BigFix Performance Management

an HCL Product

HCLSoftware

Document version 11.x.6

Mark Leitch
BigFix Development

CONTENTS

LIST OF FIGURES

REVISION HISTORY

Date	Version	Revised By	Comments
January 1st, 2025	11.x.6	MDL	First branch off Capacity Planning guide.

Figure 1: Revision History

THANKS

Thanks to the BigFix development team for their continued excellence and for producing such a well performing and useful product. Thanks also go out to the incredible BigFix customers, who have helped to make the product what it is today and have provided feedback to improve this guide.

Special thanks to the BigFix performance team. Their hard work has made BigFix scale higher, go faster, and work better than ever. Notable members of the performance team include the following (in alphabetical order).

- Antonio Mangiacotti
- Donatella Girolamo
- Massimo Marra
- Paolo Cavazza
- Valeria Mazza

1 Introduction

Capacity planning involves the specification of the various components of an installation to meet customer requirements, often with growth or timeline considerations. BigFix offers endpoint lifecycle and security management for large scale, distributed deployments of servers, desktops, laptops, and mobile devices across physical and virtual environments.

Capacity planning and performance management go hand in hand. There is no standard workload for BigFix. Every enterprise has different requirements, infrastructure, and customization. This guide will build upon the base capacity planning recommendations and offer a set of defined decision points for building an optimal BigFix deployment based on the business needs.

The following documents are considered to offer a suite of performance, capacity, and maintenance reference information for BigFix. See the References section for relevant URLs.

- BigFix Performance Management (this document).
- BigFix Capacity Planning (part of a prior incarnation of this document, since extracted for brevity and clarity).
- BigFix Maintenance Guide.
  A set of configuration and maintenance recommendations for BigFix (also part of a prior incarnation of this document).
- BigFix Non-Functional Requirements: A Checklist Approach
  A set of guidelines and recommendations for BigFix performance and security. Highly recommended for practical deployment management.
- MX Performance Toolkit for BigFix.
  A set of tools and performance management approaches for BigFix.

Note: This document is considered a work in progress. Performance recommendations will be refined and updated as new BigFix releases are available. While the guide in general is considered suitable for all BigFix Version 10 and 11 releases, with new content as noted, it is best oriented towards BigFix Version 11.0.3 onwards. In addition, several references are provided in the References section. These papers are highly recommended for readers who want detailed knowledge of BigFix server configuration, architecture, and capacity planning.

2 BigFix Overview

An overview of BigFix will be provided from the following perspectives:

1. Functional.
2. Architectural.

2.1 Functional Overview

The BigFix portfolio provides a comprehensive security solution encompassing several operational areas. These areas include the following.

- Lifecycle management (asset discovery and inventory, software distribution, patch management, operating system deployment, remote control).
- Security and compliance (security configuration management, vulnerability management, patch management, anti-virus and anti-malware client management, network self-quarantine).
- Patch management.
- Power management.
- Software use analysis.
- Core protection.
- Server automation.

In general, BigFix spans the broadest OS and device set in the industry, including the management of physical and virtual servers, PCs, Macs, tablets, smartphones, embedded and hardened devices, and point of sale devices. This is managed via a scalable distributed infrastructure that includes a lightweight dedicated agent. We will describe this infrastructure in the architectural overview section.

2.2 Architectural Overview

The following diagram provided a basic view of the BigFix architecture.

