[CSSTSG-Orcas-PGFS-Perf] How to determine workload for a PostgreSQL Flexible Server

Last updated by | Shawn Xiao | Feb 8, 2023 at 8:43 PM PST

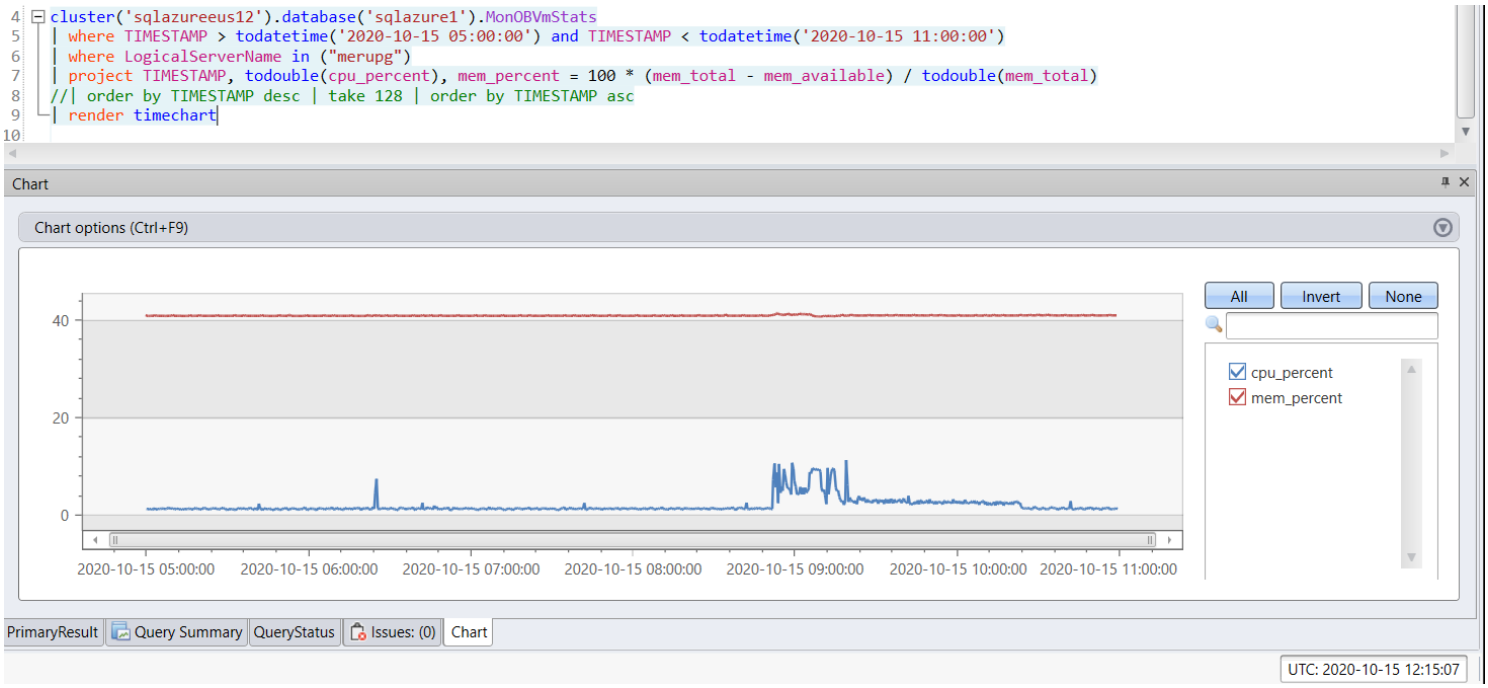
Contents

1 Check PG Server CPU and Memory Usage

Kusto query.

```
cluster('sqlazureeus12').database('sqlazure1').MonOBVmStats
| where TIMESTAMP > todatetime('2020-10-15 05:00:00') and TIMESTAMP < todatetime('2020-10-15 11:00:00')
| where LogicalServerName in ("merupg")
| project TIMESTAMP, todouble(cpu_percent), mem_percent = 100 * (mem_total - mem_available) / todouble(mem_tot
//| order by TIMESTAMP desc | take 128 | order by TIMESTAMP asc
| render timechart
```

Sample output.

