# BigFix Capacity Planning: Core Platform

*an HCL Product*

# HCLSoftware

**Document version 11.x.9**

*Mark Leitch*
*BigFix Platform, Performance & Secure Engineering*

# CONTENTS

# LIST OF FIGURES

# REVISION HISTORY

| Date | Version | Revised By | Comments |
|---|---|---|---|
| March 31st, 2020 | 10.x.1 | MDL | Initial distribution based on 9.x.18. |
| June 2nd. 2020 | 10.x.2 | MDL | Revised MS SQL DoP links. |
| July 1st, 2020 | 10.x.3 | MDL | MCM updates. |
| July 9th, 2020 | 10.x.3a | MDL | Redacted content pending code. |
| September 16th, 2020 | 10.x.4 | MDL | Revised Insights content. |
| July 9th, 2021 | 10.x.5 | MDL | Updates for BigFix 10.0.4 |
| July 16th, 2021 | 10.x.6 | MDL | Updates for Insights & IVR. |
| September 27th, 2021 | 10.x.7 | MDL | Revised 300k capacity outlook. |
| October 4th, 2021 | 10.x.8 | MDL | Added SQL monitor approaches. |
| October 18th, 2021 | 10.x.9 | MDL | Added MS SQL AG statements. |
| November 28th, 2021 | 10.x.10 | MDL | Added MS SQL secure settings. |
| April 14th, 2022 | 10.x.11 | MDL | Added BigFix 10.0.6 updates. |
| December 5th, 2022 | 10.x.12 | MDL | Added BigFix 10.0.8 updates. |
| October 1st, 2023 | 11.x.1 | MDL | BigFix 11 revision. |
| November 26th, 2023 | 11.x.2 | MDL | Added MS SQL AG information. |
| March 25th, 2024 | 11.x.3 | MDL | BES Explorer capacity planning. |
| June 1st, 2024 | 11.x.4 | MDL | MS SQL Cloud Services addition. |
| July 11th, 2024 | 11.x.5 | MDL | Remove MS SQL AG constraint. |
| March 31st, 2025 | 11.x.6 | MDL | BigFix 11.0.4 update. |
| April 22nd, 2025 | 11.x.7 | MDL | Added Remote Control & Server Automation. |
| June 15th, 2025 | 11.x.8 | MDL | Added MS SQL TDE reference. |
| November 10th, 2025 | 11.x.9 | MDL | Added BigFix OSD reference. |

*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.

This document will provide an overview of capacity planning for the BigFix Version 11.x solution. In addition, it will offer management best practices to achieve a well performing installation that demonstrates service stability. This will include the deployment of the BigFix management servers into cloud, or virtual, environments. Capacity planning for virtual environments typically involves the specification of sufficient physical resources to provide the illusion of unconstrained resources in an environment that may be characterized by highly variable demand

In this document we will provide a BigFix overview, including functionality and architecture. We will then offer the capacity planning recommendations, built on a set of base definitions. Finally, configuration recommendations will be provided for achieving a highly functional and performant installation.

**Note**: This document supersedes all previous BigFix capacity planning documentation. In addition, while the previous capacity planning reference from the author spanned several topics with a prose-based approach, this document is more specific with a bullet point approach. The rationale for this is to have smaller, more easily digestible documents with easily understood recommendations.

As a result, 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 Capacity Planning (this document).

- BigFix Non-Functional Requirements: A Checklist Approach
  A set of guidelines and recommendations for BigFix performance and security.

- BigFix Maintenance Guide.
  A set of configuration and maintenance recommendations for BigFix.

- MX Performance Toolkit for BigFix.
  A set of tools and performance management approaches for BigFix.

**Note**: This document is considered a work in progress. Capacity planning recommendations will be refined and updated as new BigFix releases are available. While the paper in general is considered suitable for all BigFix Version 9.5.x and 10.x releases, with new content as noted, it is best oriented towards BigFix Version 11.0.0 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.
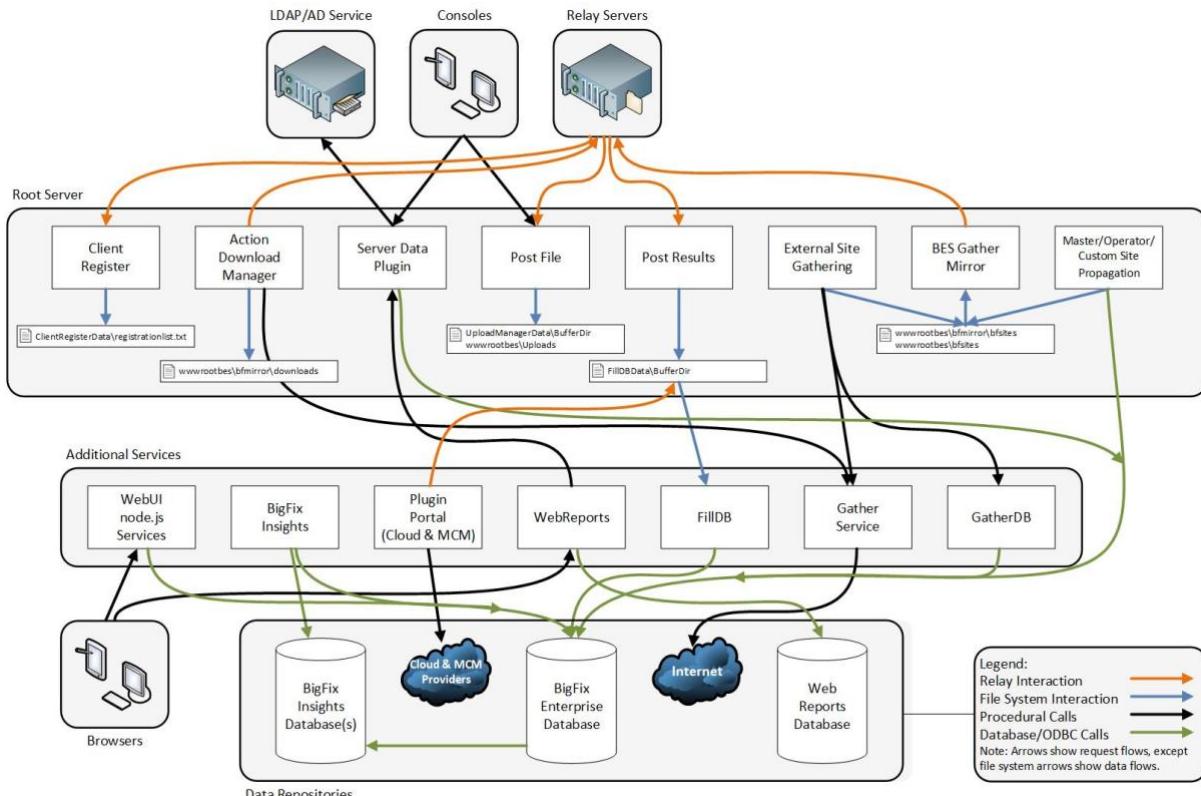


Figure 2: BigFix Architecture

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. The MCM capacity planning

section below offers further details on the MCM deployment topology.

- 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.

- 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  Capacity Planning

The capacity planning recommendations will be broken down into the following topic.

1. Units of capacity planning.
2. Capacity planning tools.
3. Capacity planning.

## 3.1  Units of Capacity Planning

To properly manage capacity, some base definitions must be understood.  Adhering to these definitions is critical to the accuracy of the results.  We will break the units down into several physical, virtual, and organizational components.

### 3.1.1  Cores, CPUs, and vCPUs

Today's multiprocessor architectures are often defined in terms of cores and may be physical cores or virtual CPUs (aka vCPUs).  The following rules of thumb apply for our capacity planning outlook.

- In terms of pure clock speed, per typical cloud definitions, we will generally consider a CPU core to be a relatively current generation 2.0+ GHz/core implementation.

