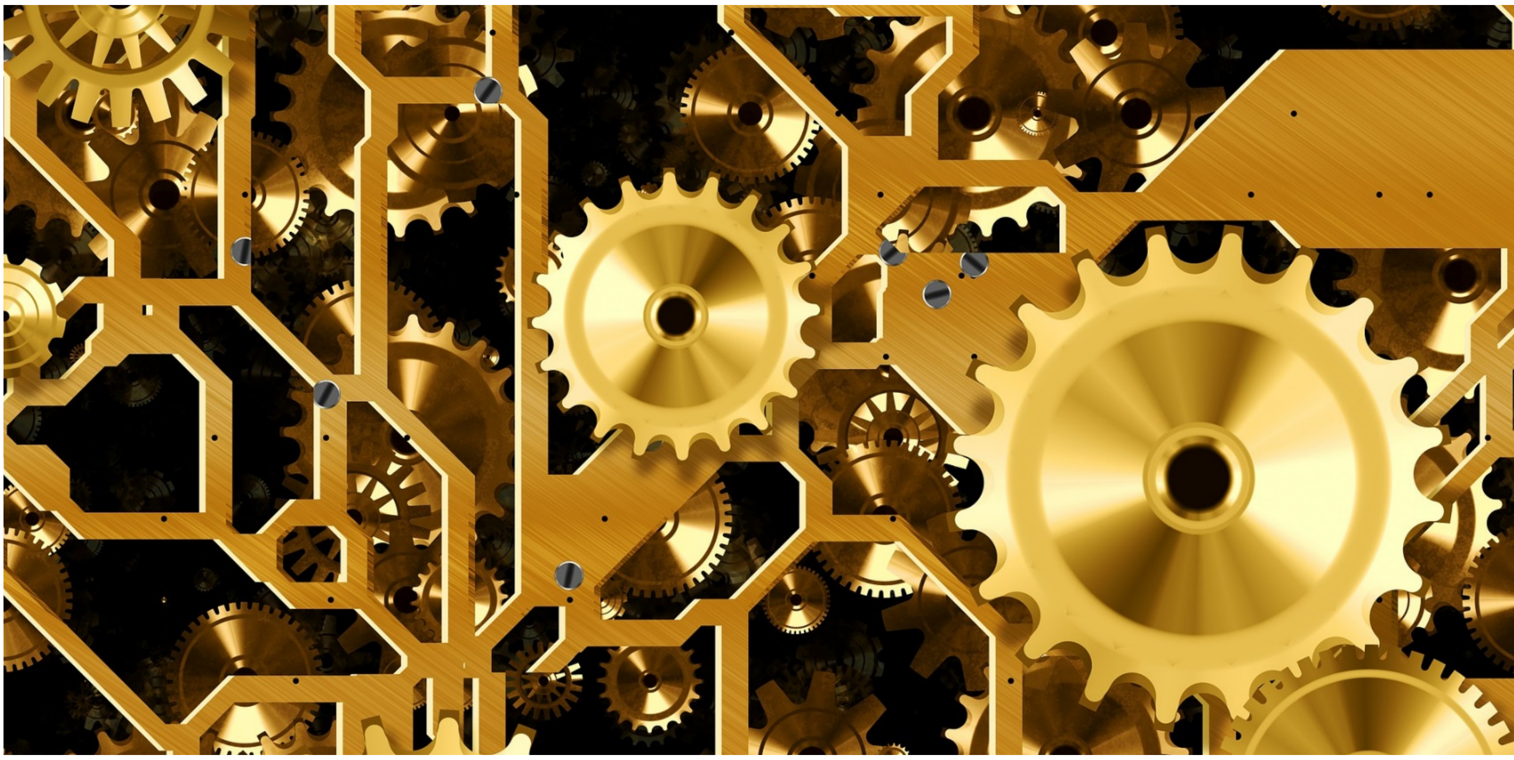


BigFix Capacity Planning

an HCL product

Document version 9.x.18

Mark Leitch
BigFix Performance & Secure Engineering

This edition applies to version 9.2 of BigFix and all subsequent releases and modifications until otherwise indicated.

