Technical white paper

Performance and Threadpools with HP Web Jetadmin

Table of contents

Overview

HP Web Jetadmin is a powerful printer management software solution designed to install, configure, troubleshoot, and manage a printer fleet. Sizing an HP Web Jetadmin installation and optimizing it for performance can be challenging. There are many variables involved to provide an accurate recommendation for the number of devices. The hardware where HP Web Jetadmin is installed, enabled features, and number of devices directly influence the performance of HP Web Jetadmin. This white paper discusses sizing an HP Web Jetadmin installation, features that affect performance, troubleshooting performance issues, and how to potentially improve the performance of many tasks in HP Web Jetadmin by manipulating the number of threads used for each task.

Sizing

A very common question when initially implementing HP Web Jetadmin is how many devices can be included in an installation. The question seems simple, but the answer is complex. HP Web Jetadmin contains a great deal of functionality and feature sets that vary in both resource requirements and performance expectations. HP Web Jetadmin is not a one-size-fits-all type of solution. The hardware where HP Web Jetadmin is installed directly influences the answer. The number of devices and types of features certainly influence the answer. This white paper does not provide an exact answer to this question. However, it does examine features that affect sizing and performance and provide examples of the number of devices tested in various scenarios by the HP Web Jetadmin test lab for comparison purposes. The true number of supported devices on any given HP Web Jetadmin installation depends on many factors, beginning with the hardware.

Hardware requirements

Performance in HP Web Jetadmin is only as powerful as the hardware where HP Web Jetadmin is installed. Commercial implementations of HP Web Jetadmin function much better on server class hosts. Disk system performance is also critical to the operation of HP Web Jetadmin. Much of the disk activity is tied to HP Web Jetadmin interacting with a Microsoft® SQL Server database instance. High-performance storage speeds up these operations and helps the application run much better.

Server hardware

HP recommends the following server hardware configuration:

- 4 or more processor cores
- 2.8 GHz or higher processor speed
- 4 GB or more of RAM
- 4 GB of available storage

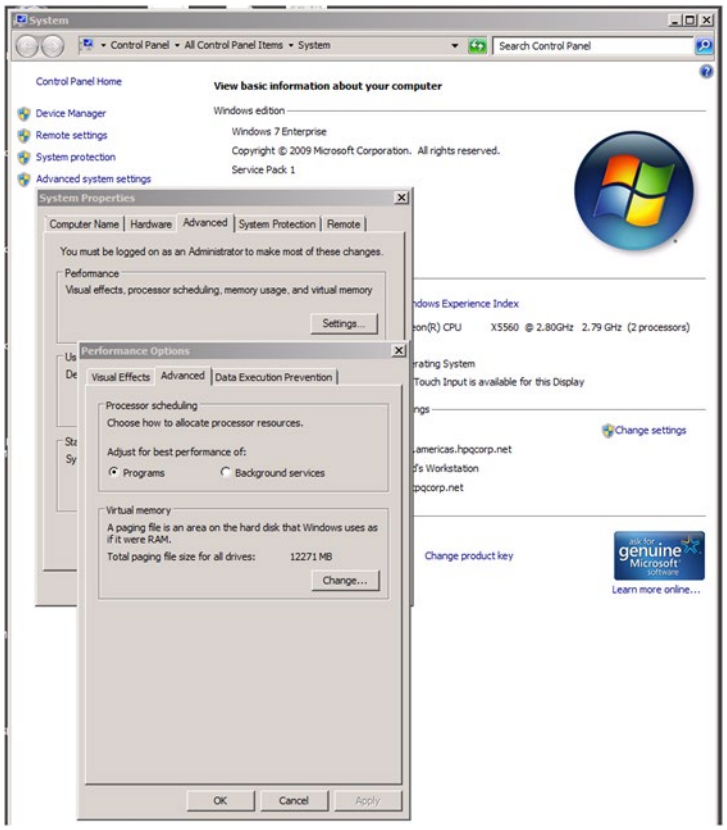
Virtual machines

For VMware environments, appropriate processors and RAM must be dedicated for each virtual machine. For a VMware server, the virtual machine network must be set to bridged to facilitate HP Web Jetadmin communications. It is very important to configure VMware so that its guest or virtual systems have enough resources to support HP Web Jetadmin and Microsoft SQL Server. To ensure that the appropriate resources are provisioned, see the support documentation for the VMware version you are using.

Memory

RAM paging is a process that Windows uses when it runs out of RAM (memory). If HP Web Jetadmin runs on a host that runs out of RAM and begins RAM paging, the application responds very slowly. The UI is very sluggish. Most of the applications on the host are also sluggish. Steps can be taken to troubleshoot this problem. Free memory can be observed by going to **Windows Task Manager** > **Performance** tab > **Physical Memory** > **Available**. More memory might need to be added if available memory is low. Performance counters can be used to see if HP Web Jetadmin or the database needs additional memory. Running the client on a separate machine than the HP Web Jetadmin server can also help.

The Windows default for the maximum paging file size might be inadequate, even on 64-bit systems, for large HP Web Jetadmin installations that support thousands of devices. When this happens, Windows might be unable to fulfill requests for additional memory, causing the HPWJA Service to fail. The Windows 7 paging file size is configured in the Virtual Memory dialog of the Advanced Performance Options tab of the Advanced System Properties settings.