- A discrete core and a vCPU can be considered equivalent.  However, more management is typically required for a vCPU due to the hypervisor scheduler and contention for vCPUs due to the virtual configuration and over commitment policy.  As a result, hypervisor monitoring is required to ensure a vCPU is operating in the range of 90+% of the capability of a physical core.

- A hyper threaded core does not have the throughput capability of a "pure" core and can be considered to have on the order of 30% of the capability of a pure core.

- Our sizing approach is based on pure, physical cores at 2.0+ GHz/core.  For virtual or hyper threaded cores, the above efficiency ratios should be part of the sizing methodology.

### 3.1.2  Memory

- Modern systems are extremely good at exploiting memory.  If you are able to exceed the memory requirements, the operating system will use the additional memory to improve caching and throughput.

- On modern systems, it is not unusual to see high memory utilization approaching 100% of physical memory.  This is often quite healthy, as long as the system is not paging significantly.

- Database servers in particular like to exploit memory, and if extra memory is available, the first priority should be to allocate it to the database server.

### 3.1.3  Network

- High bandwidth and low latency memory is readily available and should be used wherever possible in the data center.  For example, 10 gigabit ethernet and teamed adapters are common and will offer excellent network capability.

- Beyond the data center, mapping the network for latency, bandwidth, etc. is critical for relay and endpoint management.

### 3.1.4  Storage

- The BigFix storage requirement is for storage to offer in excess of 5,000 IOPS (IO Operations

per Second) with 1ms latency.  This is especially critical for the database server volumes.

- This capability is easily achieved with local SSD with an AHCI interface.

- This capability can be massively over-achieved with flash-based storage appliance and local NVME based flash storage.  For example, on benchmark systems we can manage in excess of 100,000 IOPS with 1ms latency with a single NVME flash-based drive.  See the References section for further information on NVME and the "storage revolution".

### 3.1.5  Cloud vs SotP vs SotA

We will provide a few definitions for server deployments.

- Cloud:  A public or private instance with:
CPU: 2.0 Ghz or better
IO: 5,000 IOPS with 1ms latency or better.

- SotP: A "State of the Practice" environment.  This is an affordable present-day server with:
CPU: 3.0 Ghz or better.
IO: 100,000 IOPS with 1ms latency or better.

- SotA: A "State of the Art" deployment.  The definition is in continuous upheaval due to the array of continuous improvements for server deployments.

In general, SotP deployments are recommended for their combination of affordability, economy, and performance.

### 3.1.6  Concurrent Users

For the user interface component, the number of concurrent users that need to be supported is important.  It is also highly misunderstood.  The following definition for concurrent users should be adhered to for the WebUI instance.

- For the concurrent user population, consider:
P = total population for an instance (including administrators, end users, etc.).
C = the concurrent user population for an instance of BigFix.

- Concurrent users (C) are the set of users within the overall population (P) that are actively managing the environment at a point in time (e.g., administrator operations in the user interface, endpoint operations, etc.).

- In general, P is a much larger value than C (i.e., P >> C).  For example, it is not unrealistic that a total population of 200 users may have a concurrent user population of 20 users (i.e., 10%).

- The number of concurrent users supported by a solution is a function of request response times under load.  Numbers are driven by lab and field-based performance evaluations that consider the user experience (response times), server load, and server impact.

- Lab based simulations use aggressive think times.  For example, user response times in the interval [1s,10s] are used.  As a result, when mapping an organization's user population to the number of concurrent users, the standard is for actively engaged users driving high request rates.

## 3.2  Capacity Planning Tools

Several complex capacity planning tables will be provided.  These tables constitute the reference content for BigFix.  However, a set of performance and capacity planning tools are provided to greatly simplify the capacity planning experience.  See the MX Performance Toolkit in the References section for obtaining the distribution and associated documentation.

The following figure shows an example of invoking the capacity planning tool.  The options are intended to be self-explanatory, with full descriptions provided in the reference document.

```
MXCapacity –-endpoints 60000 –-concusers 10 –-service root dbms webui webreports relays
```

*Figure 3: Capacity Planning Tool Invocation Example*

## 3.3  Capacity Planning

The capacity planning recommendations will be broken down across the following components.

- The root server, including the database server.

- The WebUI server.

- The console.

- The relays and endpoint infrastructure.

- The Plugin Portal.

- Modern Client Management (MCM).

- Remote Control (RC).

- Operating System Deployment (OSD).

- Server Automation (SA).

- Message level encryption.

- Note: Prior versions of this guide provided capacity planning recommendations for BigFix Insights.  The recommendations are now part of the core product documentation: URL.

Some considerations should be kept in mind for the recommendations.

- Capacity planning recommendations are general purpose for a "typical" workload.  We model the "typical" workload in our performance labs and consolidate field results.

- Capacity planning recommendations are provided in terms of the number of managed endpoints.  This is a simplification to make the recommendations consumable, but there are many more dimensions that may apply for a specific installation.

- In the event you are within a range of capacity planning recommendations (e.g., somewhere between 50,000 and 100,000 managed endpoints for the BigFix root server), you may start at the low end of the range and grow, assuming your workload and system behavior is well understood.  This applies to the CPU, memory, and storage allocations.  The IO subsystem and network requirements are universal.

- Monitoring over time is always recommended.  This guide includes specific references for how to monitor at the operating system, application, and database levels.

- Capacity planning is seldom static.  Systems grow over time.  Entropy increases. Maintenance operations are typically required, especially at the database level, to manage this and ensure performance stability.  References for this are provided in this guide.

### 3.3.1  The Evolution of BigFix Capacity Planning

Historically, BigFix has supported a 250k agent deployment on a single root server.  With BigFix 10.0.0, this was extended to 250k agents and 50k objects managed via the Plugin Portal.

With BigFix 10.0.4 through BigFix 11, we now support 300k for any set of devices, whether agent based or managed via the Plugin Portal.  For example, the following deployments are all supported on a single root server.

- 300k agents.

- 200k agents with 100k MCM objects.
- 300k MCM objects (with no agents).
- 100k agents distributed amongst 100k cloud objects and 100k MCM objects.

The general support statement is 300k devices in the BFEnterrprise "Computers" table, that may or may not be correlated devices.

## 3.4  The Root Server

The following table describes the root server's (aka the management server) capacity planning recommendations as a function of the scale of the "managed to" estate.

- The recommendations are for a collocated server configuration.  Finer grained, non-collocated recommendations are available via the referenced capacity planning tools.
- Requirements for the root server include the base operating system requirements.  All other capacity planning numbers are in addition to the base operating system requirements.
- Storage requirements:
    o It is advantageous to have distinct storage subsystems/controllers for specific components.  For example, a recommended best practice is for discrete storage subsystems for each of the base system, the database logs, and the database containers.
    o The database logs are typically characterized by the need for very fast sequential IO.
    o The database containers can have much more diverse access patterns and are highly insulated from storage impact by virtue of the database buffer pools.
    o A reasonable storage breakdown is 30% for the base, 5% for the database logs, and the remainder (65%) for the database containers.
    o A good rule of thumb is the core BigFix Enterprise database grows by 5GB for every 10,000 computers added.  The overall storage impact will be higher than this (logs, buffers, temps, backups, etc.).
    o Suitable storage is recommended to accommodate the growing database size and associated management overhead (e.g., a working set of database backups).  The storage requirements reflect the requirements for BigFix 9.5 onwards, where the Unicode base has driven greater storage requirements.
    o The BigFix application directory also contains a download cache which defaults to 1GB.  We recommend increasing this cache size to somewhere between 100GB and 1TB (or higher) depending on needs.  The storage requirements below should be increased to account for the desired cache size.
- Network requirements:  A 1 Gbps network or better for the management server infrastructure.

| Agent Deployment Size | CPU | Memory (GB) | Storage (GB) |
|:---:|:---:|:---:|:---:|
| < 1,000 | 4[1] | 16 | 100 |
| 10,000 | 4 | 16 | 250 |
| 50,000 | 6 | 24 | 300 |
| 100,000 | 12 | 48 | 500 |
| 150,000 | 16 | 72 | 750 |
| 200,000 | 20 | 96 | 1,000 |
| 250,000 | 24 | 120 | 1,250 |
| 300,000 | 28 | 144 | 1,500 |

*Figure 4: Root Server Capacity Planning Requirements*

## 3.5  The Web Reports and BES Explorer Services

The Web Reports service is assumed to be collocated with the root server instance.  The BES Explorer service, introduced in BigFix 11.0.2, must always be anti-collocated with the root server instance.  Their capacity planning requirements are similar given the Explorer is an advanced form of Web Reports, and there are options for anti-collocation for Web Reports as well.  Both services may be deployed in a horizontal cluster configuration, where multiple service instances are utilized.