CONTENTS

LIST OF FIGURES

REVISION HISTORY

Date	Version	Revised By	Comments
December 1st, 2018	9.x.17	MDL	Initial distribution.
January 8th, 2020	9.x.18	MDL	Version & WebUI updates.

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).

- Davide Cosentino
- Francesco Lecciso
- Massimo Marra
- Nicola Milanese
- 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 9.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 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.x releases, it is best oriented towards BigFix Version 9.5 onwards. In addition, a number of 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 a number of 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.

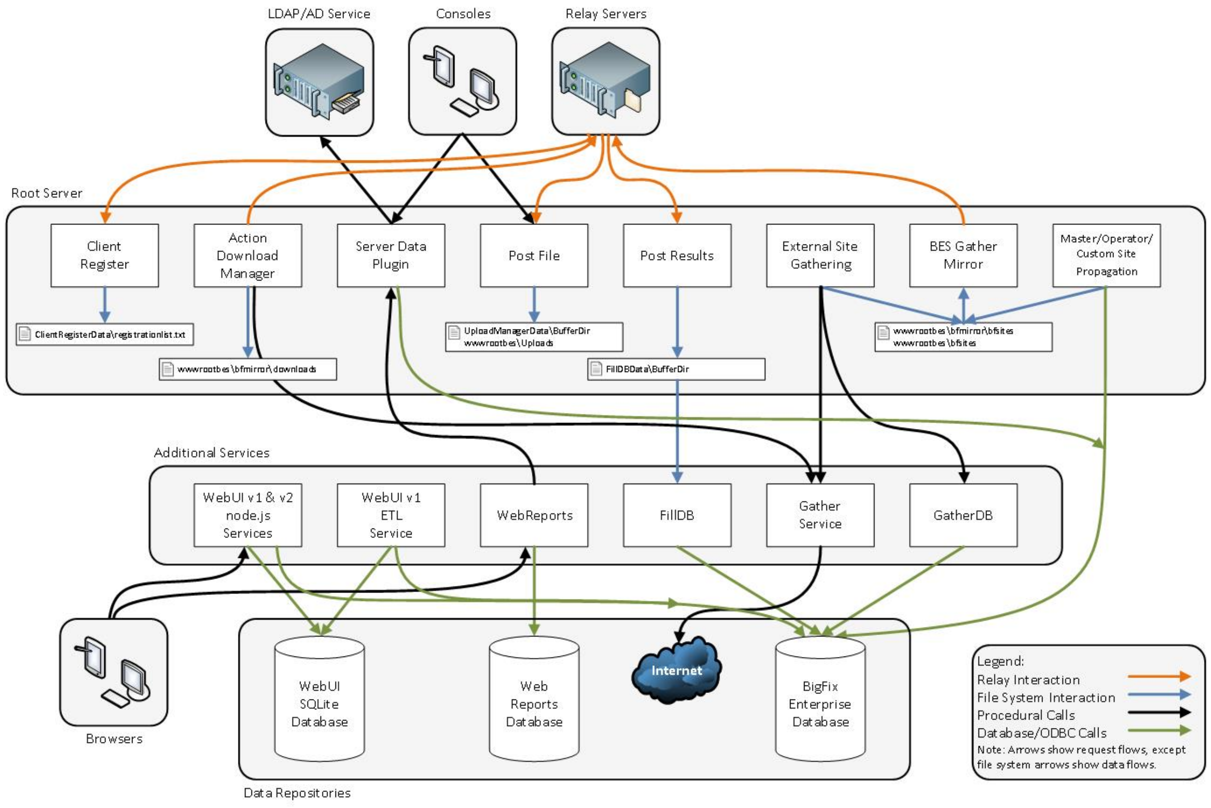


Figure 2: BigFix Architecture

The notable components of this diagram follow.