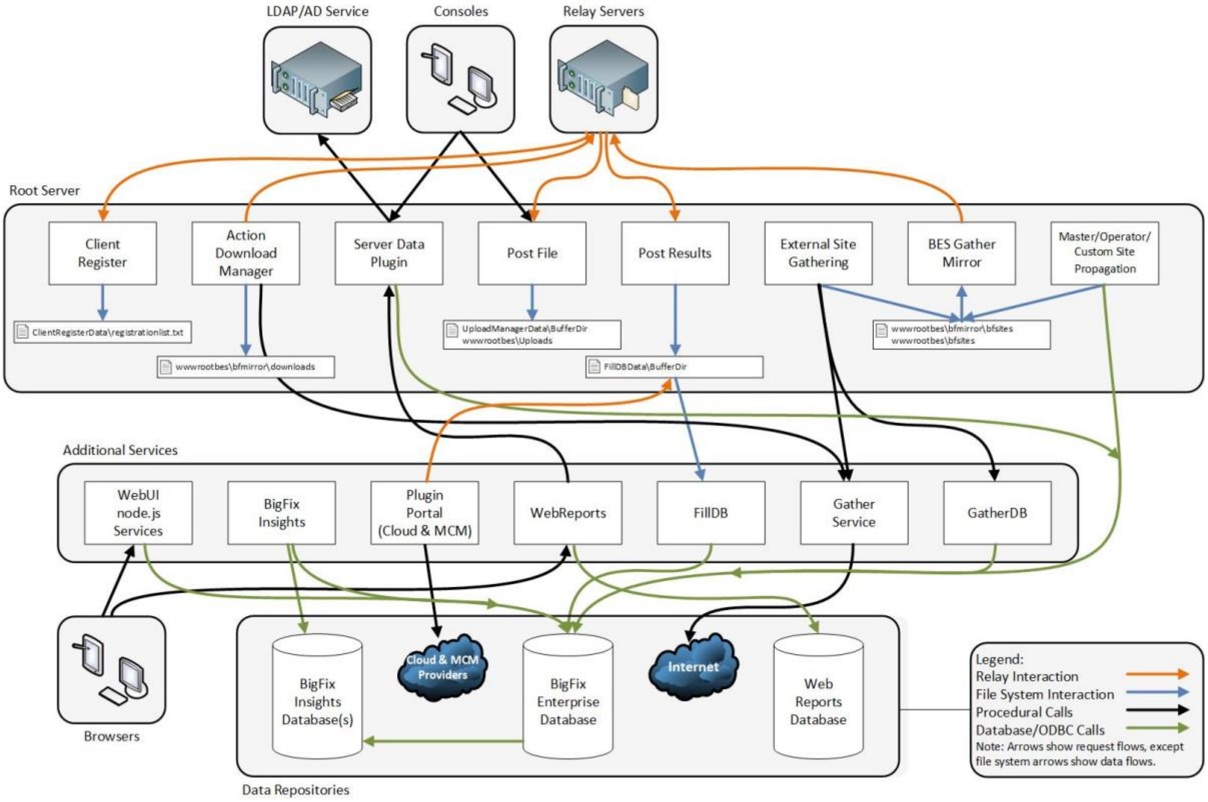

Figure 2: BigFix Architecture

The notable components of this diagram follow.

- Root Server.
  The base BigFix Enterprise Server. It is comprised of a set of core services as identified.
- Console.
  A management console (user interface) for BigFix operators. The console is a Windows only application. A console server is used to support one or more instances of the BigFix Console.
- The WebUI.
  A Node.js instance with an associated database intended to support the Web based user interface.
- The Web Reports and BES Explorer Services.
  The Web Reports service can provide a variety of stock and custom reports for one or more BigFix server installations. With BigFix version 11.0.2, the Web Reports service has been refactored with the introduction of the BES Explorer service. A future version of this document will revise the architecture diagram.
- Relays.
  A distributed, hierarchical infrastructure to manage the deployment of BigFix agents across diverse network topologies.
- Plugin Portal (new for BigFix 10).
  The Plugin Portal provides the common server infrastructure for the Cloud and Modern Client Management (MCM) technologies delivered in BigFix 10. The initial cloud offering provides support for VMware, Azure, and Amazon Web Services (AWS) deployments. The MCM capability applies to Android, Windows, iPhone, iPad, and Mac deployments.

  Note: The full MCM stack is not shown in the architecture figure.

- BigFix Insights (new for BigFix 10).
  BigFix Insights offers new reporting capability, typically based on a standalone data replica to isolate its workload from the BigFix root server. It is accompanied by new reporting content in the WebUI, to improve the visualization and understanding of BigFix deployments.
- Agents (as part of the client population).
  A lightweight, native agent that manages the endpoint.
- Content (Fix or Fixlet Servers, represented via the Internet content).
  These servers are used as the object repository for all client content (Fixes, Fixlets, tasks, baselines, and analyses). In addition, dashboards, wizards, and WebUI applications are delivered via the servers. Content is utilized by the agent to determine relevance, compliance, and remediation activities.
- Disaster Server Architecture (aka DSA, not shown).
  The DSA is a server replication architecture intended to provide fault tolerance. DSA is discussed in more detail under high availability recommendations.
- The Database Management Server, or DBMS (either Microsoft SQL Server or IBM DB2 for Linux, UNIX, and Windows, also referred to as DB2 LUW).

In general, anti-collocation is possible for the core BigFix server components. For example, the root server, Web Reports, BES Explorer, WebUI, and DBMS may all be distinct physical or virtual server instances. The pros and cons of anti-collocation are described later in this document.

3 Performance Management

We will first describe some general performance management approaches. We will describe several management and configuration recommendations at the BigFix, database, operating system, and hypervisor levels. Database maintenance recommendations are provided in the separate maintenance guide (see the References section).

3.1 Performance Management Approaches

Several generic approaches will be described, ranging from ease of management through a decision tree for managing performance.

3.1.1 Performance vs Ease of Management

- In pure performance terms, the ideal environment is a physical deployment with multiple distinct NVME based flash drives. This offers maximum efficiency and parallelism, and scales in a deterministic way.
- The alternative approach is for ease of management: a virtual deployment with remote storage subsystems, and pooled database management services. This offers maximum density, and easily partitions across multiple teams within the business.
- Every installation needs to assess these trade-offs, and hybrid approaches are often used. It is possible to combine performance and ease of management, based on understanding and managing the overhead associated with ease of management.

3.1.2 The Storage Revolution

- When a system is perceived as running slow, the first reaction is to throw more processor cores at the problem. However, in approximately 80% of cases, the limiting factor is the storage subsystem.
- While improvements in processor capability have made headlines for decades, the most exciting improvements have been in storage.
- Most are aware of the improvements in flash-based storage, leaving behind solutions that required spinning magnetic platters with moving heads. However, the interfaces associated with the storage improvements have improved dramatically.
- Interfaces have migrated from IDE to AHCI to, at present, NVME. The NVME interface offers greater parallelism with reduced application and system load.
- The best thing you can do for your BigFix deployment is implement NVME flash storage.