For both services, the following capacity planning recommendations may be used when anti-collocated with the root server.  The "plus sizing" indicates these resources are in addition to the base deployment node requirements (e.g., OS and other applications).  The MAX function means take the maximum of the two values in parentheses.  Note the additional storage requirements are only for Web Reports as BES Explorer does not have a dedicated database or cache.  As always, monitoring is required as utilization will be highly dependent on reporting and application workloads, as well as request parallelism.

| Component | Additional CPU | Additional Memory (GB) | Additional Storage (GB) (Web Reports Only) |
|:---:|:---:|:---:|:---:|
| Web Reports/ BES Explorer | MAX(2, 10% of Root Server CPU) | MAX(4, 10% of Root Server Memory) | MAX(10, 5% of Root Server Storage) |

*Figure 5: Web Reports and BES Explorer Capacity Planning Requirements*

## 3.6  The WebUI Server

The BigFix WebUI offers a scalable and highly responsive management interface.  There have been several iterations of the WebUI server.  This document will describe the results for the BigFix March 2020 update (typically 10.0 based) based WebUI server.  If you are on a prior version of the server, an upgrade is strongly recommended as significant improvements have been delivered providing improved scale, function, and user experience.

---

[1] While more granular sizing is possible here, to ensure base operating system health and given the commodity level of a system with four processor cores, this is considered a reasonable and minimal base.

- The recommendations are in addition to the root server requirements.

- If an anti-collocated instance is deployed (meaning an instance not deployed on the root server), the CPU requirements should be split across the database and WebUI servers, and the storage should be added to the database server.

- In terms of recommended scalability limits, both the Windows and Linux WebUI instances support 36 concurrent users on a 300k device deployment base. Once again, these are highly active concurrent users per the previously provided capacity planning definitions.

- Concurrent users would typically be non-master operators, managing a subset of the estate.

- It is possible to manage at a larger scale based on user operations, infrastructure capability, etc. However, the stated bounds should be considered a good "rule of thumb" for the scale of the solution.

| Component | Additional CPU | Additional Memory (GB) | Additional Storage (GB) |
|---|---|---|---|
| BigFix WebUI | +2 per 10 concurrent users | +2 per 10 concurrent users | 15% of BigFix database |

*Figure 6: WebUI Capacity Planning Requirements*

## 3.7  The Console

The following table shows capacity planning requirements for a workstation-based console.

- Expectation is data center level network speeds are available.

- The site cache is expected to be on the order of 20MB per external site.

- The question often arises how many console operators may be supported. A primary selling feature is the ability of a small number of operators to manage a large estate. However, if fine grained management is required, a base of 300 operators may be managed with careful attention to the console infrastructure. Proceeding beyond this value would require understanding of the infrastructure and associated workload impact.

| Deployment Size | CPU | Memory (GB) | Storage (GB) |
|---|---|---|---|
| < 10,000 | 1 | 2 | 0.25 + site cache |
| < 100,000 | 2 | 4 | 2GB + site cache |
| > 100,000 | 2 | 6 | 4GB + site cache |
| > 200,000 | 2 | 8 | 8GB + site cache |

*Figure 7: Console Capacity Planning Requirements*

The following table shows capacity planning requirements for a terminal or Citrix server-based implementation.

- The expectation is data center level network speeds are available for the server, and each server may be managing on the order of 10-20 concurrent users (remote users, meaning they may not reside in the data center).

- In the event a greater number of concurrent users are in effect, the general rule of thumb is to add on the order of 1 CPU and 2-6 GB of RAM for every additional concurrent user (RAM is dependent on the deployment size).

- As always, requirements are workload dependent so monitoring of the system under load is always recommended. Ranges are given where appropriate with the expectation that monitoring may be used to fine tune in the customer environment.

| Deployment Size | CPU | Memory (GB) | Storage (GB) |
|:---:|:---:|:---:|:---:|
| < 10,000 | 8 - 16 | 8 - 16 | 20 |
| < 100,000 | 10 - 20 | 16 - 48 | 80GB |
| > 100,000 | 10 - 20 | 32 - 80 | 80GB - 240GB |

*Figure 8: Terminal Server Capacity Planning Requirements*

## 3.8 The Relays and Endpoint Infrastructure

The following diagram provides a deployment scenario for the BigFix relay and agent infrastructure[2].
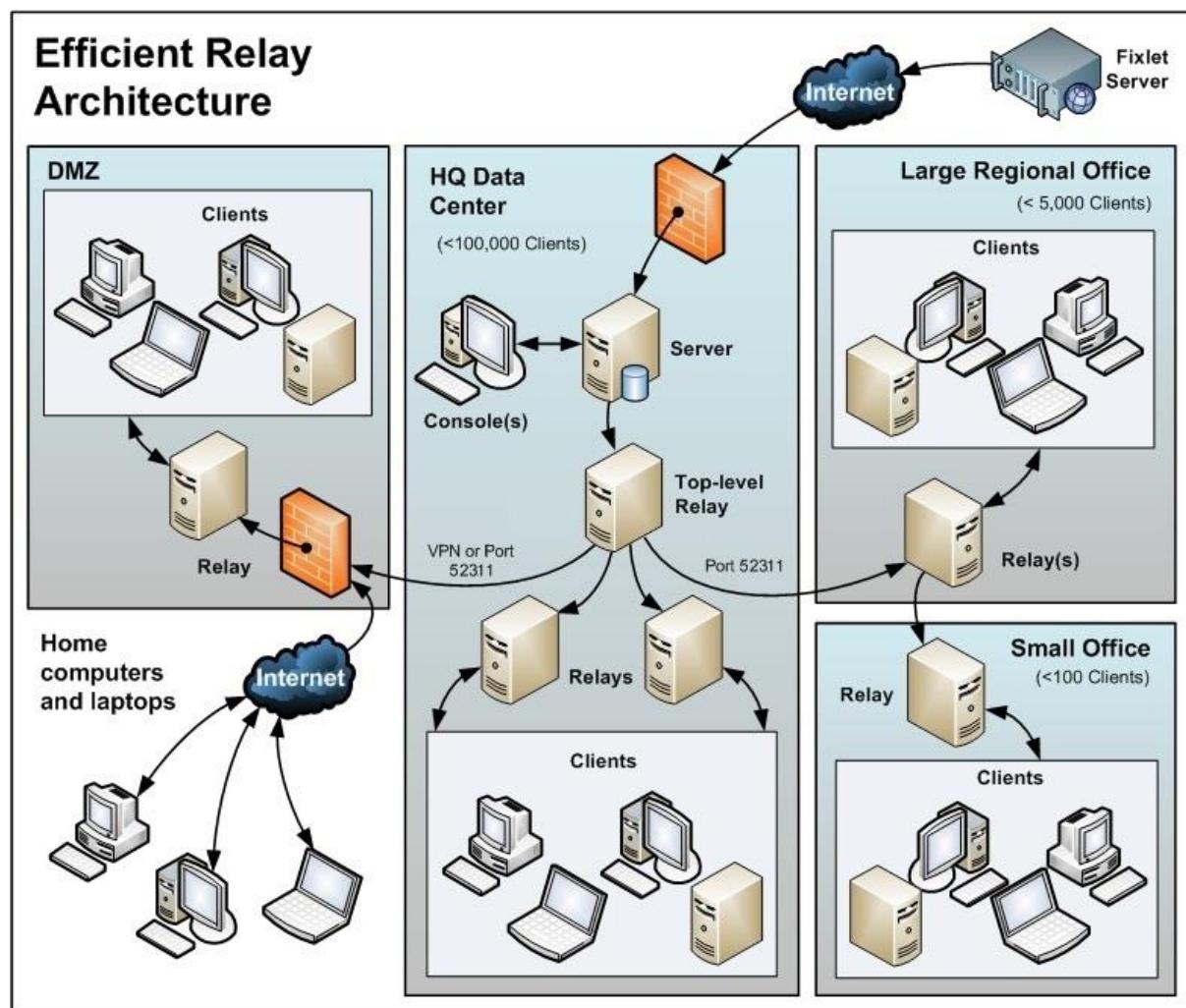