- Root Server.
  The base BigFix Enterprise Server. It is comprised of a number 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 associated database intended to support the Web based user interface.
- The Web Reports Server.
  The Web Reports Server can provide a variety of stock and custom reports for one or more BigFix server installations.
- Relays.
  A distributed, hierarchical infrastructure to manage the deployment of BigFix agents across diverse network topologies.
- 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, 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

In order 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 a number of 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 model. As a result, hypervisor monitoring is required to ensure a vCPU is 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 rations 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 based 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 number of 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 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 considered to be 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 organizations user population to the number of concurrent users, the standard is for actively engaged users driving high request rates.

3.2 Capacity Planning Tools

A number of 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. They 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.
- Message level encryption.

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.

3.4 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:
    - 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, the database logs, and the database containers.
    - The database logs are typically characterized by the need for very fast sequential IO.
    - The database containers can have much more diverse access patterns, but are highly insulated from storage impact by virtue of the database buffer pools.

  - A reasonable storage breakdown is 30% for the base, 5GB for the database logs, and the remainder (30% - 5GB) for the database containers.
  - 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.).
  - 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, where the Unicode base has driven greater storage requirements.
  - 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.

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	128	1000
250,000	24	128	1250

Figure 4: Root Server Capacity Planning Recommendations

3.5 The WebUI Server

The BigFix WebUI offers a scalable and highly responsive management interface. There have been a number of iterations of the WebUI server. This document will describe the results for the BigFix January 2020 update (9.5.14 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.

- 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 250k endpoint 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,

[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.

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 5: WebUI Capacity Planning Recommendations

3.6 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, in the event that 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 6: Console Capacity Planning Recommendations

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 7: Terminal Server Capacity Planning Requirements