Disk and SQL Memory

Another performance consideration is the disk subsystem. If the database runs from a very slow disk, performance is affected. Also, the disk should be defragmented periodically to make sure the files are in a good state. Use **drive**, **Properties**, **Tools**, **Defragmentation** to determine if the drive needs to be defragmented.

HP Web Jetadmin is continuously querying the database. In order to minimize hard disk I/O operations, HP recommends increasing the SQL server memory size. In HP Web Jetadmin, go to **Tools** > **Options** > **Shared** > **General** > **Database**, which can be set to a maximum of 1024 MB. With Microsoft SQL Management Studio, this value can be increased to a much higher value. In SQL Management Studio, right-click the Instance and select **Properties** > **Memory**. The **Maximum server memory** can be increased to a higher value, such as 2048 (2GB) or more. As long as no paging occurs, this value can be increased to prevent disk I/O operations.

Concurrent user login

HP Web Jetadmin is a client/server application, which means that the client can run on a different machine than the server. The majority of the work is contained and managed by the server. Multiple clients can be connected to the same server. A client connects to the server and uses the .NET Remoting channel to communicate. There are two types of communication that occur between the client and the server—client-driven requests and events from the server.

A client makes requests to the server in response to user actions or server events. This starts when the user launches the client. The client establishes a connection with the server, retrieves updates, and gets any information required to launch the UI from the server. As the user uses the application, the client requests data from the server or requests work to be done by the server as necessary. This traffic is generally very predictable because it is a direct result of the user's work.

HP Web Jetadmin is architected for multiple-user access. The application is designed to support 15 or more concurrent connections while maintaining a high degree of performance. Of course, application usage makes the performance at each client vary somewhat depending on the client load and the hardware hosting the application. The hardware and software requirements should be considered when the application is loaded with multiple simultaneous user connections. An example would be where one user is tasked with discovering all the devices in the enterprise and at the same time another user is tasked with establishing groups and generating reports. In this case, on a slower host, HP Web Jetadmin might exhibit slow performance when large reports are being generated and discoveries are being run at the same time. System resources, network bandwidth, and other factors have an impact on how well the HP Web Jetadmin performs under these conditions.

Application performance

HP Web Jetadmin processes large amounts of data while allowing users to work productively. When users view device lists or access device configuration or status details, HP Web Jetadmin spends CPU bandwidth on multiple device data retrievals. In addition, the application collates and communicates device information back to multiple client sessions.

An observant administrator can use tools, such as Microsoft Performance Monitor, to view the results of this background processing.

Here are some characteristics about performance and the HP Web Jetadmin software:

- During background task executions, HP Web Jetadmin might consume a high percentage of the host's processor for only short durations of time.
- Consumption of the system processor might be pronounced on host systems that meet only the minimum hardware requirements.
- Consumption of the system processor might be pronounced on single host systems that run both the HP Web Jetadmin application and the client application on the same host.
- Performance gains can be realized when HP Web Jetadmin is run on multiple processor systems.

HP Web Jetadmin relies on network traffic to keep the information displayed to the user reasonably up-to-date. HP Web Jetadmin carefully makes numerous tradeoffs to balance network traffic with up-to-date information and performance.

The following sections examine several types of features and functionality in HP Web Jetadmin and their effect on performance. The following definitions are used to describe resource usage:

- *Light*—General minimal resource usage with occasional spikes in resource usage. The impact on other applications is minimal, but sometimes noticeable.
- *Fluctuating Moderate*—Moderate resource usage with frequent spikes in resource usage. The impact on other applications is noticeable, but not generally excessive.
- *Fluctuating High*—Resource usage, especially CPU usage, is generally at 100% with regular dips down to lower levels. The impact on the OS and other applications is severe, although they remain functional.
- *Severe High*—Resource usage is high and remains there for extended periods. HP Web Jetadmin is generally not very responsive. The impact on the OS and other applications is severe as resources are generally unavailable.

Generally, all activities that engage in significant database activity, especially those that both read and write, jump to the Fluctuating High resource-usage level for the duration of the action. The system returns to a Light resource-usage level after the action is completed.

Discovery

Light to Fluctuating Moderate

Running a discovery generates traffic between the server and the device to identify the device and add it to HP Web Jetadmin. Some discovery methods are more expensive than others when it comes to network traffic. For example, the Active Directory discovery method is one of the cheaper discovery methods to perform with regards to the amount of network traffic generated. An Active Directory discovery gleans IP addresses using print queue objects in the customer's Active Directory, and then only talks to network nodes that have printers on them. The same can be said of Specified Address discoveries because they are directed to specific IP addresses already known to be printers. Conversely, an IP Range discovery is more expensive on the network because HP Web Jetadmin must communicate with each network address in the range to determine if it has a known printer.

For the most part, resource usage is somewhat minimal during discoveries regardless of the method.

Adhoc requests

Fluctuating Moderate

Many parts of the client require device data. Depending on how up-to-date the client wants the device information to be, the server might return data about the device that it already has or it might need to talk to the device to get the information. This class of requests is generally driven by the user on an on-demand basis. With multiple clients, on-demand requests have the potential to overwhelm the system. To compensate for this, the HP Web Jetadmin server tries to reduce the device traffic through the use of thresholds. If multiple clients request the same data on the same device within a specified time threshold, only one communication is sent to the device. Also, the relevant results of all device requests are shared with all clients regardless of which client originally requested the data. This allows all clients to potentially benefit from the most up-to-date information at a low cost.