*Figure 9: BigFix Relay Infrastructure*

The relays are typically deployed in a hierarchy with top level relays serving other relays, and the leaf node relays (often referred to as site level relays) serving the endpoints.   In terms of capacity planning, we will describe the infrastructure requirements, and then break down the discussion in terms of top level relays and leaf node relays.

---

[2] Diagram courtesy of the BigFix Knowledge Center: URL.

### 3.8.1 Relay Infrastructure Requirements

We may break down the infrastructure requirements into CPU, memory, network, and storage.

### 3.8.1.1 CPU Requirements

- For the CPU specification, we will continue with the standard vCPU configuration, and provide a pair of [minimum,recommended] vCPU core values.

- BigFix can run successfully with the minimum configuration. However, in the event additional resources are available, or deploying into a virtual environment where over commitment is possible, and resources will be dynamically utilized, the recommended values will permit greater concurrency and reduce processor latency.

### 3.8.1.2 Memory Requirements

- Similarly the memory requirements consist of a [minimum,recommended] pair. The minimum values may be deployed successfully, but the recommended values will permit improved operating system memory management and buffering.

### 3.8.1.3 Storage Requirements

- Storage requirements are subject to configuration of the relay. The following calculation is representative of general relay storage requirements.

    Storage in GB = OS + 3GB (BigFix binaries + logs) +
        2GB (Default Cache) + Extended Cache (if additional cache space is configured) +
        300 MB/site (site cache for relay)

- For the relay storage requirements, the operating system storage may vary widely. For example, Windows 10 64 bit requires 20GB. Something like the Linux Tiny Core, where relays have been successfully demonstrated, requires on the order of megabytes. The net is we have high performance Windows relays running with 25GB, and they have been proven to virtualize extremely well.

- For Linux relays it can be important to manage the available file system handles, especially for large deployments. For example, the ulimit "nofiles" configuration parameter should be set to a minimum of 16384. See section 4.5.3 for more information on managing ulimit.

### 3.8.1.4 Network Requirements

- Network requirements depend heavily on the deployment, including available bandwidth and the possibility for dynamic network shaping.

- From a relay perspective, to push a 10MB download to 5,000 endpoints can result in on the order of 50GB of network traffic depending on the relay topology. For such a distribution, it is important to determine if the relay throttling configuration needs to be managed, or the download controlled and distributed over a specific timeline.

- In terms of base relay health, obviously the more available network bandwidth the better, but it is possible to deploy BigFix relays in highly constrained environments (e.g., < 10 Mbps) and manage successfully.

- A more difficult decision to reach is actual placement of the relays, and in particular the hierarchy of nodes. Decision points include network bandwidth, network latency, firewalls, server infrastructure, etc. Network topology maps typically exist for most enterprises. However, these maps rarely contain accurate metrics for network performance between nodes. Even when metrics are provided, they are often out of date or represent ideal conditions. In addition, network shaping often applies, meaning network characteristics may be dynamic based upon load.

- To facilitate understanding of the network and placement considerations, basic network ping

tests may be performed.  Sample ping commands follow.
Basic ping: ping –c 10 <ip>
Flood ping: ping –f –c 100000 –s 1500 –l 4 <ip>

- o Using these commands, a map may be built showing latency and packet loss.

- o Additional tests to demonstrate the number of hops via trace route (e.g., tracert) or equivalent are recommended.  Secure copy (e.g., scp) tests for sample payloads are also helpful.  In essence, once the network characteristics are defined, placement decisions typically become very straightforward.

- In lab tests, a general rule of thumb for a leaf relay is for every 1,000 endpoints served, bandwidth requirements of 100Mbps are typical.

  - o This implies for 10,000 endpoints served by a leaf relay, 1Gbps or greater bandwidth is required.

  - o Bandwidth is highly dependent on configuration and workload.  Monitoring is recommended.

- See section 4.2.2 for further information on relay configuration management.

### 3.8.2  Top Level Relays

For the relays serving other relays (also known as top level relays), the following capacity planning recommendations apply.

- Top level relays are generally recommended once you approach 40,000 endpoints or over 100 relays (whichever comes first).

- A top level relay should manage no more than 40,000 endpoints or 120 relays (whichever comes first).

- The relays managing the endpoints offer low utilization and may possibly be collocated with server nodes already distributed in the enterprise.

| Deployment Size (Endpoints Served) | CPU | Memory (GB) | Storage & Network |
|---|---|---|---|
| 10,000 | [1,2] | [2,4] | Per the infrastructure recommendations section. |
| 20,000 | [2,4] | [2,4] | |
| 40,000 | [2,4] | [4,8] | |

*Figure 10: Top Level Relay Capacity Planning Requirements*

### 3.8.3  Leaf Node Relays

For the relays managing BigFix agents directly (aka the leaf relays), we will offer a range of configuration choices.

1. A "stock" relay capable of managing 1000 agents/endpoints (a 1:1,000 ratio).  These may be authenticating or non-authenticating relays.

2. For Windows and Linux relays, the option for a "high scale" relay capable of managing 5,000 agents/endpoints (a 1:5,000 ratio).  At higher scale, usage of authenticating relays should be monitored to ensure infrastructure stability.

3. For Windows and Linux non-authenticating relays running version 11.0.4 or later, up to 10,000 agents/endpoints may be managed (a 1:10,000 ratio).  Future work will be enabling authenticating relay support at this scale.

In the event you would like a relay to serve somewhere between the specified ranges, you may use intermediate values in the ranges.

| Deployment Size (Endpoints Served) | CPU | Memory (GB) | Storage & Network |
|---|---|---|---|
| 1,000 | [1,2] | [2,4] | Per the infrastructure recommendations section. |
| 5,000 | [2,4] | [4,8] | |
| 10,000 | [4,6] | [6,12] | |

*Figure 11: Leaf Node Relay Capacity Planning Requirements*

### 3.8.4 Persistent Connection Support

Persistent connections are enabled between leaf relays and clients in the event that UDP notifications are not available.  The maximum number of persistent connections has been elevated in BigFix 11.0.4.

| BigFix Version | _BESRelay_PersistentConnection_MaxNumber |
|---|---|
| BigFix 11.0.3 & Below | 100 |
| BigFix 11.0.4 & Above | 1,000 |

*Figure 12: Persistent Connection Limits*

### 3.8.5 The Archive & Upload Manager

The Archive & Upload Manager is a service that permits the client to upload files.  In BigFix 11.0.4 an optimization may be enabled that avoids landing the file payload at each step in the relay chain.  This optimization is not enabled by default. Internal benefits show a 600,000% improvement, with fault tolerance added to the protocol to achieve much higher reliability.  This also benefits applications such as BigFix Inventory.

| BigFix Version | <magical setting parameter> |
|---|---|
| BigFix 11.0.3 & Below | n/a |
| BigFix 11.0.4 & Above | Enable |

*Figure 13: Archive & Upload Manager Configuration*

### 3.8.6 Relay Virtualization

The BigFix relays virtualize extremely well.  Their small system requirements and lightweight workloads are ideal for virtual machine deployments.  In addition, it is possible to collocate several virtual relay instances in order to achieve virtual clusters for system throughput and redundancy.