3.7 The Relays and Endpoint Infrastructure

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

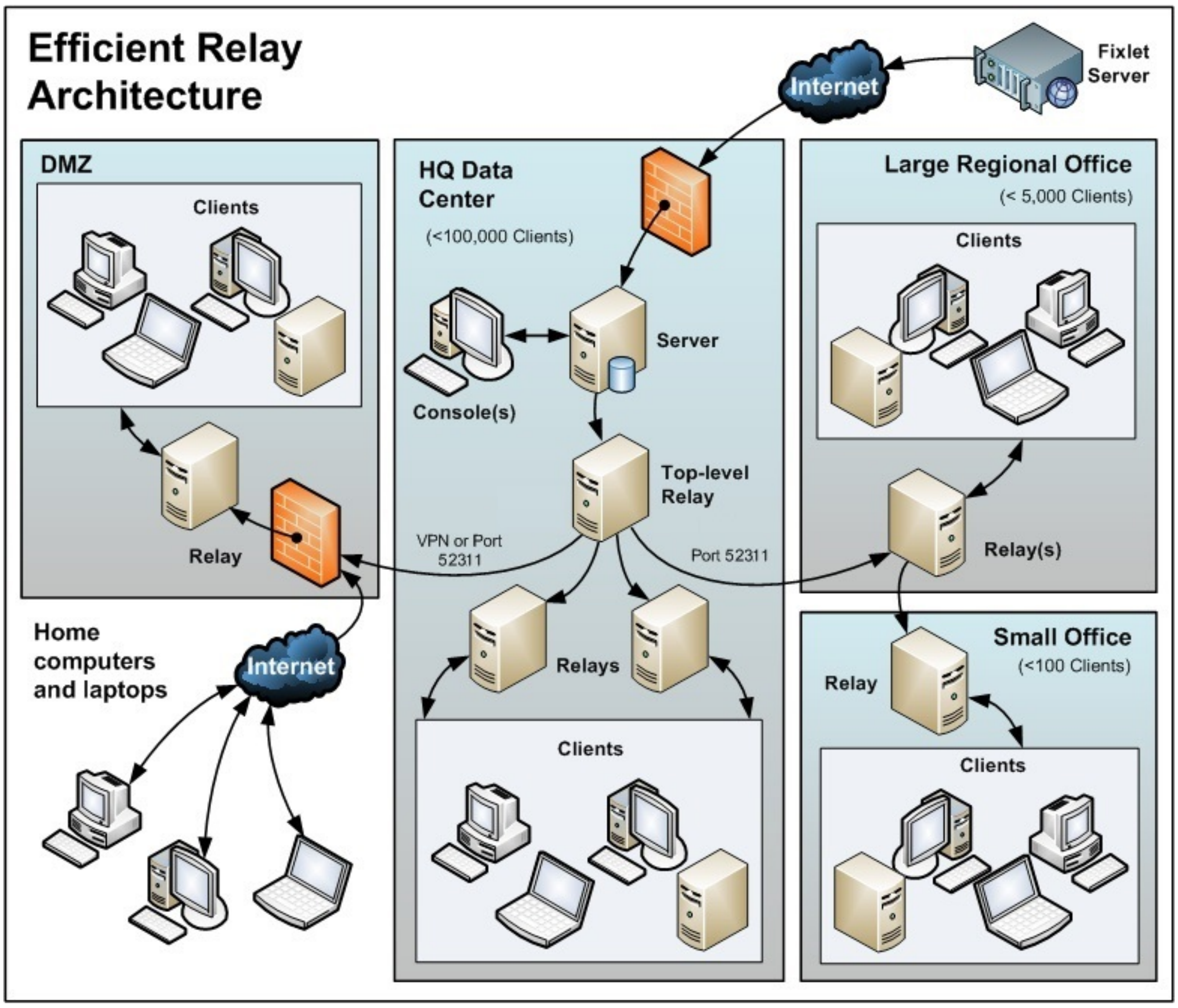


Figure 8: BigFix Relay Infrastructure

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.

3.7.1 Relay Infrastructure Requirements

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

3.7.1.1 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

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

greater concurrency and reduce processor latency.

3.7.1.2 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.

3.7.1.3 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 10 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 4.2.3.3 for more information on managing ulimit.

3.7.1.4 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.
- In order 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 –I 4 <ip>
  - Using these commands, a map may be built showing latency and packet loss.
  - 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.
- See section 4.2.1.2 for further information on relay configuration management.

3.7.2 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 9: Top Level Relay Capacity Planning Requirements

3.7.3 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).
2. For Windows and Linux based relays, the option for a “high scale” relay capable of managing 5,000 agents/endpoints (a 1:5,000 ratio).

In the event you would like a relay to serve somewhere between 1,000 and 5,000 endpoints, you may use intermediate values in the ranges.

Deployment Size (Endpoints Served)	CPU	Memory (GB)	Storage & Network
1,000 (“stock” relay)	[1,2]	[2,4]	Per the infrastructure recommendations section.
5,000 (high scale relay)	[2,4]	[4,8]	

Figure 10: Leaf Node Relay Capacity Planning Requirements

3.7.4 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 a number of 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 a number of 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.