Device polling

Light (spikes in CPU usage during polling, dropping to negligible between polling)

The device list is core to HP Web Jetadmin. HP Web Jetadmin automatically keeps the device list content reasonably up-to-date through a slow polling mechanism on the server to each device. The entire device list is registered on a background poller for the columns in the current layout on a per-client basis when that client is on a device list page. This mechanism is intended to gradually fill in the entire set of data so that it is ready when the user requests it by scrolling down through the device list. Each device that is displayed registers with the device list poller. This mechanism is intended to update the visible space at a much faster rate than the background poller would retrieve the information. The registration occurs after scrolling in the device list stops for a short time. The devices are unregistered after scrolling starts again.

NOTE: With a single client, updates to the visible devices in a device list should start happening quickly at a rate consistent with the settings for the device list poller. With multiple clients, the updates should continue to be regular, but with less frequency.

Each object has a time threshold associated with it that allows HP Web Jetadmin to prevent a device communication if it already has reasonably up-to-date information. If one client just asked for the device description one minute ago, the threshold might cause that device communication to be skipped. Device data is cached in the client to minimize additional communication with the server. Generally, if a part of the UI requests device data, the client cache can prevent communication back to the server.

The settings for device polling can be changed in the Web Jetadmin client UI by going to **Tools** > **Options** > **Device Management** > **Device Polling**.

For an explanation of all the polling settings, see the topics in the HP Web Jetadmin online Help available at **Introduction to HP Web Jetadmin** > **Device Management Configuration Options** > **Device Polling Configuration Options**.

Set the next time to poll all device custom

This configuration setting would override the default background poller settings. Once this setting is configured, the next background polling for all devices would happen only after the specified number of minutes from the time when the last polling got completed.

1.This file is available in the following directory on the HP Web Jetadmin server:

C:\Windows\ServiceProfiles\NetworkService\AppData\Local\HP Inc\HPWebJetadmin\WjaService\config

2.Open the MoabDevice.config.xml file in Notepad or a similar editor.

3.Change the <value>*100*</value> attribute in the AllDeviceTimeBetweenBGpollingInMin. This will cause the WJA background poller to run after 100 mins from the last poll completed time.

For example.

```
<property name="AllDeviceTimeBetweenBGpollingInMin">
    <type>HP.Imaging.Wjp.Sdk.Core.Framework.ConfigurationItemString</type>
    <value>100</value>
 </property>
```

4.Close and save the file.

5.Restart the HP Web Jetadmin service (HPWJAService).

6.Launch the HP Web Jetadmin client.

Automatic groups

Fluctuating High

Automatic groups make use of the background poller. Any automatic group or filter group in the system registers a set of requests from the device when polled. When HP Web Jetadmin detects changes in the information being requested, it sends out a device-changed event that causes groups and filters to re-evaluate the device in question against the group or filter criteria.

Basic (no-policy) bound groups cause a slight increase in resource usage, but not a significant amount. Increasing the number of groups populating at one time and the number of devices going into the groups does little more than extend the time taken to populate the groups.

Creating automatic groups with multiple policies, including an alert subscription, can dramatically increase resource use. Large quantities of disk access can be caused by simultaneously reading the device data from the database (to filter the devices), writing the new group membership to the database, writing the content of the custom request to the database, and reading the alert eligibility information from the device requests.

Report generation

Fluctuating High

Data collection is performed once per day and the amount of data collected fluctuates. Report generation analyzes the data that is collected nightly and formulates reports for the user. Both of these areas can cause Fluctuating High resource usage, returning to Light as soon as the task is completed.

Device list export

Fluctuating High to Severe High

Device exports can be defined to pull data only from the database or to pull all data from all devices on-the-fly, so resource usage can vary. Large numbers of devices and columns in the device lists can cause high levels of resource usage during the duration of the device polling.

Device configuration

Light to Fluctuating Moderate

Device configuration occurs at a steady rate and never taxes the system above Fluctuating Moderate resource usage. Some configuration items cause more overall network traffic than others, but resource usage always remains low.

Alerts

Fluctuating High to Light when complete

Subscribing to a simple set of alerts (such as Paper Out, Toner Low, Toner Out, Offline, Printer Error, Intervention Needed, Paper Jam, Cover Open, and Manual Feed Needed) can cause varied system resource usage during this time, dropping to Fluctuating Moderate for short periods or running up to Severe High for short periods. The majority of the time it runs at Fluctuating High resource usage. After completion, it drops back down to Light resource usage.

Light to Fluctuating Moderate

Processing alerts usually involves device polling that is dictated either by traps or an adaptive/static polling rate. HP FutureSmart devices use Web Services Eventing to process alerts. In all cases, resource usage remains acceptable unless the pollers become overloaded.

Firmware

Light to Fluctuating Moderate (when running eight concurrent upgrades)

Populating firmware information on the **Firmware** tab can spike due to the size of the files being imported. Downloading firmware is typically light on system resources because it involves opening a port 9100 TCP connection and sending down an RFU file for many devices. HP FutureSmart devices use Web Services for upgrades.

Sizing examples

While it is difficult to provide exact recommendations because all installations vary, the HP Web Jetadmin team has run many tests to provide examples of what can be performed under certain scenarios before performance degrades occur. The team benchmarked the following scenarios to attempt to determine the tipping point when HP Web Jetadmin performance degrades.