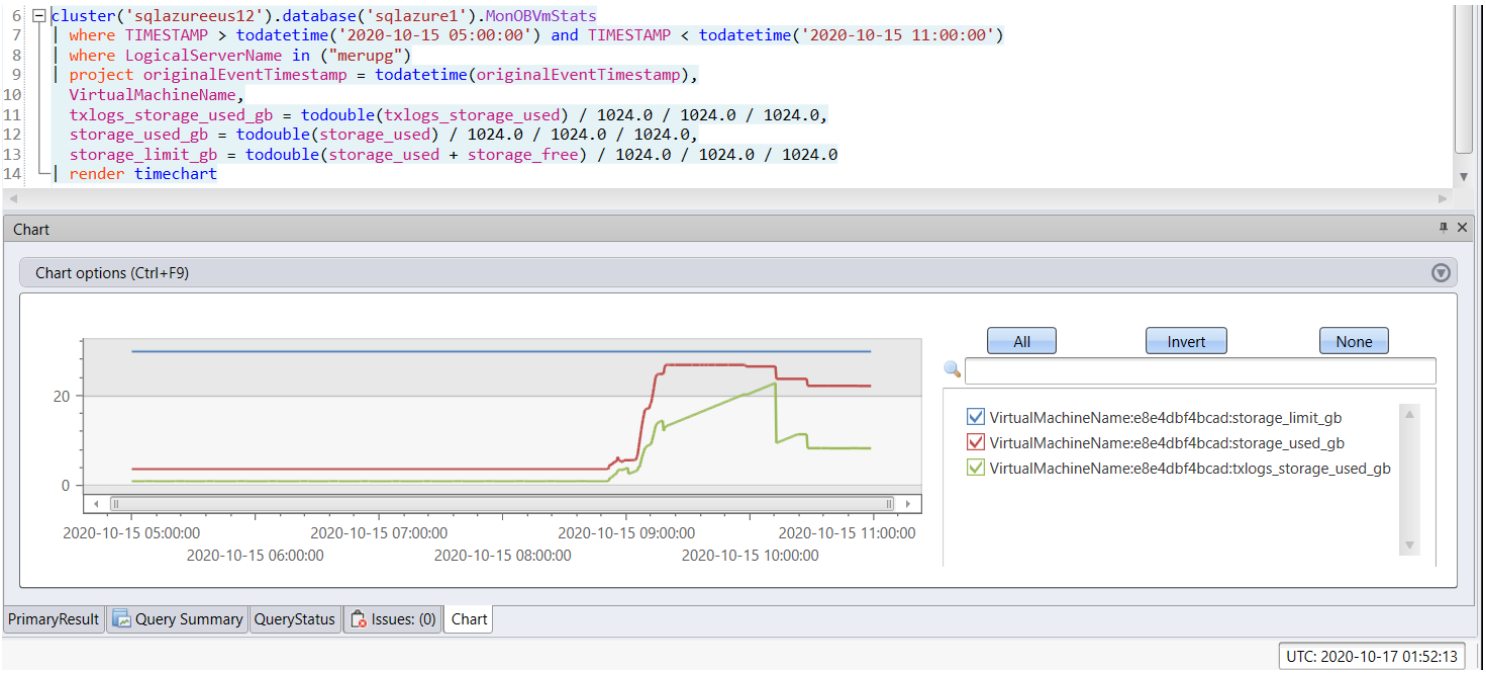


2.1 Check PG Server Storage Usage and Total WAL File Size

Kusto query.

```
cluster('sqlazureeus12').database('sqlazure1').MonOBVmStats
| where TIMESTAMP > todatetime('2020-10-15 05:00:00') and TIMESTAMP < todatetime('2020-10-15 11:00:00')
| where LogicalServerName in ("merupg")
| project originalEventTimestamp = todatetime(originalEventTimestamp),
  VirtualMachineName,
  txlogs_storage_used_gb = todouble(txlogs_storage_used) / 1024.0 / 1024.0 / 1024.0,
  storage_used_gb = todouble(storage_used) / 1024.0 / 1024.0 / 1024.0,
  storage_limit_gb = todouble(storage_used + storage_free) / 1024.0 / 1024.0 / 1024.0
| render timechart
```

Sample output.



