VMApps Publishing Replication is slow_ACG

Last updated by | Anton Iulian Mocanu | Sep 19, 2022 at 11:18 PM PDT

Tags

cw.Incubation

Contents

Summary

This Wiki is helpful for scenarios where customers complain that their Replication of Application Version is slow to regions

Symptom

Customers trying to replicate application version to different regions may experience significant delay in the replication to complete, they complain that the replication is taking hours. Here are some of the symptoms customers may see:

1. The customer image replication is taking longer than 6 hrs
2. The customer is noticing a 2x increase in replication times than they used to
3. If customer has more than 10 replica, the copy time will be at least 3 times more than single replica count
4. The customer application replication is failing with timeout consistently

Mitigation

Replication of Application Versions may some times fail due to various reasons and some of the symptoms are listed above, Shared Image Gallery Product team and Azure Extensions team has agreed to take ICM's for the above symptoms as it's difficult for CSS to investigate the cause of the replication time out and delays.

ICM

Please use the following queries to capture the logs and submit an ICM to : AzureRT -> CRP-PIR.

Use ICM Template

If you have the correlation ID of the replication job, get the correlationId and run the below query to get the details of the job to understand the start and end time of the job. If you don't have the correlationid, then query using the subscriptionId and narrow down using resourceuri to find the correlationId

```
cluster('armprod.kusto.windows.net').database('ARMProd').EventServiceEntries
| where correlationId == '3edd2d76-6ba3-488a-8ef9-43413edfda80'
| where subscriptionId == 'subscriptionid'
| project PreciseTimeStamp, operationName, eventName, status, subStatus, resourceProvider, properties
```

The properties field from the above query will give you the ServicerequestID going to CRP, Use the ServicerequestID(which is the operationID in CRP -> CapsApiQosEvent) table and run the following query to get details of the operation, look for the requestEntity Field to find the replication details, you can also review the E2EDurationinMilliseconds field to see how much time it took for the operation.

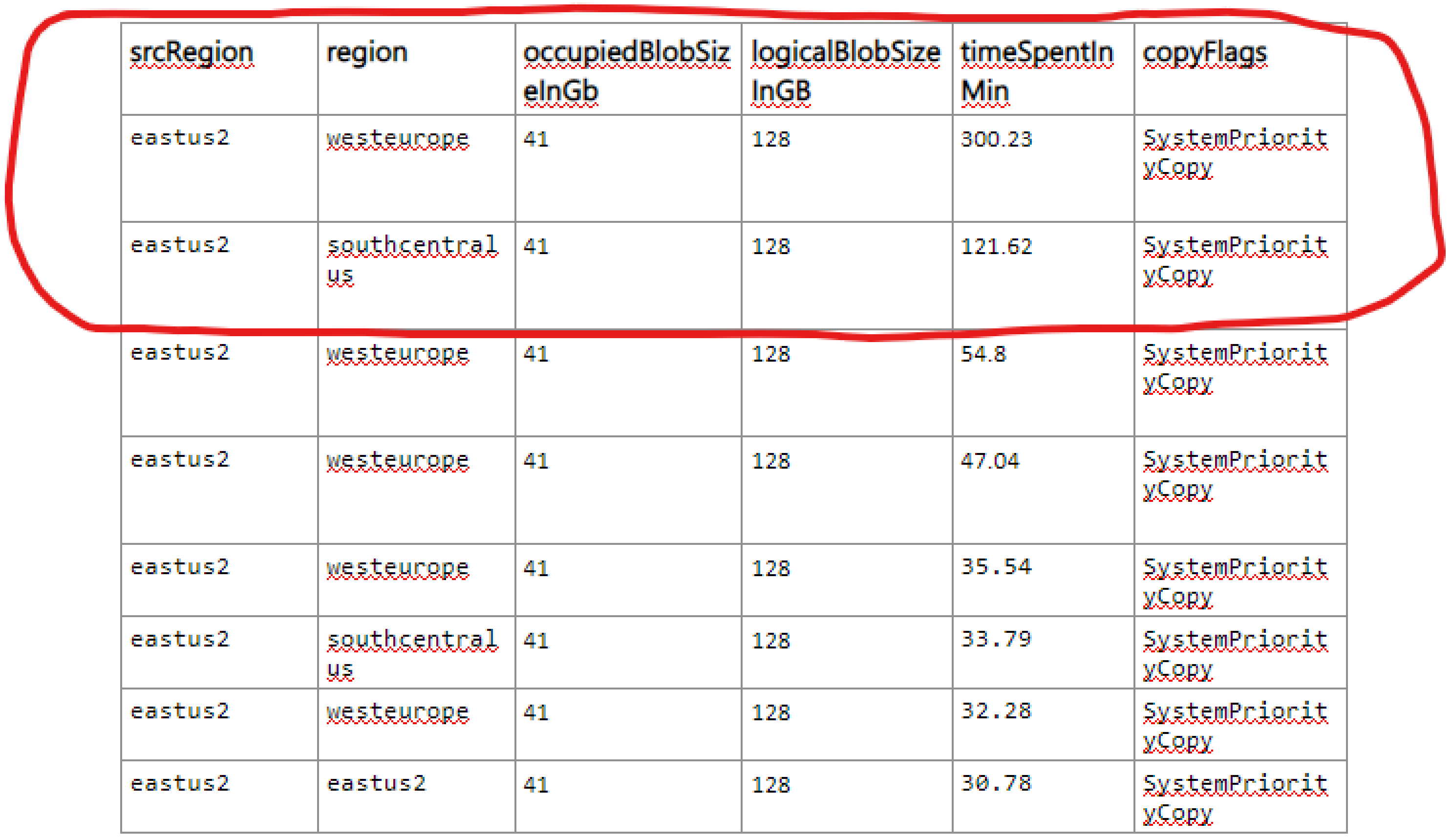
```
cluster('azcrp.kusto.windows.net').database('crp_allprod').CapsApiQosEvent
| where subscriptionId == <SUBSCRIPTION ID>
| where operationId == 'bff8f310-1812-4e19-b4b4-faaebcc3f60d'
| where PreciseTimeStamp >= datetime(2020-10-27 00:00:00) and PreciseTimeStamp <= datetime(2020-10-27 23:59:00
| project operationId, operationName, resourceGroupName, resourceName, durationInMilliseconds, e2EDurationInMi
```