Test Plan Platforms—all systems running Windows Server 2003 R2

- 8 VMs running the base hardware configuration—Dual core processor with 4 GB RAM; client, server, and DB running on the same system
- 1 VM running the base configuration with an off-box DB
- 1 VM running a quad processor with 12 GB RAM

Test Plan Scenarios

- Two of the machines performed a Discovery (one with a device list of 1,100 and the other with a device list of 4,000).
- Two of the machines configured Alerts that began at a safe level and slowly increased until a failure point was encountered (one with a device list of 1,100 and the other with a device list of 4,000). The Alerts exercised were General Alerts with the options of Advisory, Media Path, and Supplies.
- Two of the machines performed Data Collections that began at a safe level and slowly increased until a failure point was encountered (one with a device list of 1,100 and the other with a device list of 4,000).
- Two of the machines performed Data Collections with Hourly Peak Usage exclusively (one with a device list of 1,100 and the other with a device list of 4,000). This is due to early experiments showing that this particular collection presents a heavy load due to the frequency of queries to the devices.

Test Plan Strategy—Scenario Matrix

- At the completion of these scenarios, a matrix is applied that entails each system performing the various operations (Discovery, Alerts, Data Collections) simultaneously with a slowly increasing load that mirrors the ramp of the individual scenarios (one with a device list of 1,100 and the other with a device list of 3,900 = 16* subnet).

Completed Discovery

- Discovery—Imported list of 1,100 devices

  Result = Success

- Discovery—Run Full Subnet through Templates with each portion being smaller than a Class B Network

  Result = Success

Completed Alerts

- Alerts—(1,100 devices) General Alerts: 100 to Max

  Result = Success

- Alerts—(4,000 devices) General Alerts: 100 to Max

  Result = Success

Completed Data Collection

- Data Collection—(1,100 devices) All except Hourly Peak Usage: 100 to Max

  Result = Success

- Data Collection, Hourly Peak Usage—(1,100 devices) Data Collection: 100 to Max

  Result = Success

- Data Collection—(4,000 devices) All except Hourly Peak Usage: 100 to Max

  Result = Success

Completed Remote DB with SSD

- Data Collection—(4,000 devices) SSD Remote DB, All except Hourly Peak Usage: 100 to Max

  Result = Success

- Alerts—(4,000 devices) SSD Remote DB, General Alerts: 100 to Max

  Result = Success

- Data Collection—(4,000 devices) SSD Remote DB, Hourly Peak Usage: 100 to Max

  Result = Success

Conclusions

- Individual tasks, no matter how intensive, do not cause issues for HP Web Jetadmin when running on 64-bit systems if the device list is limited to approximately 1,100 devices.

- System-intensive tasks, such as Hourly Peak Usage Data Collections, can be successfully applied to 1,100 devices with no adverse reactions in HP Web Jetadmin, including responsiveness of the client even when the client runs on the same host.

- When device lists of 4,000 devices are used, individual tasks slow performance only at the client, but do not cause system losses (for example, crashes).

- The only limitation on a system with 1,100 devices is when simultaneously applying a large Alert subscription and creating a large Data Collection. This scenario can cause the client to become temporarily unresponsive. In this case, the server continues to run without interruption.

As a result of these tests, one might conclude that a system in full usage can support roughly 4,000 devices or more. Remember, this is a system that is being used with many HP Web Jetadmin features activated. Can one HP Web Jetadmin server support more than 4,000 devices? Absolutely. However, the answer also depends on how you load features and devices as well as the types of features and settings activated. As more features are activated on devices in a single HP Web Jetadmin host, the number of potential devices under management decreases. Administrators might or might not plan to use all the HP Web Jetadmin features in one HP Web Jetadmin implementation. For example, there has been an installation of HP Web Jetadmin that exceed 20,000 devices in the device list, but in this case, the installation only runs data collections and reports.

Distribute HP Web Jetadmin across multiple servers

The first factor in sizing is to decide if one implementation of HP Web Jetadmin is possible or if the implementation of HP Web Jetadmin should be broken into multiple servers. If it is believed that the number of devices and desired features will overload a single server, perhaps the installations can be split by groups of devices or by functionality.

Consider the following multiregional approach:

- Europe—2,300 devices and Reports are needed

- North America—6,000 devices and both Reports and Alerting are needed

- Asia—3,000 devices and Reports are needed

In this case, both functionality and devices must be considered. In the Europe and Asia case, one server might be implemented for each population, while in the North America case, a two server implementation might be considered.

Here are some more hints on how HP Web Jetadmin features and devices impact the system.

Consolidated reports—This item cannot be accomplished through distribution across multiple servers. If the number of devices is large, deployment should consider one server dedicated to reports data collection. In highly scaled environments, data might have to be drawn out of multiple HP Web Jetadmin servers using Database Views.

By User/Peak Usage reports—If By User or Peak Usage reports are required across a large number of devices, consolidated reports is probably not an option. These data collections must be distributed, so hopefully this work can be distributed across multiple servers by geography, function, or some other logical grouping.

Alerts—Alerts is a very common bottleneck. However, it is also the easiest to distribute across servers. Reducing the alerts load on any given server increases the performance and responsiveness. Regional or organizational alerting across multiple servers is easy to implement.