2.2 Check PG Side Car Log for WAL File Ready Count

Kusto query.

```
cluster('sqlazureeus12').database('sqlazure1').OBvmagentsidecarpgsql
| where TIMESTAMP > todatetime('2020-10-15 05:00:00')
| where LogicalServerName in ("merupg")
| project originalEventTimestamp, VirtualMachineName, LogLevel, MessageString
| where MessageString contains "Found" and MessageString contains "WAL files"
| order by originalEventTimestamp asc
```

Sample output.

```
originalEventTimestamp  VirtualMachineName      LogLevel        MessageString
......
2020-10-15 08:51:01.000000      e8e4dbf4bcad    1       [WalUploader].MoveNext: Found 0 'ready' WAL files for
2020-10-15 08:51:31.000000      e8e4dbf4bcad    1       [WalUploader].MoveNext: Found 0 'ready' WAL files for
2020-10-15 08:52:01.000000      e8e4dbf4bcad    1       [WalUploader].MoveNext: Found 21 'ready' WAL files for
2020-10-15 08:54:06.000000      e8e4dbf4bcad    1       [WalUploader].MoveNext: Found 25 'ready' WAL files for
2020-10-15 08:56:31.000000      e8e4dbf4bcad    1       [WalUploader].MoveNext: Found 58 'ready' WAL files for
2020-10-15 09:01:36.000000      e8e4dbf4bcad    1       [WalUploader].MoveNext: Found 57 'ready' WAL files for
2020-10-15 09:06:45.000000      e8e4dbf4bcad    1       [WalUploader].MoveNext: Found 139 'ready' WAL files fo
2020-10-15 09:18:40.000000      e8e4dbf4bcad    1       [WalUploader].MoveNext: Found 654 'ready' WAL files fo
2020-10-15 10:13:27.000000      e8e4dbf4bcad    1       [WalUploader].MoveNext: Found 123 'ready' WAL files fo
2020-10-15 10:24:02.000000      e8e4dbf4bcad    1       [WalUploader].MoveNext: Found 1 'ready' WAL files for
2020-10-15 10:24:32.000000      e8e4dbf4bcad    1       [WalUploader].MoveNext: Found 0 'ready' WAL files for
2020-10-15 10:25:02.000000      e8e4dbf4bcad    1       [WalUploader].MoveNext: Found 0 'ready' WAL files for
......
```

Note: [1] **We see that WAL file count go up after heavy workload begins and go back near zero after heavy workload ends.**

2.3 WAL File Ready Count and Upload Progress

Kusto query.

```
cluster('sqlazureweu2').database('sqlazure1').OBvmagentsidecarpgsql
| where TIMESTAMP > todatetime('2020-10-16 10:55:47')
| where LogicalServerName in ("flexserver")
| where tolower(MessageString) contains "wal"
| project originalEventTimestamp, VirtualMachineName, LogLevel, MessageString
| extend new_iteration = iff(MessageString contains "[WalUploader].MoveNext: Found", 1, 0)
| extend ready_count = toint(extract("\\[WalUploader\\].MoveNext: Found (\\d+) 'ready' WAL files", 1, MessageS
| extend uploaded_wal_file = extract("\\[Archive\\].UploadWalFiles: Succeed: ([\\w\\.]+) ", 1, MessageString)
| order by originalEventTimestamp asc | serialize iteration = row_cumsum(new_iteration, VirtualMachineName !=
| summarize min_ts = min(originalEventTimestamp), max_ts = max(originalEventTimestamp),
  ready_count = any(ready_count), newly_uploaded_count = countif(isnotempty(uploaded_wal_file)) by VirtualMach
| order by min_ts asc
| where ready_count != 0 or newly_uploaded_count != 0
```