- The following diagram shows the performance of an increasing number of relay requests, across several physical and virtual Linux instances.

- Performance across the instances is extremely consistent, providing excellent support for virtualizing relays.

- It should be pointed out that at the 50,000 request level, results are more inconsistent as we are at the infrastructure threshold for both virtual and physical deployments.  For tuning virtual instances, the virtualization section later in this paper should be followed.
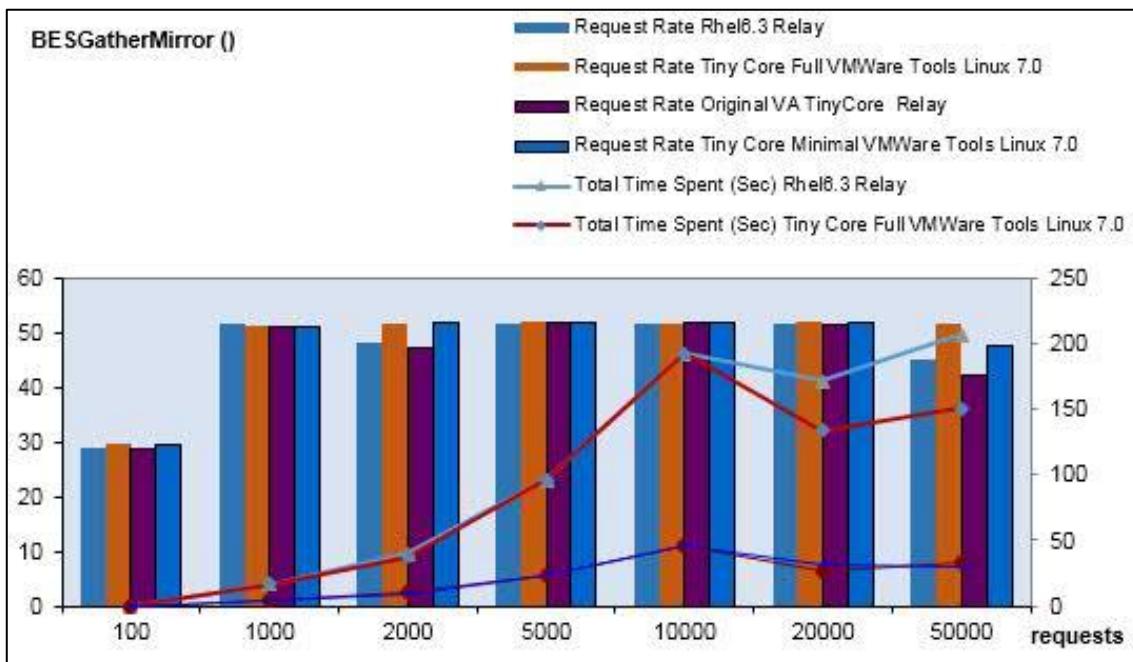
*Figure 14: BigFix Relay Virtualization Performance*

## 3.9  The Plugin Portal

The Plugin Portal offers a shift in BigFix capacity planning requirements.

- BigFix capacity planning has typically been in terms of the number of agents deployed.  With the Plugin Portal and the associated Modern Client Management (MCM) and Cloud capability, there is now an agentless based scale component.

- In BigFix 10.0.4 and beyond, each Plugin Portal instance may manage up to 75,000 compute devices.  For example, a Portal deployment for the MCM feature may manage 15k MacOS devices, 20k Windows devices, and 40k cloud devices for an aggregate 75k devices. Note prior to this, each Plugin Portal instance could support up to 10,000 compute devices.

- There may be several Plugin Portal instances, limited to an aggregate of 300k compute devices.  For example, you may want to deploy multiple plugin portals geographically collocated with the devices they manage.

  o  For the clustered Plugin Portal deployment, each portal instance should manage a distinct set of devices (i.e., disjoint sets of cloud instances, MDM servers, etc.).  If there is overlap, there is the potential for duplicate devices.  To ensure the clarity and integrity of operations, it is not supported to have portals overlap device sets.

  o  For a full 300k deployment of portal managed devices, a minimum of four (4) Plugin Portals should be deployed.

- The Plugin Portal may be deployed on a physical or virtual server.  For virtual deployments, care around hypervisor efficiency applies.

- The capacity planning requirements are simplified.  Capacity requirements are a function of the number of plugins and endpoints, as well as Fixlet content.

  o  The capacity requirements are on top of the base operating system requirements for the compute instance.  As always, upon deployment, runtime monitoring is recommended.

- The Plugin Portal may include two database instances: SQLite and MongoDB.  While these are relatively small and highly specific database servers, it should be ensured that the server has reasonable IO capability.  The recommendation is for the Portal to support a minimum IO

throughput of 1,000 IOPS with < 1ms latency.

- In terms of deployment topologies for the Plugin Portal, there are many options.

  o A Portal instance may be deployed geographically remote from the BigFix root server. This may be desirable if there is a large, remote cloud instance that the portal is managing.

  o The Portal may generate large volumes of reports. As such, the higher-level relay infrastructure (if any) should offer high bandwidth. For example, having the Plugin Portal report to a top level relay, false root relay, or the root server itself is recommended.

The following capacity planning requirements are based on the 10.0.8 and above versions. Significant performance enhancements were delivered in the 10.0.8 code stream. Note CPU requirements vary by class of Plugin (Cloud vs MCM). If both classes are active (i.e., Cloud and MCM plugins are enabled for a single portal instance), then the CPU requirements should be added.

| Deployment Size | CPU | Memory (GB) | Storage (GB) |
|---|---|---|---|
| 25,000 | Cloud: +1 / Plugin MCM: +4 | +4 per Plugin | +20 |
| 50,000 | Cloud: +1 / Plugin MCM: +8 | +6 per Plugin | +40 |
| 75,000 | Cloud: +1 / Plugin MCM: +12 | +8 per Plugin | +60 |

*Figure 15: Plugin Portal Capacity Planning Requirements*

## 3.10 BigFix Modern Client Management (MCM)

The Modern Client Management (MCM) offering provides device management capabilities for Android, Windows, iPhone, iPad, and MacOS devices. The high-level features of this offering include the following.

- Over-the-air enrollment.
- Bulk enrollment.
- Auto-pilot enrollment.
- Policy enforcement.
- Deep visibility.
- Automated actions.

In architectural terms, the MCM offering consists of three main components.

1. The MCM management interface via the WebUI.
   Recommendations are in line with the WebUI capacity planning recommendations provided in a previous section.

2. The Plugin Portal.
   This is shared with the cloud offering, and capacity planning recommendations have been provided in the previous section.

3. The Mobile Device Management (MDM) server host machine.
   This is a Docker-ized entity. While fine grained configuration details are possible, we will provide an aggregate sizing for the Docker host itself (i.e., encompassing all containers).