3.1.3 When to Throw Hardware at the Problem

We will comment on simple approaches of “throwing hardware” at the problem.

- CPU:
  The first temptation is to throw CPU at the problem. On a physical system, improvements in the number of cores or clock speed can drive improvements. However, on a virtual system, allocating more cores than needed can put pressure on the hypervisor scheduler and degrade performance. CPU “upgrades” should be done carefully with awareness of the base problem.
- Storage:
  Storage improvements are always beneficial. Whether adding device paths, or upgrading storage, it is a beneficial move. However, storage should always be benchmarked. Just because a solution looks good on paper, does not mean it is configured and deployed correctly.

- Network:
  Network resource improvements are always beneficial, though often difficult to manage beyond the data center.
- Memory:
  Modern systems will generally use all the memory you can throw at them. If you have it, use it.

3.1.4 Collocation and Anti-collocation Options

As stated, the ideal server implementation is a single node physical instance. However, there can be good motivation for not collocating components. The ultimate decision is typically determined by organization structure and business needs.

- A general benefit of collocation in physical deployments is database communication paths tend to be optimal and advantages like shared memory may be exploited.
- A general benefit of anti-collocation in virtual deployments is smaller and easier to manage virtual machines may be deployed.
- The database server: A primary benefit of anti-collocation is being placed in a database administrator (DBA) managed pool with monitoring, backup support, etc.
- The WebUI server: there is no inherent advantage in running a standalone WebUI server, beyond the general benefits already described.
- Web Reports and BES Explorer: For these components, it can be advantageous to run multiple standalone instances for scalability and resource isolation.

3.2 BigFix Management Recommendations

We will provide configuration recommendations for the BigFix components.

3.2.1 BigFix FillDB Configuration

- The FillDB service has a “database boost” parameter that is key for maximum throughput.
  - For Windows, the boost is enabled by default for BigFix 9.5.10 onwards.
  - For Linux, the boost is enabled by default for BigFix 9.5 onwards.
  - For all prior releases, please consult the version specific product documentation.
- In the BigFix 9.5.5 release, a parallel FillDB implementation was delivered. The following parameters are managed, with the first set being for the base FillDB capability, and the second set being for the BigFix Query result processing capability that is also managed by FillDB.
  - ParallelismEnabled
    NumberOfParsingThreads
    NumberOfDBUpdatingThreads
  - ParallelismEnabledForQuery
    NumberOfParsingThreadsForQuery
    NumberOfDBUpdatingThreadsForQuery
  - The parallelism is enabled by default by the BigFix installer based on the hardware capability (number of cores). For example, there is a base report concurrency of [3 readers, 3 writers] if the machine has 6+ cores. If the machine has 10+ cores, the BigFix Query report concurrency is also set to [3, 3].
  - When the base parallelism is enabled, FillDB throughput doubles with a nominal increase of 1 to 1.5 cores. Similarly, for BigFix Query, the parallelism can essentially cut the processing time for large result sets in half. Both of these are significant

improvements, with no modification required by the user.

- It is possible in an environment with significant available system capability, that more parallelism may be enabled by the user with additional throughput improvements. However, caution should be used. The default parallelism provides an excellent combination of high throughput with low resource impact.
- The following figure provides an example of parallelism improvements. Here modest improvements in parallelism are in evidence as we proceed beyond the default of “3/3”. However, once you go beyond “10/10”, degradation is in evidence.