Sample output.

```
VirtualMachine  iteration       min_ts  max_ts  ready_count     newly_uploaded_count
cba2ba7ec5e5    3       2020-10-16 10:57:17.000000      2020-10-16 10:58:00.000000      45      45
cba2ba7ec5e5    4       2020-10-16 10:58:00.000000      2020-10-16 10:58:55.000000      73      73
cba2ba7ec5e5    5       2020-10-16 10:58:55.000000      2020-10-16 11:00:03.000000      90      90
cba2ba7ec5e5    6       2020-10-16 11:00:04.000000      2020-10-16 11:01:19.000000      107     107
cba2ba7ec5e5    7       2020-10-16 11:01:19.000000      2020-10-16 11:02:31.000000      100     100
cba2ba7ec5e5    8       2020-10-16 11:02:31.000000      2020-10-16 11:03:52.000000      104     104
cba2ba7ec5e5    9       2020-10-16 11:03:52.000000      2020-10-16 11:05:10.000000      115     115
cba2ba7ec5e5    10      2020-10-16 11:05:10.000000      2020-10-16 11:06:02.000000      109     109
cba2ba7ec5e5    11      2020-10-16 11:06:17.000000      2020-10-16 11:06:58.000000      40      40
cba2ba7ec5e5    12      2020-10-16 11:06:58.000000      2020-10-16 11:08:51.000000      191     191
cba2ba7ec5e5    13      2020-10-16 11:09:06.000000      2020-10-16 11:13:27.000000      382     382
cba2ba7ec5e5    14      2020-10-16 11:13:27.000000      2020-10-16 11:22:45.000000      779     779
......
```

3 Check PG Engine Insert, Update and Delete

Kusto query.

```
cluster('sqlazureeus12').database('sqlazure1').MonOBPgSqlTransactionStats
| where TIMESTAMP > todatetime('2020-10-15 05:00:00') and TIMESTAMP < todatetime('2020-10-15 11:30:00')
| where LogicalServerName in ("merupg")
| summarize sum(Tup_inserted), sum(Tup_updated), sum(Tup_deleted), sum(Commits) by VirtualMachine = VirtualMac
```

Sample output.

VirtualMachine	TIMESTAMP	sum_Tup_inserted	sum_Tup_updated	sum_Tup_deleted	sum_Commits
e8e4dbf4bcad	2020-10-15 05:13:20.0000000	48666306	459	278	2580265
e8e4dbf4bcad	2020-10-15 05:28:20.0000000	48666306	459	278	2580949
e8e4dbf4bcad	2020-10-15 05:43:30.0000000	48666306	459	278	2581616
e8e4dbf4bcad	2020-10-15 05:58:40.0000000	48666306	459	278	2582271
e8e4dbf4bcad	2020-10-15 06:13:40.0000000	48666306	459	278	2582975
e8e4dbf4bcad	2020-10-15 06:28:50.0000000	48666306	459	278	2583685
e8e4dbf4bcad	2020-10-15 06:43:50.0000000	48666306	459	278	2584379
e8e4dbf4bcad	2020-10-15 06:59:00.0000000	48666306	459	278	2585087
e8e4dbf4bcad	2020-10-15 07:14:10.0000000	48666306	459	278	2585695
e8e4dbf4bcad	2020-10-15 07:29:10.0000000	48666306	459	278	2586312
e8e4dbf4bcad	2020-10-15 07:44:20.0000000	48666306	459	278	2586917
e8e4dbf4bcad	2020-10-15 07:59:20.0000000	48666306	459	278	2587537
e8e4dbf4bcad	2020-10-15 08:14:30.0000000	48666306	459	278	2588144
e8e4dbf4bcad	2020-10-15 08:29:30.0000000	48666306	459	278	2588856
e8e4dbf4bcad	2020-10-15 08:44:40.0000000	48666306	459	278	2589555
e8e4dbf4bcad	2020-10-15 09:01:20.0000000	90666423	463	344	2590305
e8e4dbf4bcad	2020-10-15 10:13:10.0000000	145631136	594	347	2592989
e8e4dbf4bcad	2020-10-15 10:33:50.0000000	145631136	594	347	2593854
e8e4dbf4bcad	2020-10-15 10:48:50.0000000	145631136	594	347	2594529
e8e4dbf4bcad	2020-10-15 11:04:00.0000000	145631136	594	347	2595195
e8e4dbf4bcad	2020-10-15 11:19:00.0000000	145631136	594	347	2595878

Note: [1] We see that a lot of rows are inserted into tables between 2020-10-15 08:44:40 UTC and 2020-10-15 10:13:10 UTC.

