

AZURE Billing Issues ticket 2208240030000696

Summary of finding

5 Test were created to investigate the AZURE billing. The results of the test show that in the least it is 9 times or 900% higher than what it should be. The description of each test and result are given below.

I have also provided the subscriptions in which the test results can be found. The source code for carrying out the test together with setup steps has also been provided.

I lodged this support ticket on 24/8/2022, it is now 17/10/2022 and you have continued to charge me, what I believe is at least 900% more than what it should be. The test that I have provided should not take more than a day to run. The results should be available in 24 hours. Based on the test, I would expect Microsoft to come to the same conclusion.

I also note that this discrepancy in billing would be effecting many of your customers. As such it is in your best interest to expedite this investigation.

Test1

In this test a single file was saved and it was updated 100,000, 200000 and 300000 times. It was carried out by directly connecting to the storage account via the api provided. The result of the test match closely with the published charge and as such form the basis for finding the discrepancies of the other test.

Subscription	Number of updates	Charge
ccbc400e-8092-480d-9761-xxxx	100000	\$0.85
9f6be25a-eec4-48fb-99bb-xxx	200000	\$1.70
0628a946-e989-4908-a5f5-xxx	300000	\$2.55

Test2

This test is similar to test1, however instead of updating a single file, a new file is created every time and the files are distributed evenly across 10 blob containers. Again the charges match with the results of test1

Subscription	Number of updates	Charge
1a11290e-f241-4f6e-94a0-xxx	100000	\$0.85
6578e51e-b3e4-4bae-85f2-xxx	200000	\$1.70
bc299c75-a6ae-40e0-b7e7-xxx	300000	\$2.55

Test3

This test is the same as test1 except that it updates the blob via an Azure serverless function. The first indication of Discrepancies become evident.

Subscription	Number of updates	Function Charge	Storage charge	Charge
977b8bf6-7fbb-4259-xxx	100000	\$0.05	\$1.85	\$1.90

Comparing the above result with that of test1 it can be seen that the charge has increased by a \$1. The above charge does not include the charge for the serverless function compute time (The compute time cost is an addition \$0.05). It is to be note that the code used is exactly as recommended in your documentation for updating blobs via serverless function.

Possible cause for the higher pricing is I would think in the case of the serverless function it needs to check if the blob exist if not create it, this extra check is causing a charge overhead. However, in your documentation such a doubling of charge should be highlighted if that is by design and is expected to be the case. I also note that if this is how its meant to be serverless function are in no way cheaper than running a stand alone server as then the method used in Test1 can be used.

Test4

In this test an azure function that works very much like what I am using in the billed production application has been used. The azure function calls are done in series. The charges reflect the pricing of test 3 in that they are much greater than expected.

Subscription	Number of updates	Function Charge	Storage charge	Charge
4a90f139-4514-4a04-xxx	100000	\$0.04	\$1.88	\$1.92
0df980f9-e377-4e92-xxx	200000	\$0.08	\$3.74	\$3.82
36f6f440-442b-4c48-xxx	300000	\$0.13	\$5.60	\$5.72

The possible cause as suggested for test3 , in the case of the serverless function it needs to check if the blob exist if not create it, this extra check is causing a charge overhead.

Test5

This is very much like the way my application works. The difference between test4 and test5 is that in test5 the serverless functions are called in parallel. Calling serverless functions in parallel or in series to do an identical data transfer should cost the same. The only difference expected would be the timing with the parallel case much faster than the series case.

Subscription	Number of updates	Function Charge	Storage charge	Charge
e608ffff-6404-43c2-xx	100000	\$1.92	\$2.23	\$4.15
d16b6112-6ede-4eff-xx	200000	\$8.48	\$4.37	\$12.85
c7d8765b-15dc-4b3e-xxx	300000	\$16.62	\$6.51	\$23.13

It is to be noted that the charge increases exponentially to a linear increase in volume. Further the Function charges are the main contributor. The function charges should have been identical to test4 as the only difference is that in the case of test4 they are called in series while test5 the functions are called in parallel.

The cause for the increase is that the function that fail due to AZURE inability to create them or because they cannot connect with the storage are also charged. That is functions that fail before even reaching even a line of my code are getting billed. All of the code that runs and fails is part of the AZURE infrastructure and a not a line of it is my code. As such I should not be getting charged for even a single one of these function calls. I have provided all the errors that I have received from AZURE for your information.

Conclusion

The Azure bills include charges for failed function and storage calls that are caused purely by the code that is part of the AZURE infrastructure. These invalid charges increase exponentially to a linear increase in volume. Test5 which should have had a costing similar to test1 or test2 shows that its charges instead at a volume of 300000 is $23.13/2.55 = 9.06$ times what is expected. Given that in my live applications the volume is in the millions the exponential increase can be expected to be much higher than 9 demonstrated in these test.

Appendix 1

Test Software setup.

Prerequisite: It is assumed that the person reading and trying out this software is a .net developer who is also knows to create azure serverless function apps








Overview

The software to carry out the above test consist of two applications

1. Azurebill : A serverless function application consisting of two serverless functions.
 - saveItem
 - saveItemTest1
2. azurebillConsole: A console application that carries out each of the test.

Install procedure:

1. Create a subscription
2. Create a storage account
3. Open the solution in visual studio. Publish the Azurebill to the subscription created
4. Add a connection string with the name storageAccountKey pointing to the storage account created in step 2 as show in fig 1.

Name	Value
AzureWebJobsStorage	 Hidden value. Click to show value
FUNCTIONS_EXTENSION_VERSION	 Hidden value. Click to show value
FUNCTIONS_WORKER_RUNTIME	 Hidden value. Click to show value
storageAccountKey	 Hidden value. Click to show value
WEBSITE_CONTENTAZUREFILECONNECTIONSTRING	 Hidden value. Click to show value
WEBSITE_CONTENTSHARE	 Hidden value. Click to show value
WEBSITE_RUN_FROM_PACKAGE	 Hidden value. Click to show value

azurebillConsole

All settings for this console app is in the settings.json file. The available setting and there meaning is given in the following table

Setting name	Purpose	Value for test5 with 100000
siteCount	The number of blob containers to create	10
folderCount	The number of folders in each container	10
fileCount	The number of files in each container	10
NumberOfUpdates	The number of updates to files that have been created	10
maxtrycounts	Maximum number of times to try and save a file when the post to azure fails	100
trypause	The number of milliseconds to pause when a post fails before trying to post the same file again to azure	1000
fulllog	The path to save the full log	C:\\azurebill\\logs
errorlog	The path to save the full log	C:\\azurebill\\logs

saveItem	The saveItem serverless function url in azurebill	
saveItemTest1	The saveItemTest1 serverless function url	
sitedirectlink	The azure storage account connection string created in step 2	
runmode	The name of the Test to run	test5

The runmode value in the settings file as noted above will determine the test that will be run. The runmode will be test1, test2, test3, test4 or test5 respectively depending on the test you wish to try.

The total volume of post or file saves = $\text{siteCount} * \text{folderCount} * \text{fileCount} * \text{NumberOfUpdates} * 10$

Please note the value 10 in the equation above for the volume saved. This as every post in my application will result in 10 files getting created.

To run the console app complete the setting above as required and then double click on the exe. It will then open a console and display the setting value. Enter any key to start the run.

It is recommended that a subscription is created for each test/volume combination. This way the resulting cost will be just for the combination and as such easy to see.