Auto-grouping—Many customers use auto-grouping to drive configuration policies. Auto-grouping causes a constant draw from system resources, so there is often a benefit to having a server dedicated to configuration and auto-grouping policies. Many times there is a difference between groups that provide general management structure and organization and groups that drive automatic functionality. Whether those auto-groups need to be replicated across all of their servers should also be considered.

Number of devices per server—When looking to distribute functionality across servers, it is useful to limit the number of devices in each system. This increases the responsiveness across devices for the intended functionality. For example, if a server is dedicated for alerts in the North America region, make sure that the server only knows about the devices in the North America region that need alerts.

Client load—Client load is another item to consider, although it is harder to distribute unless there is a clear delineation by geography or other logical grouping of devices. Many times the recommendation is to set aside a server for help desk functionality that tends to have more ad hoc use models.

Sample design proposal

After all options have been considered and the customer environment is understood, it is time to design a proposal. Consider a case study of an HP Web Jetadmin implementation. The following key functionality is required:

Deploy

- Discover, configure, and update firmware on a fleet of 3,000 devices
- Use auto-groups and templates to apply security settings
- Export various device list fields to a CSV file on a schedule
- 15 user potential

Resolve issues

- Investigate device issues remotely
- Apply resets remotely
- Send PJL and configuration templates to devices
- Monitor (proactive) 3,000 devices
- 50 user potential

Reporting

- Establish data collection on all but Peak Usage and By User on 3,000 devices
- Retain data for up to 5 years
- 15 user potential

Design recommendation—Distribute functionality across three servers

Here is an actual proposal from a customer implementation in 2009:

- Server 1
  - Proactive Management - Distributed Alerts
  - 2 or 4 core SQL Express
  - ~3,000 devices
  - Subscriptions 1-3
- Server 2
  - Fleet Deployment and Trend Reporting - Consolidated Reports and Admin support
  - 4 core and off-box SQL
  - 3,000 devices
  - ~15 admins

Increase SQL memory size

With a large database for HP Web Jetadmin, disk swapping and continuous data reading happens, which negatively impacts overall performance. To prevent this, increase the virtual memory size for SQL in the HP Web Jetadmin client UI by going to **Tools** > **Options** > **Shared** > **General** > **Database**.

NOTE: The default SQL memory size is 2 GB for new installations of HP Web Jetadmin10.4 SR3 and later and 0.5 GB for earlier versions (this default value is not changed after upgrading to HP Web Jetadmin 10.4 SR3).

NOTE: HP Web Jetadmin 10.4 SR2 and later only offer up to 1 GB for the SQL memory setting. For higher settings with the previous versions, the memory size must be adapted directly in the SQL database.

Limit configuration log size or add additional indexes to the configuration log

HP Web Jetadmin stores the configuration results in two tables: dbo.DAV_DCONFIG_ERROR and dbo.DAV_DCONFIG_MOE_VALUE. When the number of entries in these tables is extremely high and SQL memory is limited, HP Web Jetadmin startup time increases. To increase performance, there are a couple of options:

1. Increase the SQL memory size.

2. Delete old entries from the tables. In the HP Web Jetadmin UI, go to **Tools** > **Options** > **Shared** > **Server Maintenance** > **Configuration**. If old entries have to be saved, enable the **Archive expired entries to file first**, and then click **Apply**. Change the **Retention time** to a shorter time period, click **Apply**, and then click **Clear History**.

3. Add additional indexes to the tables. By default, the dbo.DAV_DCONFIG_ERROR and dbo.DAV_DCONFIG_MOE_VALUE tables have clustered indexes. Adding non-clustered indexes improves performance when there is a high number of entries in the tables. For a low number of entries, adding a non-clustered index decreases the performance. Testing showed a decrease in performance with the additional index with only hundreds of entries, and an increase in performance when there were several thousands of entries.

4. To add indexes follow these steps:

    a. Open Notepad and copy and paste the following text into Notepad:

    ```
    @echo off
    osql -E -S ServerName\HPWJA -d HPWJA -Q "CREATE NONCLUSTERED INDEX
    [NonClusteredIndex-For-table-Dconfig_Error] ON [dbo].[DAV_DCONFIG_ERROR]([MoeNdx]
    ASC)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
    DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS =
    ON) ON [PRIMARY]" -b
    osql -E -S ServerName\HPWJA -d HPWJA -Q "CREATE NONCLUSTERED INDEX
    [NonClusteredIndex-For-table_DAV_DCONFIG_TEMPLATE_MOEVALUE] ON
    [dbo].[DAV_DCONFIG_TEMPLATE_MOEVALUE]([MoeNdx] ASC)WITH (PAD_INDEX = OFF,
    STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE
    = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]" -b
    echo  Index added successfully
    ```

    b. Change the ServerName to the actual name of the server (where the SQL database is running).

    c. Select **Files** > **Save As**, and select **All Files** from the **Save As Type** list.

    d. Specify a file name with the extension .bat, and then click **Save**.

    e. Open a command prompt with admin rights, go the path where the file is saved, type the file name, and then press **Enter**. The additional indexes are created.

Clustered indexes sort and store the data rows in the table or view based on their key values. These are the columns included in the index definition. There can be only one clustered index per table because the data rows themselves can be sorted in only one order.

Nonclustered indexes have a structure separate from the data rows. A nonclustered index contains the nonclustered index key values and each key value entry has a pointer to the data row that contains the key value.