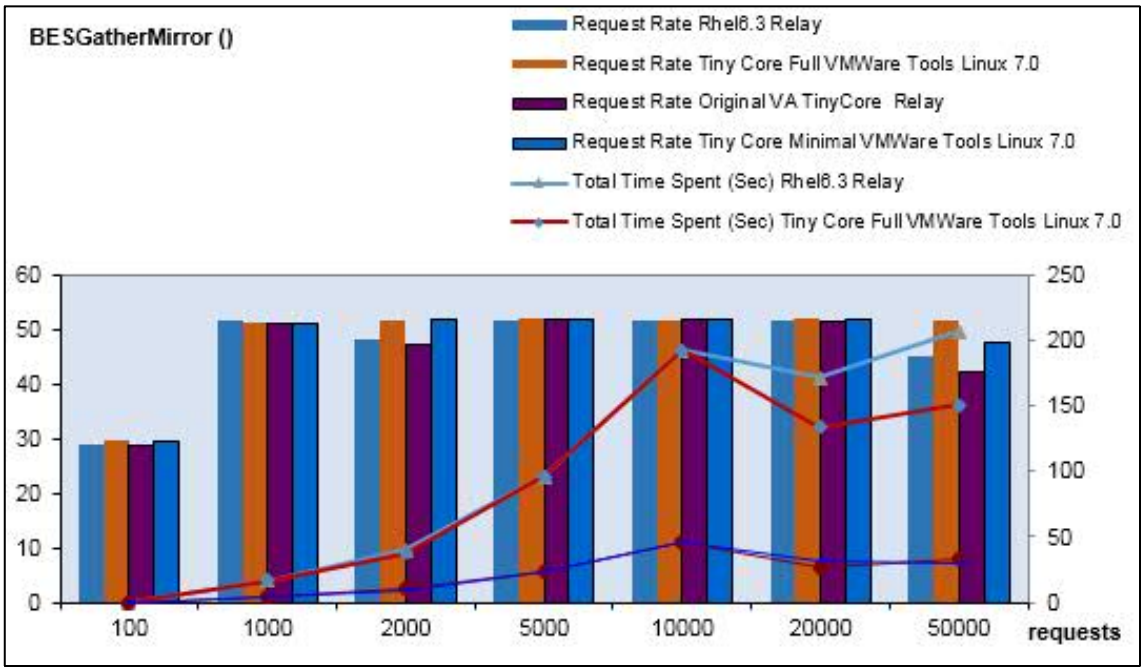


Figure 11: BigFix Relay Virtualization Performance

3.8 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 12: 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 then make specific configuration recommendations.

4.1 Performance Management Approaches

A number of 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 the majority 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 the area of 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: For this component, it can be advantageous to run multiple standalone instances for scalability and resource isolation.

4.2 Performance Configuration Recommendations

We will describe a number of 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).

4.2.1 BigFix Configuration Recommendations

We will provide configuration recommendations for the BigFix components.

4.2.1.1 BigFix FillDB Configuration Management

- 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.

