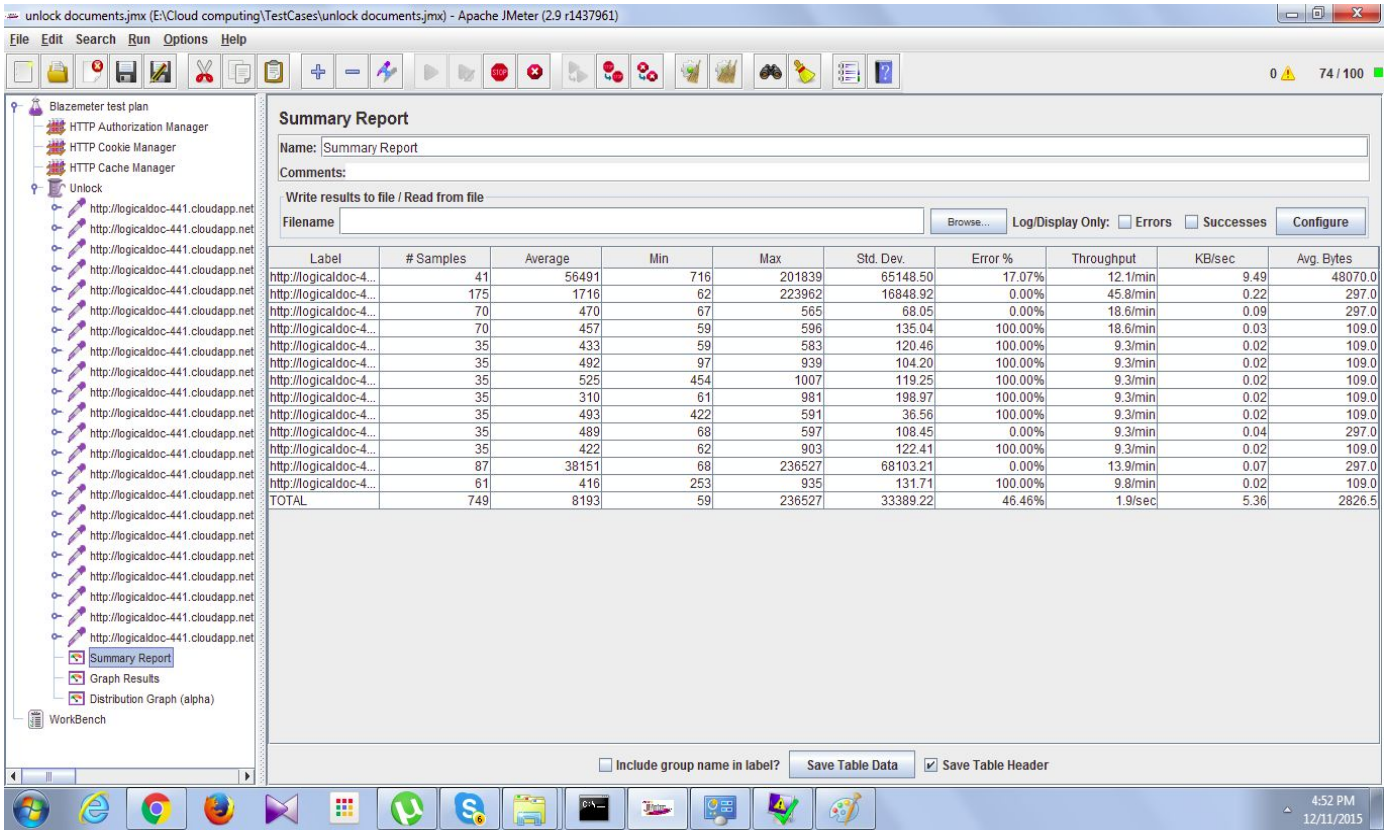

Table for finding bottleneck

Test Case title	Number of Users	CPU Usage	Throughput	error %
Uploading	65	98.71%	1.9/sec	46.46
Downloading	400	95.1%	6.2/min	70.20
Deleting a document	170	95.75%	6.1/s	52.5
Locking	250	95.5%	23.7/s	55.1
Unlocking	100	95.99%	4.6/s	49.71
Move/Cut a document	250	87.05%	5.5/s	52.3
Copy/Paste document	270	91.2%	3.1/s	51.22
Search	200	92.43%	5.9/s	41.4

We are defining bottleneck in terms throughput and CPU usage i,e. CPU usage goes high till 80% - 95%. So, we stressed the application till the above limit we achieved.
From above table, bottleneck for all 8 functions on given users can be found. After running stress testing for all the functions we found that uploading function bottleneck is achieved after only 65 users hit and is considered as most expensive operation of the application.
Theoretically, we also know that uploading is the most expensive application among all 8 functions that we choose to test for our logicaldoc web application.
The other bottleneck can also be found from the table. So to avoid these extreme cases we apply provisioning strategies to avoid the stress situations and and let the application run smoothly.