Threadpool settings

HP Web Jetadmin provides an optional technique for improving the performance of many tasks by manipulating the number of threads used for each task. Threadpool settings can be configured in an XML file named PerformanceTuning.config.xml in the following directory for Windows 7 and Windows Server 2008 (R2):

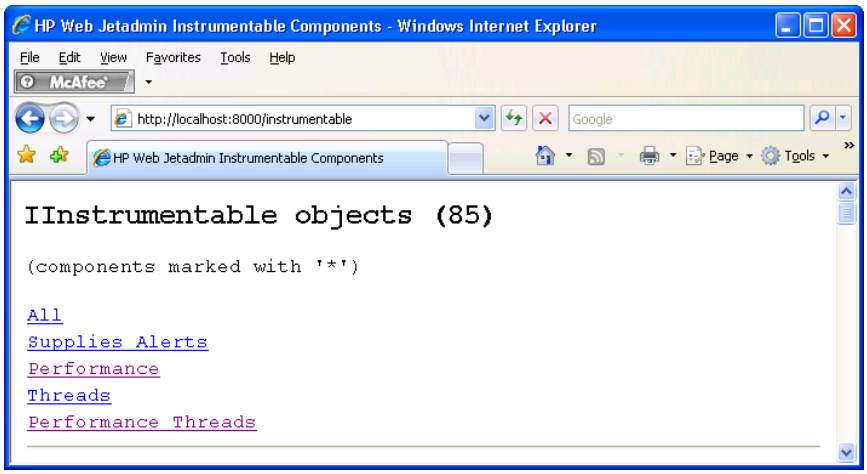
C:\Windows\ServiceProfiles\NetworkService\AppData\Local\HP Inc\HPWebJetadmin\WjaService\config

HP recommends monitoring the performance threads page under the **Instrumentable** page as each default value is updated. Look for large numbers of queued items where running is equal to the maximum. Increasing these values might increase performance and cause other threadpools to back up.

MOAB Instrumentable page

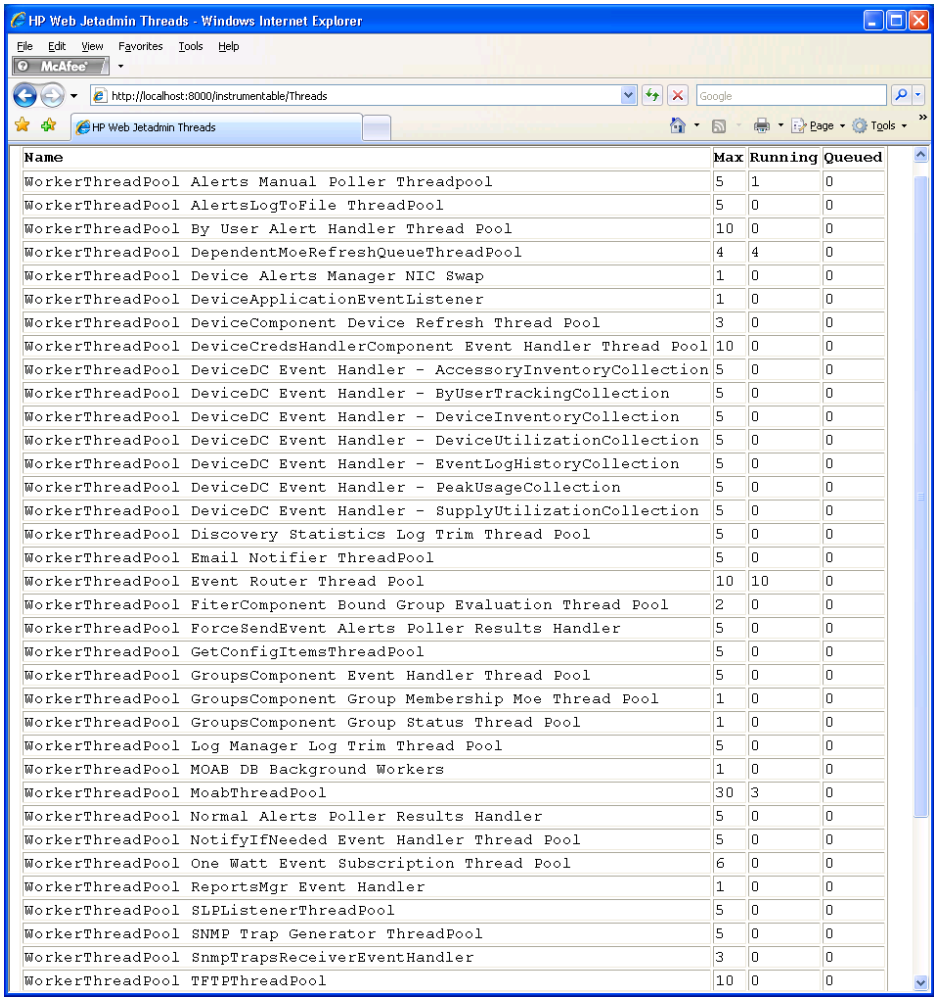
To understand threadpools, one must first understand a critical piece of architecture called Managed Object Abstraction (MOAB) in HP Web Jetadmin. After nodes are discovered during a discovery, MOAB further processes the nodes to determine if they are devices. MOAB is more or less responsible for getting and setting information on devices. MOAB manages device communication, trying to reduce it where possible by keeping a local cache of data and thresholds for how fresh the data should be. MOAB also manages grouping requests to devices. MOEs are the individual requests that MOAB issues.