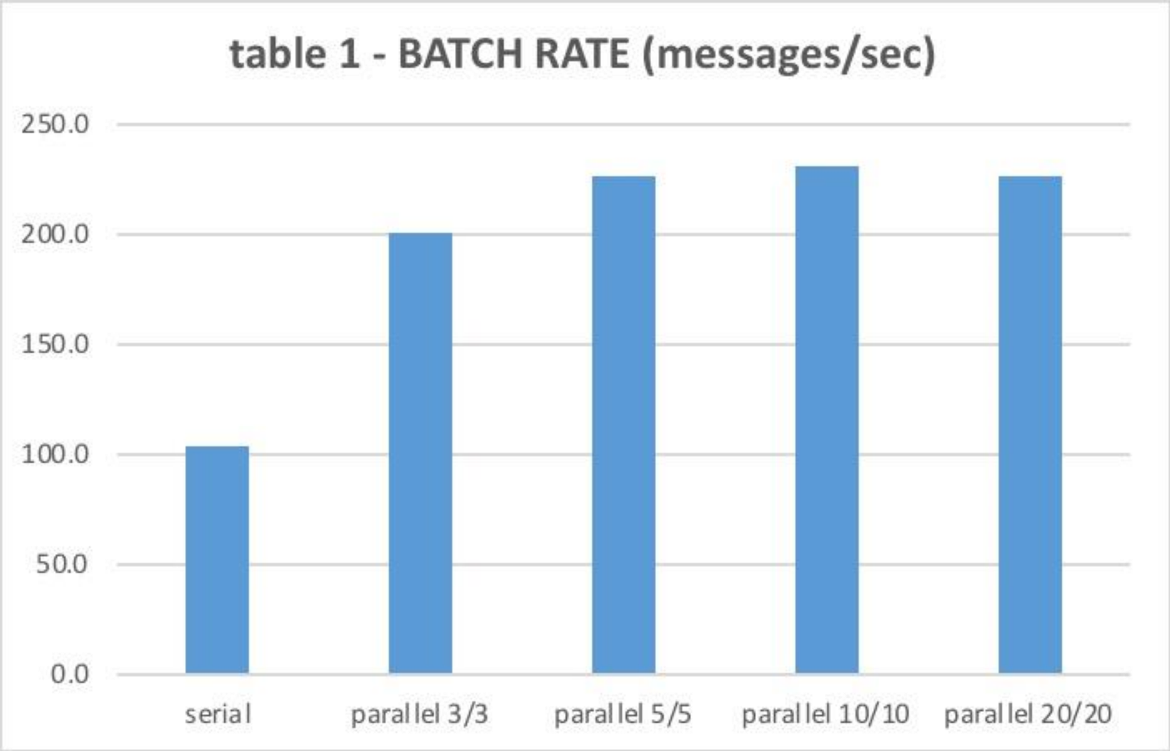

Figure 3: FillDB Parallelism Impact

3.2.2 BigFix Relay Configuration

- The _BESRelay_HTTPServer_MaxConnections configuration parameter will determine the maximum number of concurrent connections for the relay. This defaults to 2048 and may be changed for extreme high concurrency scenarios.
- For Windows it is possible that altering the TCP/IP configuration may be beneficial. See section 3.6.1 for more information on Windows TCP/IP management.

3.2.3 Agent Heartbeat Frequency

- The agent sends a heartbeat to the server (essentially, reporting in), every 15 minutes. As a BigFix deployment scales, the heartbeat activity can become significant. For example, if 250,000 agents are reporting every 15 minutes, that is over 278 heartbeats per second!
- In order to mitigate this, the general recommendation is to set the heartbeat interval to 15 minutes for every 10,000 agents. For example, for a 250,000 agent deployment, this would mean a heartbeat on the order of 6 hours. At this setting, there would be on the order of 12 heartbeats a second. This is much more manageable.
- The heartbeat may be set in the BigFix console preference section.

3.2.4 Console Currency

Major performance improvements in console load times have been delivered in the 10.08 and 11.0.2 streams. For example, 11.0.2 customers report a 50% reduction in load times, though experience varies based upon deployment specifics. Further improvements are pending in the version 11 stream. It is highly recommended to upgrade to take advantage of these improvements.

3.2.5 Console Refresh Frequency

- The BigFix console refresh frequency determines the interval for populating (refreshing) the console cache. The default value is 15 seconds, meaning every 15 seconds there is a database workload (impacting CPU, disk, network, etc.) in order to maintain the cache.
- Setting the refresh frequency is a trade-off between data currency and system workload. The setting is ultimately derived from the number of console operators, the size of the estate, and the desired user experience. For a large estate with many concurrent users, a refresh frequency of 300 seconds is reasonable.
- The console refresh frequency may be set in the BigFix console preference section.

3.2.6 Data Archiving

A variety of data archiving approaches are available and should be performed at regular intervals.

- Deletion of closed and expired actions.
  This is a console activity that will remove the actions from the console view. While the action will still persist in the database, this cleanup approach will reduce the console workload.
- Computer removal.
  A computer removal utility (URL) is available to remove obsolete computers and thereby reduce the overhead for database operations.
- Audit cleanup.
  An audit cleanup utility (URL) is available to prune entries based on a variety of criteria, including time. The audit cleanup should be done in accordance with the enterprise audit policies. For example, a database archive may be generated to store the audit content for that point in time, with the subsequent cleanup serving to reduce the overhead for database operations.
- The BigFix 9.5 release offers significant improvements for data archiving and management.
  - Database cleanup tools.
    You may use the BESAdmin interface on Windows systems or the BESAdmin command line on Windows and Linux systems to remove data about computers, custom Fixlets, properties, analyses, and actions and to update the PropertyIDMap table with changes.
  - FillDB log rotation.
    The log rotation feature is active by default with LogFileSizeLimit set to 100 MB.

3.2.7 WebUI

The WebUI is configured to perform well “out of the box”. Some recommended guidelines follow.

- Improvements in BigFix 9.5.10 provide a superior base for the WebUI deployment, with improved maintenance operations and transaction concurrency management. As a result, the WebUI should be deployed on a minimum of a 9.5.10 base.
- The WebUI employs significant caching and queue management. For example, several tables under the WebUI schema will be in evidence in the BigFix Enterprise database. There is also a stored procedure queue workload manager governed via a Stored Procedure Queue Concurrency (SPQC) setting for the WebUI. This setting is pre-configured and should not be

adjusted without deep understanding. For example, reducing the setting will result in longer refresh times, while increasing the setting will impact concurrency and locking.