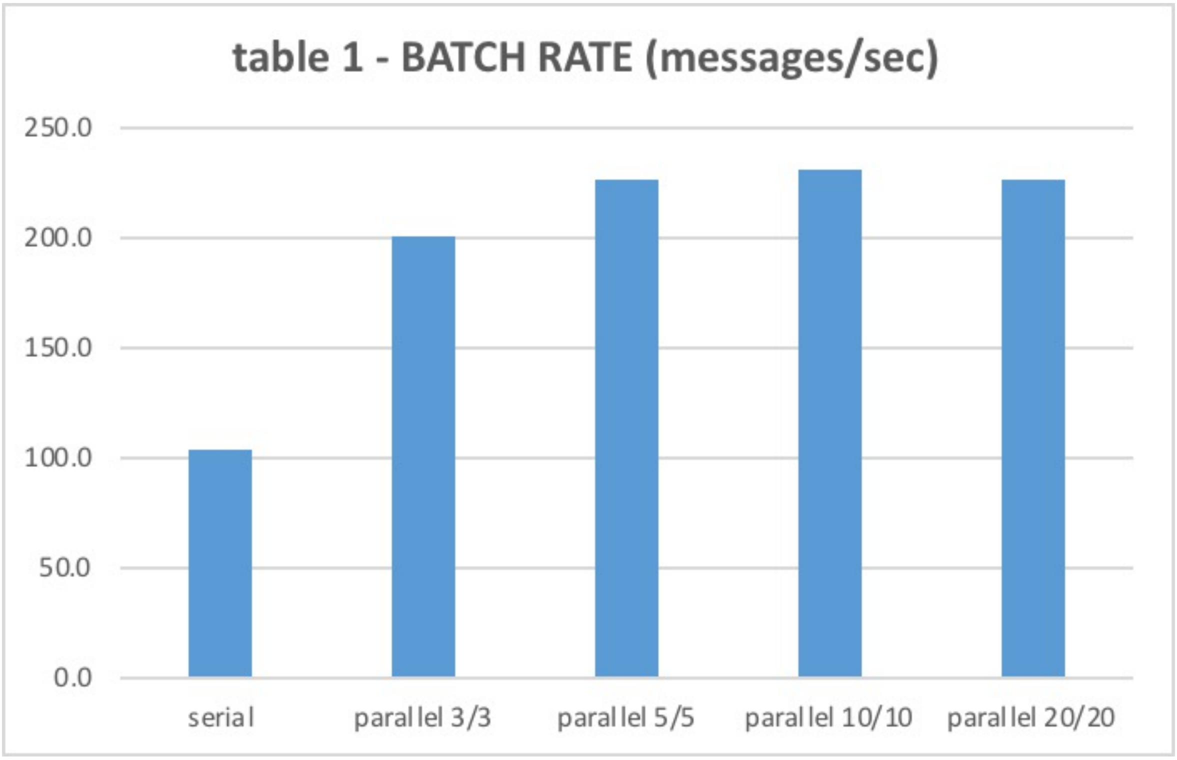


Figure 13: FillDB Parallelism Impact

4.2.1.2 BigFix Relay Configuration Management

- 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 4.2.3.1 for more information on Windows TCP/IP management.

4.2.1.3 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.

4.2.1.4 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.

4.2.1.5 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.

4.2.1.6 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).

4.2.2 Database Configuration recommendations

Recommendations will be provided for MS SQL parallelism management.

4.2.2.1 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. As a result, it is recommended to configure the following settings. Note a future version of the BigFix platform (> 9.5.14) is expected to manage these settings automatically.

- 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]. Information on how to set the MAXDOP and an associated calculator are 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.

4.2.3 Operating System Configuration 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”.

4.2.3.1 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.

4.2.3.2 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.

4.2.3.3 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”.

4.2.4 Virtualization

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.). In order 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.

4.2.4.1 “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.

4.2.4.2 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!
- In order 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.

4.2.4.3 VMware Latency Management

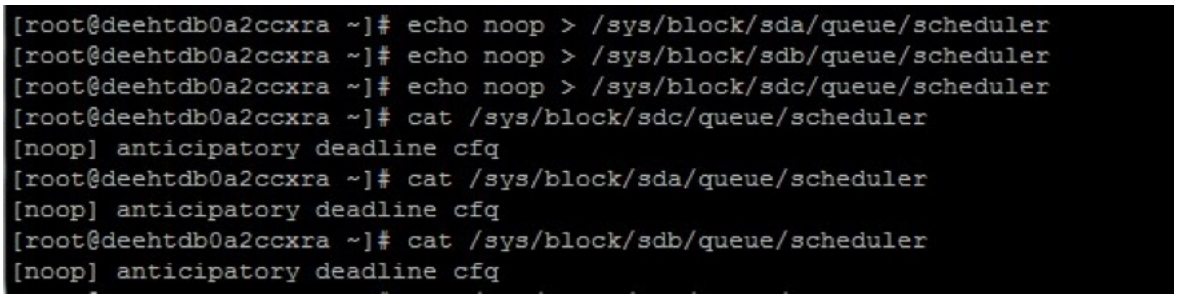
- With VMware vSphere 5.5, it is possible to set the latency sensitivity of a virtual machine. This serves to reduce the impact of virtualization with improved application performance, at the expense of "dedicated" resources. Further information is available through the VMware content in the References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

4.2.4.4 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. In order to achieve this, more information is provided in the benchmarking section below. In addition, we will next describe specific guidelines for IO management for Linux virtual deployments.

4.2.4.5 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 14: Modifying the Linux IO Scheduler

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

References

BigFix Maintenance Guide

MX Performance Toolkit for BigFix

BigFix 9.5 Knowledge Center

BigFix Resource Center

BigFix developerWorks Resource Center

BigFix 9.5 System Requirements

BigFix Message Level Encryption

BigFix Performance Considerations

BigFix Server Disk Performance

BigFix Network Management and Bandwidth Throttling

BigFix Utilities

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

MS SQL Maximum Degree of Parallelism

MS SQL Maximum Degree of Parallelism Calculator

MS SQL Cost Threshold for Parallelism

Performance Best Practices for VMware vSphere™ 5.5

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

Trademarks