When MOAB is running slow, the first place to look is on the **Instrumentable** page found by adding /instrumentable to the HP Web Jetadmin URL in a browser.



The purpose of using this page is to look for backed up threads.

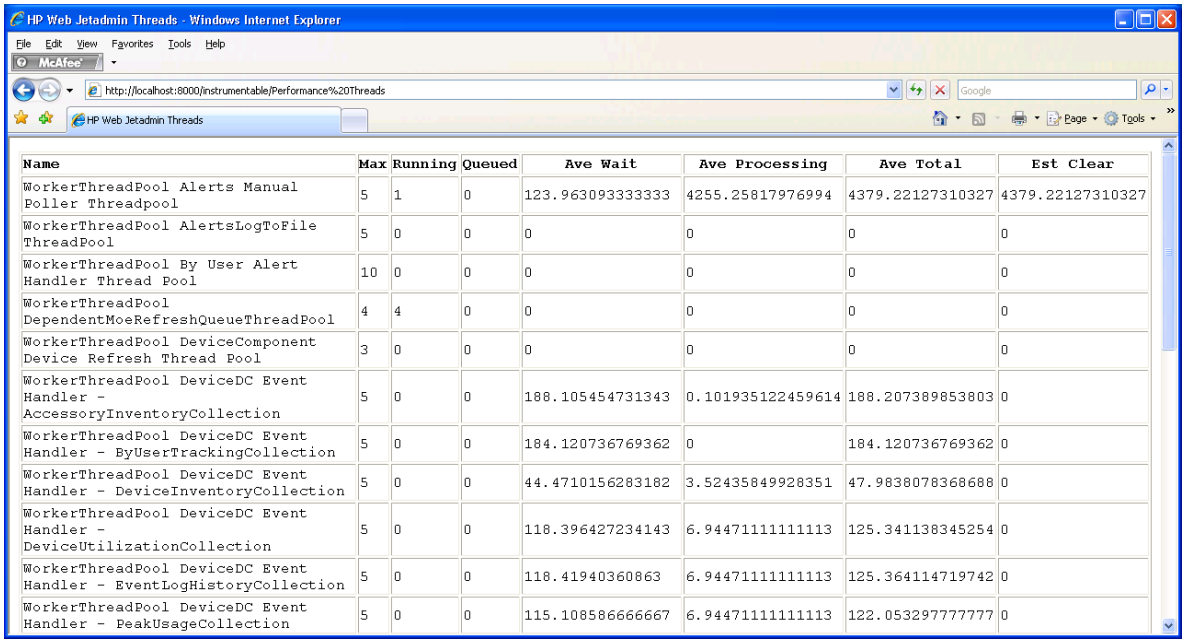
Under the Threads link, if the WorkerThreadPool MoabThreadPool line shows that the number of queued items is not decreasing, MOAB is getting too many requests to keep up.



Name	Max	Running	Queued
WorkerThreadPool Alerts Manual Poller Threadpool	5	1	0
WorkerThreadPool AlertsLogToFile ThreadPool	5	0	0
WorkerThreadPool By User Alert Handler Thread Pool	10	0	0
WorkerThreadPool DependentMoeRefreshQueueThreadPool	4	4	0
WorkerThreadPool Device Alerts Manager NIC Swap	1	0	0
WorkerThreadPool DeviceApplicationEventListener	1	0	0
WorkerThreadPool DeviceComponent Device Refresh Thread Pool	3	0	0
WorkerThreadPool DeviceCredsHandlerComponent Event Handler Thread Pool	10	0	0
WorkerThreadPool DeviceDC Event Handler - AccessoryInventoryCollection	5	0	0
WorkerThreadPool DeviceDC Event Handler - ByUserTrackingCollection	5	0	0
WorkerThreadPool DeviceDC Event Handler - DeviceInventoryCollection	5	0	0
WorkerThreadPool DeviceDC Event Handler - DeviceUtilizationCollection	5	0	0
WorkerThreadPool DeviceDC Event Handler - EventLogHistoryCollection	5	0	0
WorkerThreadPool DeviceDC Event Handler - PeakUsageCollection	5	0	0
WorkerThreadPool DeviceDC Event Handler - SupplyUtilizationCollection	5	0	0
WorkerThreadPool Discovery Statistics Log Trim Thread Pool	5	0	0
WorkerThreadPool Email Notifier ThreadPool	5	0	0
WorkerThreadPool Event Router Thread Pool	10	10	0
WorkerThreadPool FiterComponent Bound Group Evaluation Thread Pool	2	0	0
WorkerThreadPool ForceSendEvent Alerts Poller Results Handler	5	0	0
WorkerThreadPool GetConfigItemsThreadPool	5	0	0
WorkerThreadPool GroupsComponent Event Handler Thread Pool	5	0	0
WorkerThreadPool GroupsComponent Group Membership Moe Thread Pool	1	0	0
WorkerThreadPool GroupsComponent Group Status Thread Pool	1	0	0
WorkerThreadPool Log Manager Log Trim Thread Pool	5	0	0
WorkerThreadPool MOAB DB Background Workers	1	0	0
WorkerThreadPool MoabThreadPool	30	3	0
WorkerThreadPool Normal Alerts Poller Results Handler	5	0	0
WorkerThreadPool NotifyIfNeeded Event Handler Thread Pool	5	0	0
WorkerThreadPool One Watt Event Subscription Thread Pool	6	0	0
WorkerThreadPool ReportsMgr Event Handler	1	0	0
WorkerThreadPool SLPListenerThreadPool	5	0	0
WorkerThreadPool SNMP Trap Generator ThreadPool	5	0	0
WorkerThreadPool SnmpTrapsReceiverEventHandler	3	0	0
WorkerThreadPool TFTPThreadPool	10	0	0