Observe the "e2EDurationInMilliseconds" to understand the end to end operation time for replication, If you don't have the operationID, you can also run the following command and filter by subscriptionid, time and ExternalGalleryApi.PutGalleryApplicationVersion.PUT operations (see example below) to get all the PUT operations on the resource

```
cluster('azcrp.kusto.windows.net').database('crp_allprod').CapsApiQosEvent
| where subscriptionId == <SUBSCRIPTION ID>
| where operationName contains "ExternalGalleryApi.PutGalleryApplicationVersion.PUT" and resourceName contains
| where PreciseTimeStamp >= datetime(2020-10-27 00:00:00) and PreciseTimeStamp <= datetime(2020-10-27 23:59:00
| project operationId,operationName, resourceGroupName, resourceName, durationInMilliseconds, e2EDurationInMil
```

You can also run the following command to get the replication time for each region:

```
cluster('azcrp.kusto.windows.net').database('crp_allprod').PirCasApiQosEvent | join cluster('azcrp.kusto.windo
| where TIMESTAMP > ago(2d) and copyStatus == "Success"
| where  subscriptionId == <SUBSCRIPTION ID>
| extend replicateMetadataObj=parse_json(replicationMetadata)
| extend occupiedBlobSizeInGb = round(todouble(occupiedBlobSizeInBytes) / 1024 / 1024 /1024), logicalBlobSizeI
| project srcRegion, region,  occupiedBlobSizeInGb, logicalBlobSizeInGB,  timeSpentInMin, copyFlags
| where logicalBlobSizeInGB == 128 and occupiedBlobSizeInGb ==41
| sort by timeSpentInMin
```



srcRegion	region	occupiedBlobSizeInGb	logicalBlobSizeInGB	timeSpentInMin	copyFlags
eastus2	westeurope	41	128	300.23	SystemPriorityCopy
eastus2	southcentralus	41	128	121.62	SystemPriorityCopy
eastus2	westeurope	41	128	54.8	SystemPriorityCopy
eastus2	westeurope	41	128	47.04	SystemPriorityCopy
eastus2	westeurope	41	128	35.54	SystemPriorityCopy
eastus2	southcentralus	41	128	33.79	SystemPriorityCopy
eastus2	westeurope	41	128	32.28	SystemPriorityCopy
eastus2	eastus2	41	128	30.78	SystemPriorityCopy

**In the above example, you see that image was replicated more than once in Westeurope and SouthCentralUS, while some of the replication jobs were completed in less than an hour others took over 2 hours