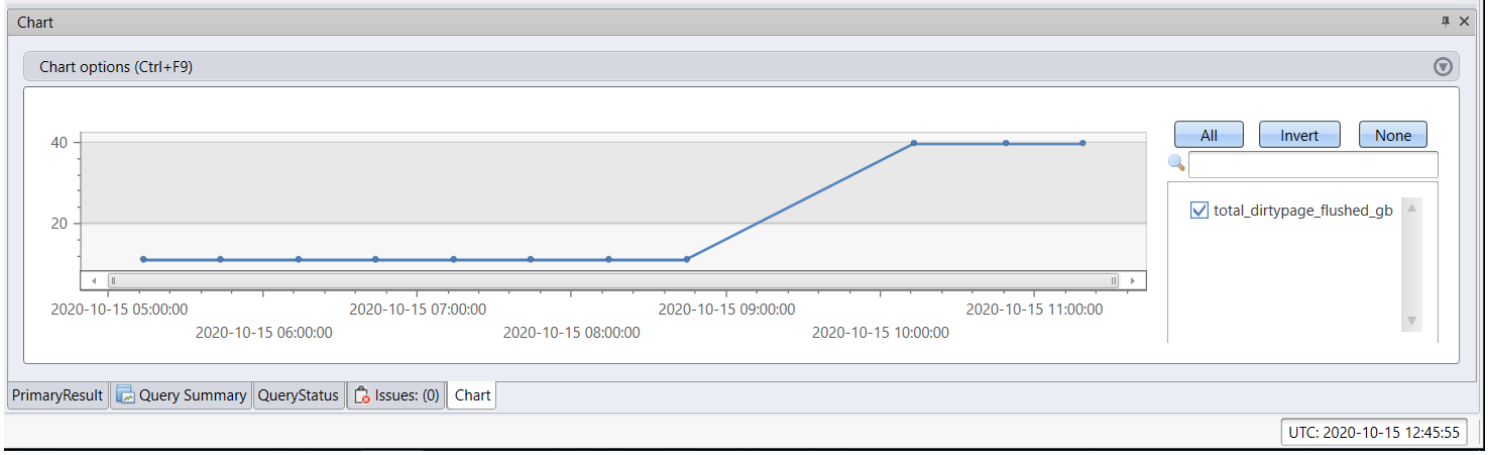
4 Check PG Engine Dirty Page Flush

Kusto query.

```
cluster('sqlazureeus12').database('sqlazure1').MonOBPgSqlBuffers
| where TIMESTAMP > todatetime('2020-10-15 05:00:00') and TIMESTAMP < todatetime('2020-10-15 11:30:00')
| where LogicalServerName in ("merupg")
| project originalEventTimestamp = todatetime(originalEventTimestamp),
  total_dirtypage_flushed_gb = todouble(BuffersCheckpoint + BuffersBackend) * 8192.0 / 1024.0 / 1024.0 / 1024.
| render timechart
```

Sample output.

```
cluster('sqlazureeus12').database('sqlazure1').MonOBPgSqlBuffers
| where TIMESTAMP > todatetime('2020-10-15 05:00:00') and TIMESTAMP < todatetime('2020-10-15 11:30:00')
| where LogicalServerName in ("merupg")
| project originalEventTimestamp = todatetime(originalEventTimestamp),
  total_dirtypage_flushed_gb = todouble(BuffersCheckpoint + BuffersBackend) * 8192.0 / 1024.0 / 1024.0 / 1024.0
| render timechart
```



Note: [1] We see that during the heavy load period about 28 GB (from 11 GB in total to 39 GB in total) dirty pages were flushed from shared buffer to disk, which indicates that a lot of WAL files are generated at the same time.

If you have any doubts when following this page, please reach out to *xixia@microsoft.com* for clarification and wiki/TSG improvement.