The following figure shows the MDM server docker deployment.
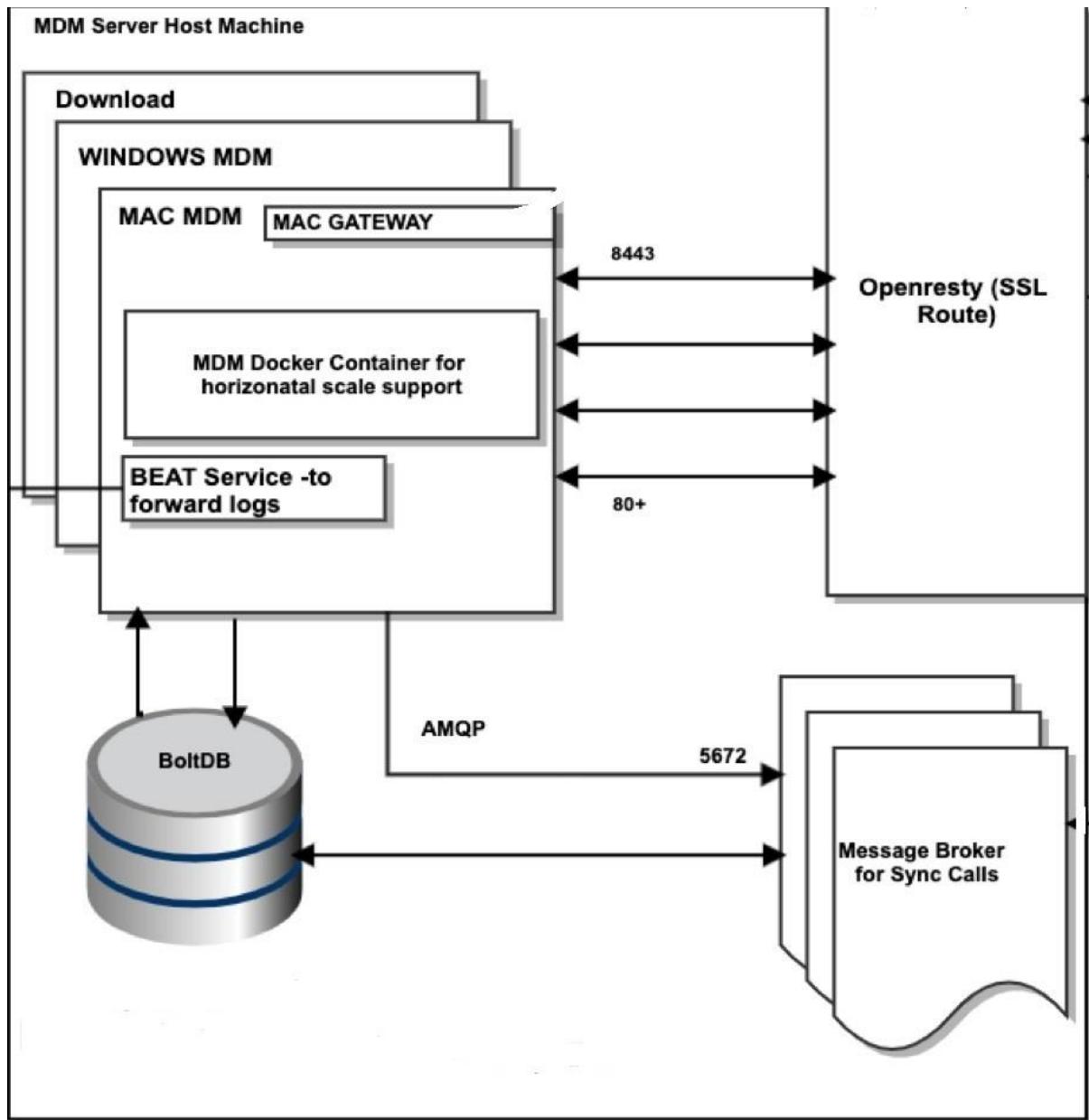
*Figure 16: MDM Server Docker Host*

Some notes on the docker host deployment.

- A range is provided for the CPU and memory requirements. In general, the top end of the range is recommended. However, "right sizing" based on workload is recommended.

- The MDM server has modest IO requirements. The recommendation is for the host to support a minimum IO throughput of 1,000 IOPS with < 1ms latency.

- Additional configuration recommendations are provided in section 4.2.8.

| Component | CPU | Memory (GB) | Storage (GB) |
|---|---|---|---|
| MDM Server Docker Host | [4,8] | [8,16] | [32,64] |

*Figure 17: MDM Server Docker Host Capacity Planning Requirements*

## 3.11 BigFix Remote Control (RC)

Remote Control (RC) offers several deployment options with the following components.

- The RC Server.
  The overall orchestrator.

- The RC Controller.
  Forms a peer-to-peer session with the target.

- The RC Target.
  Forms a peer-to-peer session with the controller.

- The RC Gateway.
  Responsible for providing network management between the Controller and Target.

- The RC Broker.
  Permits operation over the Internet by bridging the Controller and Target communication.

The Gateway and Broker are only required if specific network topologies are in place (e.g., Internet based communication).  See the following figure for a sample deployment architecture.
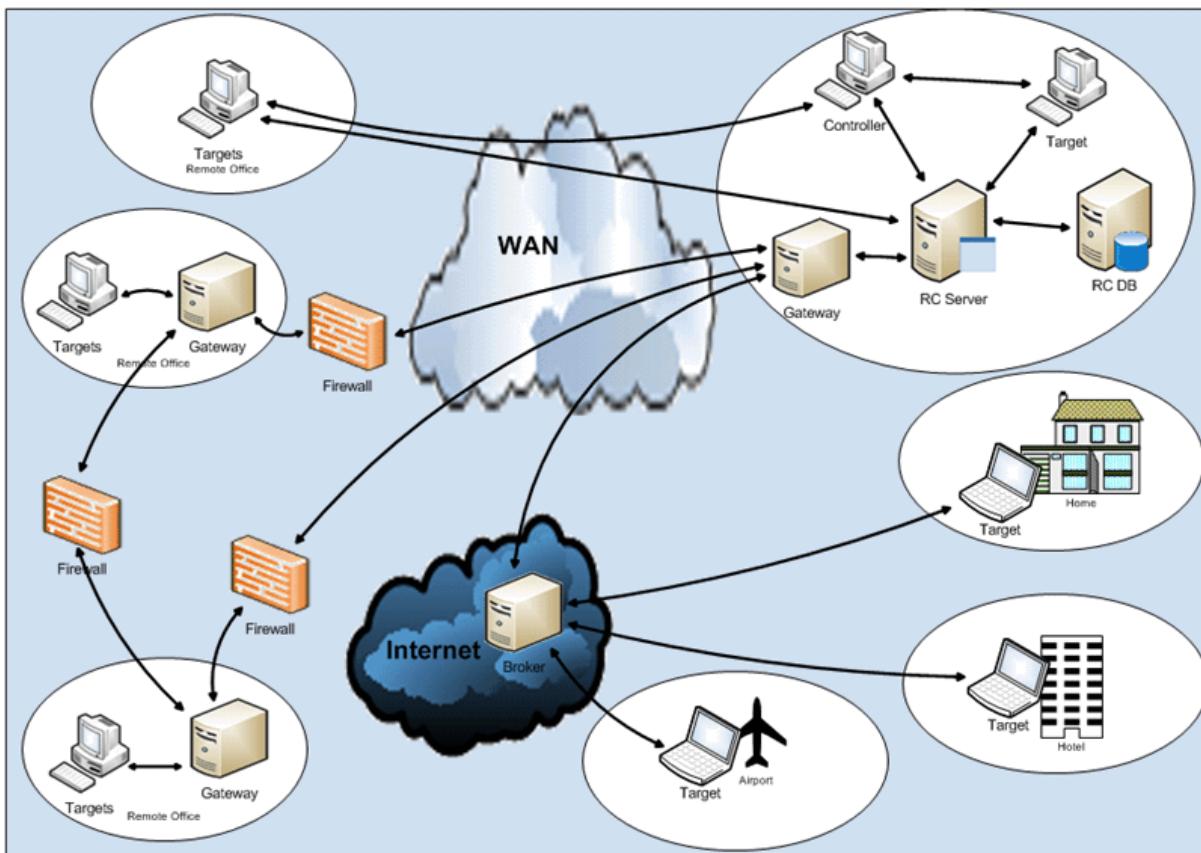


*Figure 18: BigFix Remote Control Infrastructure*

The Server and Broker capacity planning are a function of the number of targets managed.  The remaining components are more universal in requirements. Detailed information is provided in the following tables, along with the following notes.