- For best performance, the recommended MS SQL configuration settings should be applied (see the next section).

3.2.8 The Plugin Portal

The Plugin Portal involves relevance evaluation. The concurrency of the relevance evaluation may be customized through a configuration option with direct benefits for portal throughput. Some notes on this option follow.

- The configuration option is:

  Windows (via the Windows registry):
  HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\BigFix\EnterpriseClient\Settings\Client_BESPluginPortal_Performance_ThreadLimit

  Linux (via client setting): BESPluginPortal_Performance_ThreadLimit
- The default threading value for relevance evaluation is automatically set to the number of cores available. The maximum allowed value is 128.
- Given the configuration is automatically managed for new deployments in BigFix 10.0.4, it is recommended legacy deployments remove this setting so it may be automatically managed.
- The general recommendation applies to monitor performance before and after a configuration change.

3.2.9 The MDM Docker Host

The following configuration recommendations have been used for stress testing of the Docker host instance.

- /etc/security/limits.conf
  - root soft nofile 100000
  - root hard nofile 100000
  - root soft nproc 100000
- /etc/sysctl.conf
  - net.core.rmem_default=1000000
  - net.core.wmem_default=1000000
  - net.core.rmem_max=1000000
  - net.core.wmem_max=1000000
  - net.ipv4.tcp_rmem=4096 87380 167177216
  - net.ipv4.tcp_wmem=4096 65536 167177216

3.3 MS SQL Management Recommendations

We will describe management options for MS SQL, including parallelism and query performance.

3.3.1 MS SQL Parallelism

A database management system is designed to drive high parallelism on modern day multiprocessor hardware. However, parallelism typically has a threshold where throughput levels off, and may even degrade. The default MS SQL configuration is typically enabled for maximum parallelism, which is not ideal. The BigFix 10.0 release does automatic management of the following parallelism parameters upon upgrade. Changes should only be made with expert insight and monitoring.

- Maximum Degree of Parallelism (MAXDOP).
  The Maximum Degree of Parallelism determines how many processors are utilized for a parallel plan. The default value is zero (0), which essentially means “unlimited”. The proper value needs to be derived from the specific physical hardware of the database server and is typically a low value in the interval of [4,8].
  - The specific hardware parameter is the number of NUMA cores. This may be derived by querying the MS SQL table sys.dm_os_memory_nodes. The returned value will be the number of nodes + 1.
  - Information on how to set the MAXDOP and an associated calculator are provided at this location: URL. The link is also provided in the References section.
- Cost Threshold for Parallelism (CTFP).
  The cost threshold for parallelism determines the level where SQL server will build and execute parallel plans for queries. The default value is five (5). The recommended value is fifty (50).

Further details on configuring these values are provided in the References section.

3.3.2 MS SQL Query Performance

MS SQL query performance can be inspected via the package cache. The package cache is the set of currently active compiled statements. It may be derived via the following query. Note the query “order by” clause may be modified to determine the sorting of the result set. The results are ideally reviewed by an expert, but even a lay person can derive conclusions based on query times, lock impact, etc.

```
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 4: MS SQL Query Performance Package Cache

3.3.3 MS SQL Always On Availability Groups for Read Scale

An MS SQL Always On Availability Group (hereafter referred to as an MS SQL AG) may be deployed for high availability of the BigFix databases. It may also be used for "read scale". For example, you may have the following deployment.

- The BigFix primary database BFEnterprise.
- A read only replica of the BFEnterprise database.
- An off-site disaster recovery replica of the BFEnterprise database.

What is read scale? Essentially the deployment of a read only replica for the secondary provides a horizontal scalability solution where "read only" applications may have a dedicated server semi-independent of the primary. A good example where this is valuable is as an Extract-Transform-Load (ETL) data source for BigFix Insights. Custom integrations may also benefit. As of this writing, it is not recommended for the BigFix Inventory (BFI), Security & Compliance Analytics (SCA), or WebUI applications; these applications should always connect to the primary database.

Deployment of the MS SQL AG is relatively transparent to the application. The MS SQL AG may generally be enabled for BigFix upgrade operations. Care should be taken to ensure the storage and query performance is maintained across the cluster. In addition, it is recommended to have dedicated network adapters between the nodes of the group. The MS SQL query performance approach in the previous section offers a simple way to monitor.

Information on configuring the listener for an MS SQL AG is available here: URL.

3.3.4 MS SQL Cloud Services

There are several cloud services options for MS SQL. For example, Google Cloud SQL for SQL Server. The official platform documentation provides information on the specific offerings supported.

In terms of performance management of the cloud services, they will perform acceptably if the following considerations are met.

- Network latency from the application to the cloud service must be less than 1ms. For example, an on-premises deployment of the application with a remote cloud service is generally not recommended due to latency concerns.
- Suitable vCPU, RAM, and storage allocation per the capacity planning recommendations in this guide.
- Storage class selection that meets the minimum requirement of 5K IOPS @ 1ms latency.

3.3.5 MS SQL Secure Settings

A set of MS SQL secure settings are available to provide additional security enforcement for database operations. The set of settings and their sample reference documentation follows.

- Force Encryption.
  The force encryption option for MS SQL ensures encrypted communication to the DBMS engine.
  Enablement is via MS SQL best practices: URL.
- Common Criteria.
  Common criteria compliance includes such dimensions as Residual Information Protection (RIP), login audit, and authorization changes.
  Enablement is via MS SQL best practices: URL.
- Service Protection.
  By default, the MS SQL instance is configured to support up to 32,767 connections. This has the potential to degrade the service to the point it may effectively yield a Denial of Service (DoS) attack. As a result, a practical limit of 1,000 is recommended.
  Enablement is via MS SQL best practices: URL.

- ODBC Strong Encryption.
  This option will enforce strong encryption for the ODBC connection. It is recommended if the database server is anti-collocated with the BigFix server(s).
  Enablement is via MS SQL best practices: URL.

These settings do have run time impact as more work needs to be done for database operations. Diligent monitoring before and after the configuration changes is recommended. Observation on the aggregate impact of the above changes for a reference deployment follow. The impact is considered manageable, but non-trivial.

- Root server impact of 20% additional processor time.
- FillDB service impact of 5% additional processor time, with 20% throughput reduction.
- GatherDB service impact of 100% additional processor time, though this is a short duration low cost service (e.g., single core utilization levels).

3.4 DB2 Management Recommendations

We will describe DB2 instance configuration and query performance.

3.4.1 DB2 Instance Configuration

The DB2 instance configuration applies to all databases running beneath it. The following configuration parameters are typically set for the instance by BigFix, and may not be present if the instance is a custom build, migrated, etc. It should be verified they are enabled and, if they are not, enable them.

