

# BigFix 10 Infrastructure Monitoring

Authors: Mark Leitch & Davide Cosentino @ HCL Technologies

BigFix provides several methods to monitor and understand the health of a deployment, including system health dashboards and relevance. However, at times deeper introspection is required into the infrastructure BigFix runs upon.

This document will describe three infrastructure monitoring approaches.

1. System resource utilization.
2. The SQL package cache.
3. SQL fragmentation.

Through the described monitors, it is possible to provide:

1. System monitoring results under suitable load for the BigFix root and database servers.
  - If these servers are not collocated (i.e., they are on separate operating system instances), two invocations of the monitor will be required (i.e., one invocation for each discrete system).
2. Package cache results under suitable workload for the database server.
3. A report on the current database fragmentation.

# 1 SYSTEM RESOURCE UTILIZATION

---

The BigFix Performance Toolkit (see the “Further Reading” section) provides a monitor wrapper utility known as MXPerfmon. This utility generates system specific monitoring commands that may be of value on any node in a BigFix deployment.

The following sample Windows monitor commands were generated by MXPerfmon and may be used directly to monitor system resources. Some notes on this monitor follow.

- You need to run an instance on each machine from an elevated shell.
- The sample monitor samples every 60 seconds for 1440 iterations, which is equivalent to 24 hours.
  - The monitor frequency may be adjusted, but it is recommended not to go below a five (5) second interval or have an excessive number of data points. For example, in the ballpark of 1000 data points is plenty.
- It will create a system monitor output BLG file in typically the C:\Perflogs directory.
- The monitor is named BFMon and is capped at 250MB.

```
logman delete BFMon

logman create counter BFMon -f bincirc -v mmdhmm -max 250 -c "\LogicalDisk(*)\"
"\Memory\" "\Network Interface(*)\" "\Paging File(*)\" "\PhysicalDisk(*)\"
"\Processor(*)\" "\Process(*)\" "\Redirector\" "\Server\" "\System\"
"\Thread(*)\" "\SQLServer:Databases(*)\" "\SQLServer:Locks(*)\"
"\SQLServer:Transactions(*)\" "\SQLServer:Wait Statistics(*)\"
"\SQLServer:Availability Replica(*)\" "\SQLServer:Database Replica(*)\" -si 60 -
sc 1440

logman start BFMon
```

*Figure 1: MXPerfmon Windows Monitor*

Additional notes:

- The SQL counters only apply to the SQL node, and the name of the SQL counters may not match your environment. It is best to inspect your local system to confirm and possibly update the counter names.
- It is possible to include other options, such as TCP/IP/UDP counters, using the provided MXPerfmon utility.
- Once the BLG file has been generated, it may be compressed and forwarded to product support. The file typically compresses well (e.g., an 8x compression rate).

## 2 THE SQL PACKAGE CACHE

---

The SQL package cache provides a snapshot of all currently “active” MS SQL transactions. For database experts, this snapshot can prove invaluable. The following query may be used to provide a current snapshot of the package cache. Note the “ORDER BY” predicate may be altered based on the desired ordering of results.

```
SELECT TOP 100 SUBSTRING(qt.TEXT, (qs.statement_start_offset/2)+1,
((CASE qs.statement_end_offset
WHEN -1 THEN DATALENGTH(qt.TEXT)
ELSE qs.statement_end_offset
END - qs.statement_start_offset)/2)+1) AS query_text,
qs.execution_count,
qs.total_rows, qs.last_rows, qs.total_rows / qs.execution_count avg_rows,
qs.total_dop, qs.last_dop, qs.total_dop / qs.execution_count avg_dop,
qs.total_logical_reads, qs.last_logical_reads,
qs.total_logical_writes, qs.last_logical_writes,
qs.total_physical_reads, qs.last_physical_reads,
qs.total_worker_time, qs.last_worker_time,
qs.total_worker_time / qs.execution_count avg_worker_time,
qs.total_elapsed_time / 1000 total_elapsed_time_in_ms,
qs.last_elapsed_time / 1000 last_elapsed_time_in_ms,
qs.total_elapsed_time / qs.execution_count / 1000 avg_elapsed_time_in_ms,
qs.creation_time, qs.last_execution_time, qp.query_plan
FROM sys.dm_exec_query_stats qs
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) qt
CROSS APPLY sys.dm_exec_query_plan(qs.plan_handle) qp
ORDER BY avg_elapsed_time_in_ms DESC -- CPU avg
```

Figure 2: SQL Package Cache

## 3 SQL FRAGMENTATION

---

The SQL fragmentation query provides a data object breakdown of fragmentation counts. It can be used to determine the health of the database and the associated index management scripts provided by BigFix. Please see the BigFix Maintenance Guide in the “Further Reading” section for comprehensive information on the BigFix index management scripts.

```
select s.name,o.name,i.name,ips.avg_fragmentation_in_percent,ips.page_count
from sys.objects o left outer join sys.schemas s on
o.schema_id= s.schema_id left outer join sys.indexes i on
o.object_id=i.object_id left outer join sys.dm_db_index_physical_stats (db_id(),
NULL, NULL, NULL, NULL) AS IPS
on i.object_id=IPS.object_id and i.index_id=ips.index_id
where o.type='U' and i.index_id > 0 order by avg_fragmentation_in_percent desc
```

Figure 3: SQL Fragmentation

## 4 FURTHER READING

---

In the event further reading is desired, the following technical resources are available.

BigFix Platform Documentation: [URL](#)

BigFix Capacity Planning Guide: [URL](#)

BigFix Maintenance Guide: [URL](#)

BigFix Performance Toolkit: [URL](#)