Provisioning rules

Configurations of first virtual machine:

- Configuration Name: VM1
- Number of Cores: 1
- Memory: 1.75 GigaBytes

We choose minimum configurations in order to test our application run over basic configurations and it can also be said to have worked for small company for less number of employees and we want to move forward step by step. If we can successfully provision for small configuration then we can test for more users by increasing configurations.

To remove our first bottleneck i,e. upload we started with provisioning with same base configuration say VM 1. Then we tried to provision with the VM2 which is of different configuration.
Configuration Name: VM2
Number of Cores:2
Memory: 3.5 GigaBytes
But we have observed that throughput did not improved much after provisioning from VM2 than VM1. Since to provision VM2 configuration costs more than VM1 so, we can use VM1 for provisioning in normal hours. In more stress hours of the day we can use VM2 but that can be only for certain period of the day and add VM1 by default if CPU goes high.
To achieve provisioning rule we tried with different rules for CPU and got different throughput error % for different rules of CPU:

S.No	CPU usage range rules	Throughput	Error %
1	65%-80%	20.9/sec	7.15%
2	50%-70%	22.5/sec	5.29%

3	70%-90%	18.2/sec	10.11%

From above table we can see that 3 rules gives low throughput and high error rate and 1 rule gives best result but it will cost more to have that rule as it start at 50% CPU usage only. So,after comparing with all the CPU usage range, we found that 65%-80% will be optimal provisioning strategy to use.

Our application logicaldoc is a document management system which can be used within a company. Employees of the company can checkout i,e. download the document and work over it and in the evening while going back to home upload the updated document.
Since our application will be used in a company we can anticipate the maximum load times or high stress time. Following is the stress distribution for a company and provisioning strategy that can be applied at different period of time of the day:

Time limit	Reason	Load Situation	Provisioning
7.00am-9.00am	Starting of work	Over medium load (Upload, Download)	Add VM 1 when 65%=<load>=80%
9.00am-11.30am	Normal Office hours	Medium load	Add VM1 when 65%=<load>=80%
11.30am-12.30pm	Lunch hours	Low load	Scale out all VMs, only main VM1 running. If load>=85% then add VM1
12.30pm-4.00pm	Normal Office hours	Medium load	Add VM1 when 65%=<load>=80%
4.00pm-4.30pm	Tea break	Low load	Scale out all VMs, only main VM1 running. If load>=85% then add VM1
4.30-7.00pm	End office hrs	High Load (Upload)	Add VM 2 when l65%=<load>=80%
7.00-7.00am	No office	Low load	Scale out all VMs, only main VM1 running. If load>=85% then add VM1

After provisioning

Table after provisioning

Test Case title	Previous Throughput	Throughput after provisioning	% improved
Uploading	1.9/sec	2.6/sec	31.57%
Downloading	6.2/min	7.98/min	28.70%
Deleting a document	6.1/sec	8.2/sec	34.4%