```
db2set DB2_SKIPINSERTED=ON
db2set DB2_EVALUNCOMMITTED=YES
db2set DB2_EXTENDED_OPTIMIZATION=ON
db2set DB2_SKIPDELETED=ON
db2set DB2COMM=TCPIP
db2set DB2AUTOSTART=YES
```

Figure 5: DB2 Instance Configuration

3.4.2 DB2 Query Performance

A basic mechanism to manage DB2 query performance is via snapshots. The snapshot provides a point in time view of all transactions on the system. The following steps may be used to derive a snapshot. The results are ideally reviewed by an expert, but even a lay person can derive conclusions based on query times, lock impact, etc.

```
Step 1: db2 update dbm cfg using DFT_MON_BUFPOOL on DFT_MON_LOCK on DFT_MON_SORT
on DFT_MON_STMT on DFT_MON_TIMESTAMP on DFT_MON_TABLE on DFT_MON_UOW on
Step 2: Reset the monitor switches: db2 reset monitor all
Step 3: Run the workload.
Step 4: Get the dbm snapshot: db2 get snapshot for dbm > dbm.txt
Step 5: Get the database snapshots: db2 get snapshot for all on <db> >
<db>.txt  (substitute <db> and iterate over all databases).
Step 6: db2 update dbm cfg using DFT_MON_BUFPOOL off DFT_MON_LOCK off
DFT_MON_SORT off DFT_MON_STMT off DFT_MON_TIMESTAMP on DFT_MON_TABLE off
DFT_MON_UOW off
```

Figure 6: DB2 Query Performance Snapshot

3.5 High Availability and Disaster Recovery Recommendations

High Availability and Disaster Recovery (HADR) is concerned with system continuity in the event of failure. In general, High Availability involves failover within a geographically collocated data center. Disaster Recovery involves failover across data centers. The network latency for these scenarios can vary widely. For example, within a data center network latency is typically less than 1 (one) millisecond. The latency can be potentially hundreds of milliseconds across data centers.

Historically, BigFix has provided a proprietary replication solution known as DSA. DSA deployments are possible, but generally discouraged now. The reason is proprietary replication is considered expensive and less reliable than DBMS based replication. As a result, the HADR reference architecture going forward is based on a MS SQL Availability Group (described earlier in this guide), or alternatively a DB2 replication cluster. The components of such an architecture typically include:

1. The Primary.
   The primary database used for the BigFix deployment. It is typically configured to replicate synchronously to the secondary, and asynchronously to the disaster recover site if enabled.
2. The Secondary.
   The secondary database in the data center. Reads on the secondary may be leveraged for horizontal scale out, as described previously.
3. The Disaster Recovery instance.
   The optional Disaster Recovery instance. This would be used in the event of primary site failure.
4. The cluster manager.
   A third party construct to determine the health of the cluster, providing a voting algorithm and cluster integrity. Through the cluster manager, “split brain” scenarios are avoided.

How do you manage performance and capacity planning for an HADR based deployment? It is as follows.

1. Duplicate the capacity planning recommendations per SQL instance, whether two, or three, or more.
2. Ensure the data center latency is below one millisecond and configure the replication mode appropriately.
3. Involve your Database Administration and Cluster Management teams.

If more information is required on managing HADR based deployments, feel free to contact the author.

3.6 Operating System Management Recommendations

Both Linux and Windows are evolved and refined operating systems that can be used as the base for high performance systems. In terms of operating system tuning, very little tuning is typically required. However, specific guidelines for CPU, memory, IO, and network management apply.

- CPU and memory are very straight ahead, with capacity planning and monitoring approaches provided in this paper, along with recommendations for virtual deployments.
- IO and network management are more complex. Fortunately, there are some excellent BigFix Knowledge Base articles addressing this. Rather than duplicate this content here, please consult the References section.
- We will describe some special areas of operating system management: Windows port management, Linux “swappiness”, and the Linux “ulimit”.

3.6.1 Windows TCP/IP Port Management

- In the event that TCP/IP ports appear to be exhausted, configuration changes may improve the operational behavior. For example, if the BigFix logs contain a number of HTTP 28 timeout messages, it is possible that port exhaustion is the cause.
- The base configuration is to modify the Windows registry for reducing the “time wait” interval used to recycle the TCP/IP port resources. To be more specific, it can be useful to reduce the client TCP/IP socket connection timeout value from the default value of 240 seconds, to the minimum lower bound of 30 seconds.
- More information, including how to manage the specific registry update, is available in the following technical note: URL.

3.6.2 Linux Swappiness

- The Linux swappiness kernel parameter (vm.swappiness) is a value in the interval [0,100] that defines the relative weight of swapping out runtime memory, versus dropping pages from the system page cache. The default value is typically 60.
- The recommendations for setting this value are as follows.
  - For a dedicated database management server, the swappiness should be set to zero (0).
  - For a database management server collocated with the BigFix application server, the swappiness should be set to ten (10).
- Further details on managing DB2 performance are provided in the References section.

3.6.3 Linux ulimit Management