- Deployments should be 64 bit and meet or exceed the provided vCPU definition.

- The server storage requirement is on average two (2) MB per client in the database.

- Where noted, additional storage is required to maintain video sessions. For example, a five (5) minute eight (8) bit video will require 2MB of storage.  Longer times and higher bit rates will require more storage.

| Deployment Size (Targets) | CPU | Memory (GB) | Storage (GB) |
|---|---|---|---|
| 5,000 | +2 | +4 | +1 |
| 75,000 | +4 | +8 | +4 |
| 225,000 | +8 | +16 | +8 |

*Figure 19: BigFix Remote Control Server Capacity Planning Requirements*

| Deployment Size (Targets) | CPU | Memory (GB) | Storage (GB) |
|---|---|---|---|
| On Demand | +1 | +8 | +2 + video sessions. |
| 6,000 | +2 | +8 | +2 + video sessions. |
| 10,000 | +4 | +8 | +4 + video sessions. |
| 16,000 | +8 | +8 | +8 + video sessions. |

*Figure 20: BigFix Remote Control Broker Capacity Planning Requirements*

| Component | CPU | Memory (GB) | Storage (GB) |
|---|---|---|---|
| Controller | +1 | +4 | +1 |
| Target | +1 | +2 | +2 + video sessions. |
| Gateway | +1 | +2 | +1 |

*Figure 21: BigFix Remote Control Controller/Target/Gateway Capacity Planning Requirements*

## 3.12 BigFix Operating System Deployment (OSD)

BigFix Operating System Deployment (OSD) provides an imaging service. It is an extension of the BigFix platform, meaning based capacity planning derives from the platform root server, database server, and relay infrastructure previously described. However, given OSD is distributing images that can be several gigabytes in size, ensuring the configuration is proper is critical. We will describe the key configuration points, generally based on guidance in the BigFix Non-Functional Requirements (NFR) guide (see the References section).

BigFix OSD key configuration points.

- A properly configured OSD service may provision up to 32 laptops per hour with a basic image in the range of 2GB to 8GB. The larger the image size, the more pressure on the system.
    - o The maximum number of recommended concurrent deployments for a standard configuration is 45.
    - o The maximum number of recommended concurrent deployments increases to 60 when multicast is enabled. More information on the multicast deployment impact is provided below.
- Linux based relays are preferred for the distribution infrastructure. While Windows and Linux

based relays both perform well, at the time of this writing, Linux based relays are more efficient.

- The Bare Metal OS Deployment Server (aka the Preboot Execution Environment Server aka PXE Server) and/or network shares should be in close network proximity to the targets. See item PRF-INF-08 in the BigFix NFR guide for further detail.

- Network speeds throughout the deployment infrastructure are critical. For example, an 8GB image pushed to 32 laptops over an hour implies an aggregate network requirement of 18Mbps. This is in addition to existing bandwidth requirements. As always, data center speeds of 100Mb/s and up are recommended, and the laws of physics apply.

- The download cache throughout the infrastructure must be configured properly. The default is 1GB, and the recommendation is to configure in the interval [100GB, 300GB]. See item PRF-BIG-47 in the NFR guide for further detail.

- To monitor deployments, the following resources are available.

  o The Health Check Dashboard: URL

  o The Deployments Activity Dashboard: URL

- A detailed blog with screenshots for OSD is available here: URL. Note the blog recommendations for the download cache are considered too modest, and the recommendations above should prevail.

- Performance benchmarks for OSD demonstrate the following.

  o The concurrency impact is generally a 1% increase in duration per concurrent request for a well-tuned infrastructure.

  o Multicast offers benefits for concurrency via shared bandwidth. The net effect is reduced network bandwidth requirements with increased download times. The following examples apply for unicast and multicast on a 1Gbps network for a base Linux image over 5 targets.

    ▪ Unicast: 5.5GB of traffic for a duration of 2 minutes.

    ▪ Multicast: 2GB of traffic for a duration of 7 minutes.

  o Whether unicast or multicast is preferred is infrastructure dependent: in a low bandwidth environment, enable multicast.

## 3.13 BigFix Server Automation (SA)

BigFix Server Automation (SA) provides a fault tolerant solution for automation at scale. The capacity planning recommendations are for the SA versions 9.5.72 and above supporting the multi-node plan engine. SA is comprised of two (2) components.

- The Automation Plan Distribution Service.
  This is collocated with the BigFix root server and distributes work to the Plan Engine(s).

- The Automation Plan Engine(s).
  The engine(s) that executes the plans.

See the following figure for a sample multi-node deployment architecture. Some notes on the deployment architecture.

- The distribution service is typically accommodated by the root server capacity planning. When enabling SA, monitoring should be performed on the root server to verify resources are sufficient.

- Multiple Plan Engines should be deployed for redundancy and scalability.

- Each Plan Engine instance will utilize one half of the cores for the compute instance it is deployed on. If Fewer than eight (8) cores are available, only a single core will be consumed.
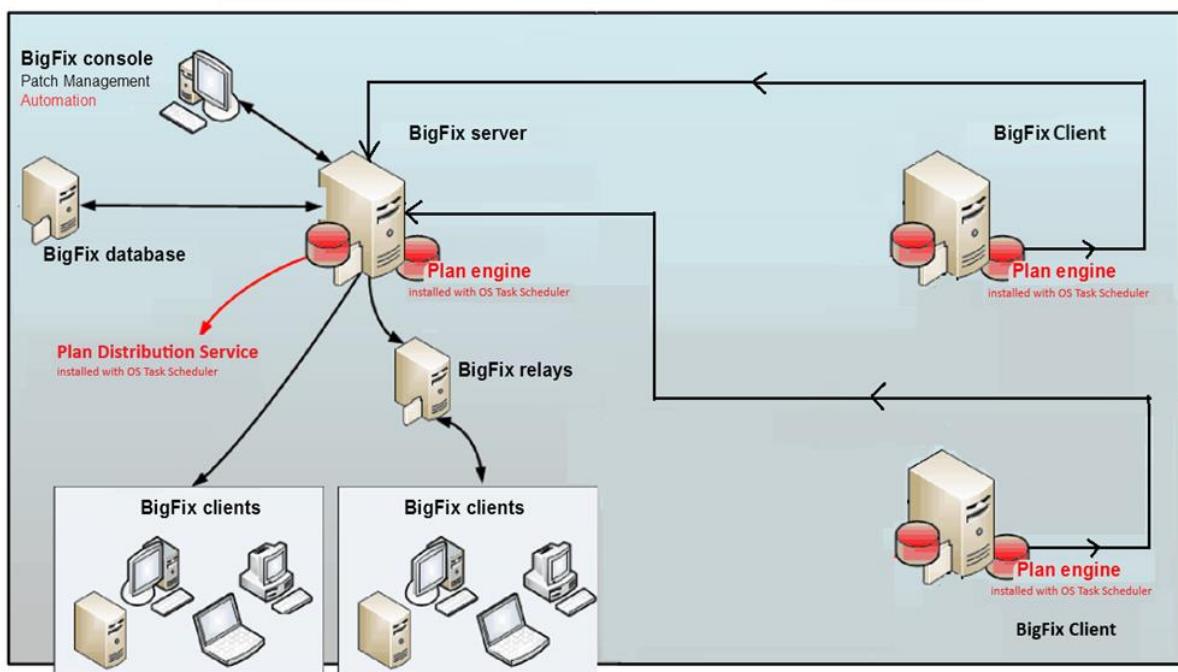
*Figure 22: BigFix Server Automation Infrastructure*

| Plan Engine Instances | vCPU per Plan Engine | Memory (GB) per Plan Engine | Storage (GB) per Plan Engine |
|---|---|---|---|
| Minimum: 1 Recommended: 2 Maximum: Based on Workload | $+ \geq 8$ | +8 | +1 |

*Figure 23: BigFix Server Automation Plan Engine Capacity Planning Requirements*

## 3.14 Message Level Encryption

Message Level Encryption (MLE) provides data encryption support for clients and is particularly valuable for insecure networks or when secure communication in general is required.  It is worth noting MLE does not affect actions taken from the console or Fixes/Fixlets that are already protected by digital signatures.  More information on MLE is provided in the References section.