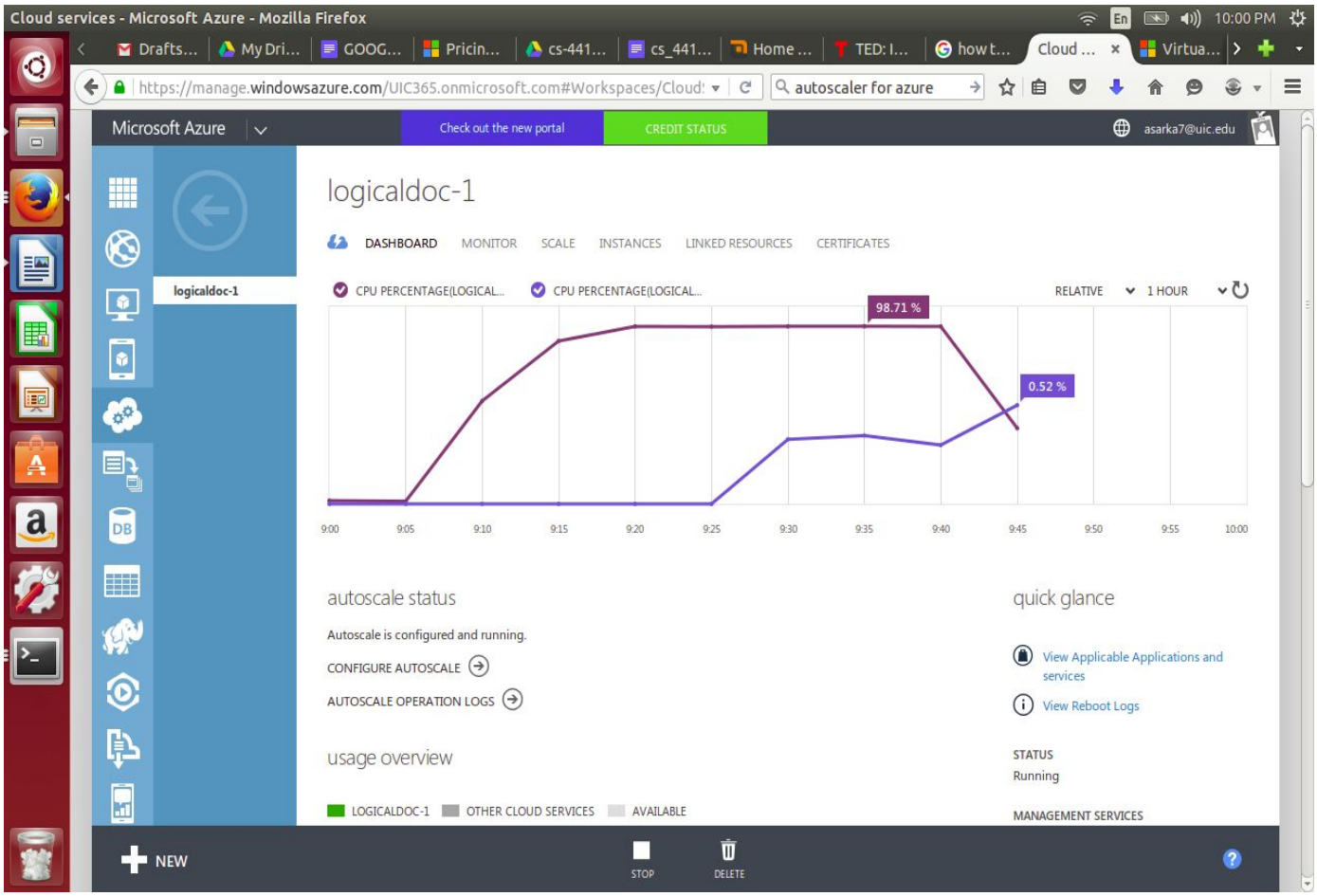

Fig: After provisioning graph for upload

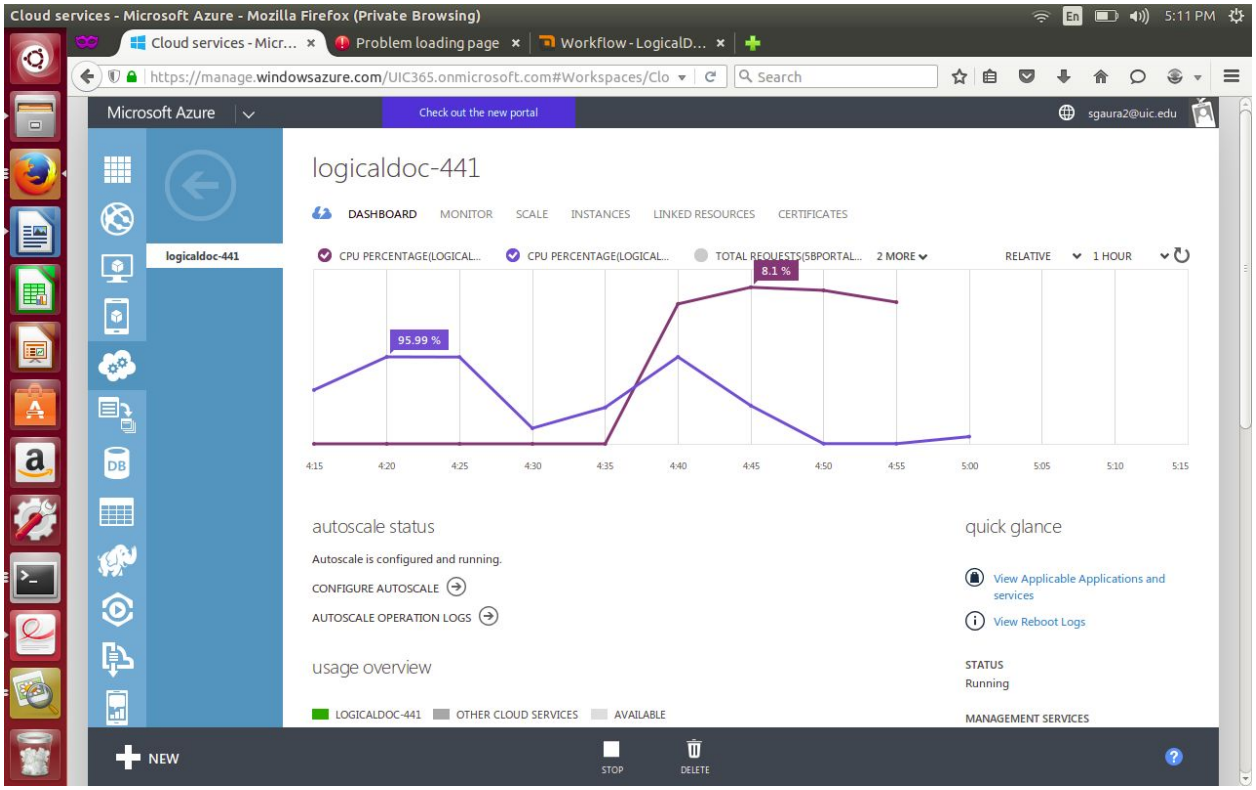

Fig: After provisioning for unlocking

Conclusion:

1. We successfully deployed our logicaldoc web application over cloud.
2. Bottleneck was found.
3. Based on the bottleneck different provisioning strategy were employed.
4. Optimal provisioning strategy was found and rules were employed on azure for good performance of the application.