For Linux operating system defines a system ulimit for the maximum number of open files allowed for a process (i.e., the nofiles option when you run the command “ulimit –a”). For the DB2 instance, the value for this kernel limit should be either “unlimited” or “65536”.

3.6.4 Virtualization Management

In today’s modern enterprise, virtualization is seen as a powerful way to address the management of cost and scale. In general terms, performance management of physical servers tends to be simpler. Resources are isolated, there is no hypervisor involved, and the operating system view of performance is a direct indicator of system and application performance.

To simplify performance management and keep latency characteristics to a minimum, the first recommendation is always to deploy on physical hardware. However, it is still possible that a virtual deployment is still desired (whether enterprise standards, high skill levels in the team for virtual system performance, etc.). To manage BigFix in a virtual environment, precautions must be taken to ensure performance. We will describe some of the key management aspects. We will then reinforce the fact that monitoring and understanding is critical in a virtual world.

3.6.5 "Right Sizing" the CPU Allocation

- When deploying a physical server, additional CPU resources are generally seen as passive, or perhaps even beneficial. In a virtual environment, an oversized VM may actually degrade performance. The reason is in a shared deployment model, larger VMs require greater scheduling and orchestration effort. This may lead to scheduling delays or wait time. As a result, "right sizing" is critical.
- The capacity recommendations in this paper are the starting point, with monitoring being essential. Some classic elements for monitoring follow.
    - CPU ready.
      This is the percentage of time the VM is ready to be run but is waiting due to scheduler constraints.
    - CPU wait.
      The amount of time the CPU spends in wait state.
- A set of VMware performance troubleshooting guides are provided in the References section.

3.6.6 VMware Snapshot Management

- Snapshots are a powerful tool in virtual environments. In addition, many teams new to virtualization start to leverage snapshots as a backup approach. In the context of VMware, it cannot be emphasized enough that snapshots should not be used for a backup policy!
- To understand why, it is strongly recommended to read the VMware literature in the References section of this paper. Essentially, snapshots result in the chaining of images with degradation incurred as a result of managing the chains. It is not unusual to see degradation on the order of hundreds of percent. To further compound the issue, such performance issues are often difficult to diagnose as they are obscured by the hypervisor.

3.6.7 VMware Latency Management

- With VMware vSphere 5.5, it is possible to set the latency sensitivity of a virtual machine. This serves to reduce the impact of virtualization with improved application performance, at the expense of "dedicated" resources. Further information is available through the VMware content in the References section.

3.6.8 Virtual IO Management

- Out of all system resources, IO is typically the most difficult to manage. High performance IO subsystems are relatively expensive, and prone to failure if redundancy is not managed. In addition, many solutions that perform well in a physical environment (say in the range of 5,000 to 10,000 IOPS) may plummet in a virtual environment (to, say, 100 IOPS).
- As a result, in any virtual environment it is critical to benchmark and monitor the IO subsystem. Some recommendations for a proper IO benchmark follow.
    - The workload pattern (block size, read %, write %, randomness) is important. The table below shows some recommended workload profiles. All should be exercised to demonstrate the IOPS of the available system.
    - As BigFix drives multiple IO "workers" so should the benchmark. A minimum of an eight (8) core system with eight (8) workers is recommended. Note more workers can lead to degradation and determining the parallelism threshold where degradation occurs can be beneficial (and, obviously, the higher this threshold is the better).
    - A recommended tool for the benchmark is Iometer (see the References section).

Workload	Block Size	Read %	Random %
Workload 1: Stock	4KB	25%	0%
Workload 2: Open Action Profile	8KB	10%	20%
Workload 3: REST API Profile	8KB	90%	20%

Figure 7: IO Benchmark Workload Profiles

3.6.9 The Linux IO Scheduler

- Each Linux instance has an IO scheduler. The intent of the IO scheduler is to optimize IO performance, potentially by clustering or sequencing requests to reduce the physical impact of IO.
- In a virtual world, however, the operating system is typically disassociated from the physical world through the hypervisor. As a result, it is recommended to alter the IO scheduler algorithm so that it is more efficient in a virtual deployment, with scheduling delegated to the hypervisor.
- The default scheduling algorithm is typically “cfq” (completely fair queuing)[1]. Alternative and recommended algorithms are “noop” and “deadline”. The “noop” algorithm, as expected, does as little as possible with a first in, first out queue. The “deadline” algorithm is more advanced, with priority queues and age as a scheduling consideration. System specific benchmarks should be used to determine which algorithm is superior for a given workload. The general recommendation is to use the “deadline” scheduler.
- The following console output shows how to display and modify the IO scheduler algorithm for a set of block devices. In the example, the “noop” scheduler algorithm is set. Note to ensure the scheduler configuration persists, it should be enforced via the operating system configuration (e.g. /etc/rc.local).

