Import Export - How to check queue size

Last updated by | Vitor Tomaz | Feb 24, 2023 at 3:26 AM PST

Contents

This TSG is a copy of the TSG [IE2001.1] Check Queue Length as it is maintained by the SQL Engineering PG.

Overview

The Import/Export service is a shared resource in the sense that there are a limited number of worker VMs in any given region. One reason that a job may appear "stuck" to the customer is simply that it is enqueued and waiting to be processed.

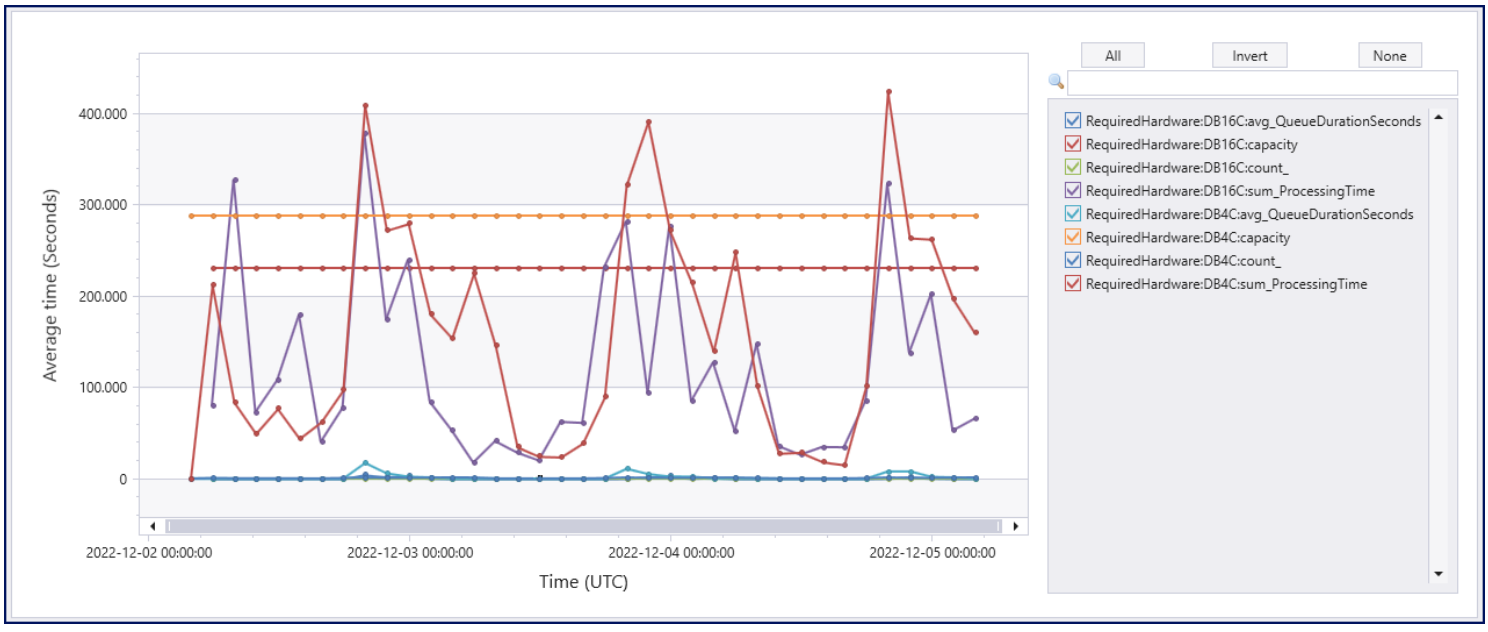
If there is no progress for an Import or Export operation and the operation appears to be stuck, you can check the Import/Export queue size for a specific time in a region to see if the operation might be queued and is waiting for a time slot. A long queue might cause high queue times for the Import/Export requests, and it can dramatically increase the duration of the operation due to long waiting time on the queue.

Check the queue size in Kusto

To use this TSG you only need to know the region name. In case of export, it will be helpful to know the used size of the DB being exported.

```
// Chart of queue time
// Resulting table shows the time the request was enqueued on the y-axis and the amount of time it was in the
// If you see the "sum_ProcessingTime" go past the "capacity" for a particular hardware type, it means that a
// If the "sum_ProcessingTime" is more than the capacity "forever", the queue will never drain.
// At some point before "forever", customers will start to notice and complain.
let binWindowHours = 2;
let secondsPerHour = 60 * 60;
let baseData = materialize(MonImportExport
| distinct MachineName
// Count the machines which have reported doing work recently. This will _under_ count if the region is lightl
// and some machines have not serviced jobs within the telemetry retention window.
| summarize num4CWorkers = countif(MachineName startswith "DB4C"), num16CWorkers = countif(MachineName startsw
);
let num4CWorkers = toscalar(baseData | project num4CWorkers);
let num16CWorkers = toscalar(baseData | project num16CWorkers);
MonManagementOperations
| where TIMESTAMP > ago(7d)
| where operation_type in ('ImportToNewDatabase', 'ImportToExistingDatabase', 'ExportDatabase')
| where event == "management_operation_start"
| extend crRequestTimestamp = originalEventTimestamp
| project-away originalEventTimestamp
| join kind=inner (MonImportExport // Determine the time the job started based on the first traces from the TR
        | summarize trStartTimestamp=arg_min(originalEventTimestamp, originalEventTimestamp), trEndTimestamp=a
        | extend request_id = toupper(request_id), ProcessingTime = (trEndTimestamp - trStartTimestamp)
        | project request_id, NodeRole, trStartTimestamp, ProcessingTime
    )  on request_id
| extend QueueDurationSeconds = (trStartTimestamp - crRequestTimestamp)
| extend RequiredHardware = NodeRole1
| summarize avg(QueueDurationSeconds), sum(ProcessingTime), count() by bin(crRequestTimestamp, binWindowHours
| extend capacity = iif(RequiredHardware == "DB4C", binWindowHours * secondsPerHour * num4CWorkers, binWindowH
| render timechart with (ytitle="Average time (Seconds)", xtitle="Time (UTC)")
```

Sample output:



Interpreting the graph

The graph should have six series per region in the Kusto cluster, one for each hardware type (DB4C and DB16C), and one for total count, total processing time, and one for queue time.

The thing which indicates a capacity problem would be if the queue time never goes down to zero in any given 24-hr interval and is instead growing infinitely. If you see that the queue time is simply high, this does not necessarilly indicate a problem (since there is no SLA on Import/Export requests).

List operations which are in the queue

You can use the following CMS query to show operations which are waiting in the queue

```
SELECT request_id, logical_server_name, logical_database_name, dacpac_action_name, instance_name, create_time,
FROM dacpac_actions
WHERE state = 'WaitingForCreatingApplication'
```

More Information

In every region, the Import/Export service has two types of VMs deployed, one with less disk space (~200GB) and one with more (~600GB).

As a rule of thumb, we assume that an Import or Export request will take 3x the local disk space as compared to the size of the database so for small databases we use the smaller VMs and for large database, we use the larger VMs. In this case, we define "large" to be any database greater than 40GB (to make sure none will fail).

All import requests use the larger VMs since we can't estimate in advance how large the DB will be.

How good have you found this content?