The following table describes the capacity planning requirements for Message Level Encryption.

| Component | Additional CPU | Additional Memory (GB) | Additional Storage (GB) |
|---|---|---|---|
| Message Level Encryption | +2 | +4 | n/a |

*Figure 24: Message Level Encryption Capacity Planning Requirements*

# 4 Performance Management

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 section 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.

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).

## 4.1 Performance Management Approaches

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

### 4.1.1 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.

### 4.1.2 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.

### 4.1.3 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 actually 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.

### 4.1.4  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.

## 4.2  BigFix Management Recommendations

We will provide configuration recommendations for the BigFix components.

### 4.2.1  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].

o 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.

o 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.

o 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.
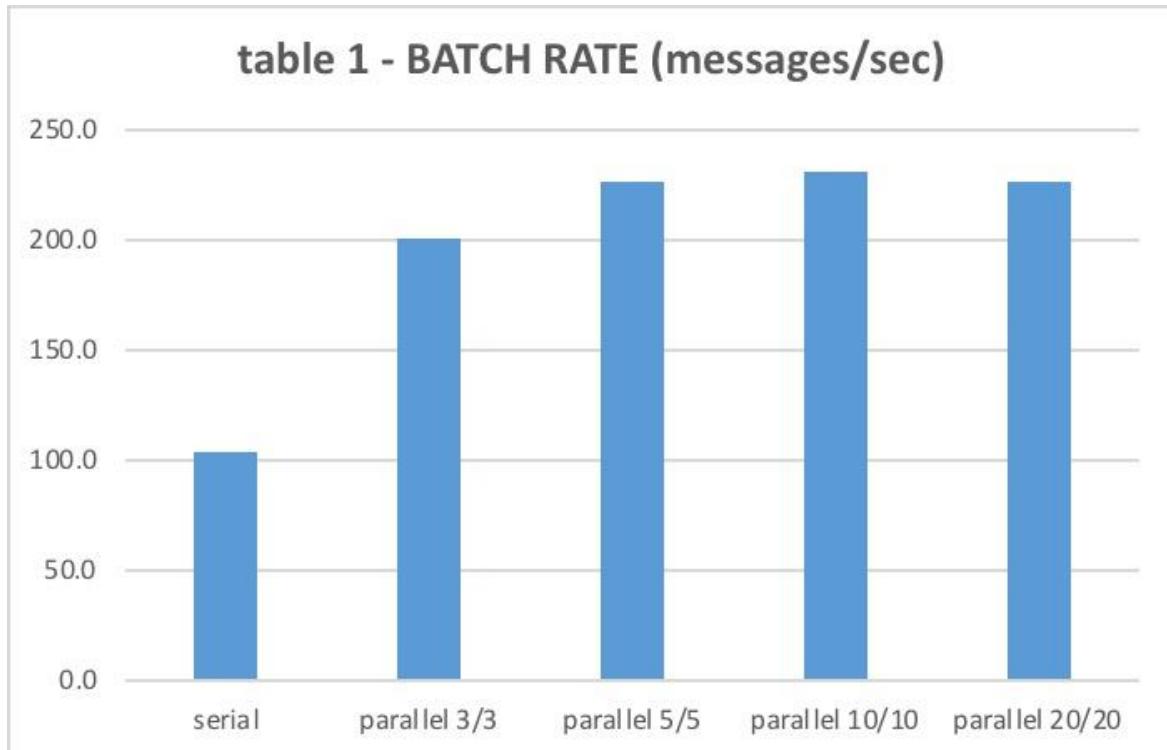


*Figure 25: FillDB Parallelism Impact*

## 4.2.2  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 4.5.1 for more information on Windows TCP/IP management.

## 4.2.3  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.

### 4.2.4  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 not out of the question.

- The console refresh frequency may be set in the BigFix console preference section.

### 4.2.5  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.

### 4.2.6  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, a number of 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).

## 4.2.7  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.

The BigFix Plugin Portal capacity planning is in terms of the number of cloud devices deployed, with implications for the completion of the loop cycle time, depending on the number of operators and devices assigned.  For more advanced Plugin Portal optimization of the loop cycle time, the following steps may be followed.

- The administrator should monitor the loop cycle time and check if it is safely contained in the default refresh interval by enabling the verbose log for the Plugin Portal instance:
  _BESPluginPortal_Log_EnabledLogs -> timing
  _BESPluginPortal_Log_Verbose -> 1

- Then monitor the Plugin Portal log file (BESPluginPortal.log) content and filter for the elapsed time, looking for the row :  Exiting RunOneLoopIteration ( *<elapsed time>* ms).

- In the event the administrator would like to change the interval for the loop cycle time, the following property may be adjusted:
  "Update <*VMware | Microsoft Azure | Google Cloud | Amazon Web Services*> Plugin Refresh Interval", updating the value: "Refresh Interval in Minutes:"

- For example if the loop cycle time exceeds the default two (2) hours, then run the task "Update <*VMware | Microsoft Azure | Google Cloud | Amazon Web Services*> Plugin Refresh Interval", updating the value: "Refresh Interval in Minutes:" to a value greater than two (2) hours.  For example, set the value to three (3) hours and restart the services.

## 4.2.8  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

- o    net.core.rmem_max=1000000

- o    net.core.wmem_max=1000000

- o    net.ipv4.tcp_rmem=4096 87380 167177216

- o    net.ipv4.tcp_wmem=4096 65536 167177216

# 4.3  MS SQL Management Recommendations

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

## 4.3.1  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].

  - o    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.

  - o    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.

## 4.3.2  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 26: MS SQL Query Performance Package Cache*

### 4.3.3  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.

### 4.3.4  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.

### 4.3.5 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).

### 4.3.6 MS SQL Transparent Data Encryption

MS SQL Transparent Data Encryption (TDE) encrypts data at rest. This is a security practice to ensure if the physical storage device is compromised the data will be inaccessible.

Pages are encrypted when written to disk, and decrypted when read into memory. As indicated by the feature name, this is transparent to the application. There is additional processing overhead for encryption management at scale. Encryption operations are typically managed asynchronously. Processor utilization should be monitored before and after the change. The size of the database does not change with TDE.

The features and limitations of TDE are described here: URL

## 4.4 DB2 Management Recommendations

We will describe DB2 instance configuration and query performance.

### 4.4.1 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 27: DB2 Instance Configuration*

### 4.4.2 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 28: DB2 Query Performance Snapshot*

## 4.5 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".

### 4.5.1 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.

### 4.5.2 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.

### 4.5.3 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".

### 4.5.4 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.

In order 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.

### 4.5.5 "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.

- o CPU ready.
  This is the percentage of time the VM is ready to be run but is waiting due to scheduler constraints.
- o 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.

### 4.5.6 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.

### 4.5.7 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.

### 4.5.8 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.

  - o 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.

  - o 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).

  - o 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 29: IO Benchmark Workload Profiles*

### 4.5.9 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)[3]. 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 30: Modifying the Linux IO Scheduler*

---

[3] With Red Hat Enterprise Linux 7, the default scheduler has been set to "deadline".

# REFERENCES

BigFix Non-Functional Requirements: A Checklist Approach

BigFix Maintenance Guide

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™ 8.0

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.

## Trademarks