```
[root@deehtdb0a2ccxra ~]# echo noop > /sys/block/sda/queue/scheduler
[root@deehtdb0a2ccxra ~]# echo noop > /sys/block/sdb/queue/scheduler
[root@deehtdb0a2ccxra ~]# echo noop > /sys/block/sdc/queue/scheduler
[root@deehtdb0a2ccxra ~]# cat /sys/block/sdc/queue/scheduler
[noop] anticipatory deadline cfq
[root@deehtdb0a2ccxra ~]# cat /sys/block/sda/queue/scheduler
[noop] anticipatory deadline cfq
[root@deehtdb0a2ccxra ~]# cat /sys/block/sdb/queue/scheduler
[noop] anticipatory deadline cfq
```

Figure 8: Modifying the Linux IO Scheduler

[1] With Red Hat Enterprise Linux 7, the default scheduler has been set to “deadline”.

References

BigFix Capacity Planning Guide

BigFix Maintenance Guide

BigFix Non-Functional Requirements: A Checklist Approach

MX Performance Toolkit for BigFix

BigFix Knowledge Center

BigFix Resource Center

BigFix System Requirements

Iometer

NVMe SSDs: Everything you need to know about this insanely fast storage

MS SQL Maximum Degree of Parallelism

MS SQL Cost Threshold for Parallelism

Performance Best Practices for VMware vSphere™ 5.5

Performance Best Practices for VMware vSphere™ 6.7

Performance Best Practices for VMware vSphere™ 7.0

Best practices for virtual machine snapshots in the VMware environment

VMware: Troubleshooting ESX/ESXi virtual machine performance issues

VMware: Troubleshooting virtual machine performance issues

VMware: Performance Blog

Notices

This information was developed for products and services offered in the U.S.A.

HCL may not offer the products, services, or features discussed in this document in other countries. Consult your local HCL representative for information on the products and services currently available in your area. Any reference to an HCL product, program, or service is not intended to state or imply that only that HCL product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any HCL intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-HCL product, program, or service.

HCL may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to HCL TECHNOLOGIES LIMITED email: products-info@hcl.com

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: HCL TECHNOLOGIES LIMITED PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. HCL may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-HCL Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this HCL product and use of those Web sites is at your own risk.

HCL may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact HCL TECHNOLOGIES LIMITED email: products-info@hcl.com

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by HCL under terms of the HCL License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-HCL products was obtained from the suppliers of those products, their published announcements or other publicly available sources. HCL has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-HCL products. Questions on the capabilities of non-HCL products should be addressed to the suppliers of those products.

All statements regarding HCL's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All HCL prices shown are HCL's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

HCL, and other HCL graphics, logos, and service names including "hcltech.com" are trademarks of HCL. Except as specifically permitted herein, these Trademarks may not be used without the prior written permission from HCL. All other trademarks not owned by HCL that appear on this website are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by HCL.

IBM and other IBM graphics, logos, products and services are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Oracle database, Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

VMware's and all VMWare trademarks and logos are trademarks or registered trademarks in the United States and certain other countries.

Dell, EMC, DellEMC and other trademarks are trademarks of Dell Inc. or its subsidiaries in the United States and certain other countries.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo and JBoss are registered trademarks of Red Hat, Inc. in the U.S. and other countries. Linux is a registered trademark of Linus Torvalds. All other trademarks are the property of their respective owners.

Mozilla and all Mozilla trademarks and logos are trademarks or registered trademarks in the United States and certain other countries.

Google LLC All rights reserved. Google and the Google Logo are registered trademarks of Google LLC.

NETAPP, the NETAPP logo, and the marks listed at www.netapp.com/TM are trademarks of NetApp, Inc.