The Instrumentable line indicates if there are queued items that are less than or equal to the thread count. The key is to determine if there are more items queued than the Max thread count consistently over time. The values are pretty meaningless for debugging if they are viewed for a given point in time. It is when one of those queued values stays high over time that there might be value in adjusting the threadpool values. HP Web Jetadmin adds quite a bit of work to many of the threadpools/queues, so the queued items might be 10 or more times the number of threads. The threads then do the work and return to zero before the next amount of work is typically added. It is when there are always too many queued items or the queued items are constantly growing that there should be a concern.

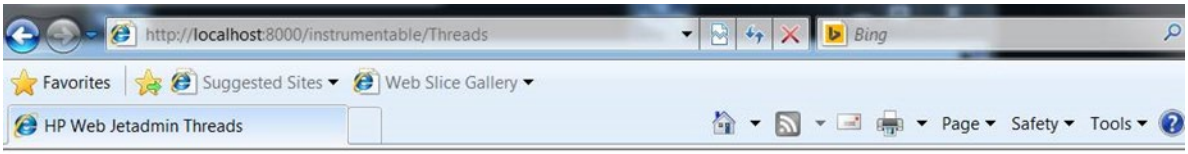
Another area to check is the Performance Threads link. If any of the categories have a large average wait time, perhaps increasing that thread count will help.



Name	Max	Running	Queued	Ave Wait	Ave Processing	Ave Total	Est Clear
WorkerThreadPool Alerts Manual Poller Threadpool	5	1	0	123.963093333333	4255.25817976994	4379.22127310327	4379.22127310327
WorkerThreadPool AlertsLogToFile ThreadPool	5	0	0	0	0	0	0
WorkerThreadPool By User Alert Handler Thread Pool	10	0	0	0	0	0	0
WorkerThreadPool DependentMoeRefreshQueueThreadPool	4	4	0	0	0	0	0
WorkerThreadPool DeviceComponent Device Refresh Thread Pool	3	0	0	0	0	0	0
WorkerThreadPool DeviceDC Event Handler - AccessoryInventoryCollection	5	0	0	188.105454731343	0.101935122459614	188.207389853803	0
WorkerThreadPool DeviceDC Event Handler - ByUserTrackingCollection	5	0	0	184.120736769362	0	184.120736769362	0
WorkerThreadPool DeviceDC Event Handler - DeviceInventoryCollection	5	0	0	44.4710156283182	3.52435849928351	47.9838078368688	0
WorkerThreadPool DeviceDC Event Handler - DeviceUtilizationCollection	5	0	0	118.396427234143	6.94471111111113	125.341138345254	0
WorkerThreadPool DeviceDC Event Handler - EventLogHistoryCollection	5	0	0	118.41940360863	6.94471111111113	125.364114719742	0
WorkerThreadPool DeviceDC Event Handler - PeakUsageCollection	5	0	0	115.108586666667	6.94471111111113	122.053297777777	0

Batch processing

When a configuration task is started, a WorkerThreadPool DeviceConfigTask-Threadpool is started with a unique number. This threadpool is only visible on the Instrumentable page under Threads as long as the configuration task is running. After the configuration is finished, the threadpool is no longer available. In the following screenshot, you can see the DeviceConfigTask-threadpool with the number 83c212ao:



Name	Max	Running	Queued
WorkerThreadPool Alerts Manual Poller Threadpool	7	1	0
WorkerThreadPool AlertsLogToFile ThreadPool	5	0	0
WorkerThreadPool By User Alert Handler Thread Pool	10	0	0
WorkerThreadPool DependentMoeRefreshQueueThreadPool	4	4	0
WorkerThreadPool Device Alerts Manager NIC Swap	1	0	0
WorkerThreadPool DeviceApplicationEventListener	1	0	0
WorkerThreadPool DeviceComponent Device Refresh Thread Pool	3	0	0
WorkerThreadPool DeviceConfigTask-Threadpool-83c212a0	12	10	0

Starting with 10.4 SR6, HP Web Jetadmin uses a maximum of 12 threads for batch processing. If a template is applied to 30 devices, HP Web Jetadmin simultaneously applies the template to 12 devices. In earlier versions of HP Web Jetadmin, this was limited to 5 threads simultaneously.

If more than 12 devices must be served simultaneously, several batch processes must be started because each batch process uses a maximum of five threads. When using autogrouping, the associated policies are applied up to a maximum of 12 devices simultaneously.

However, this requires that the maximum number of concurrent threads (the threads of all configuration tasks combined) must be increased as well. The maximum number of concurrent threads can be changed in the PerfomanceTuning.config.xml file. When HP Web Jetadmin is running as network service, the file is located in the following directory:

C:\Windows\ServiceProfiles\NetworkService\AppData\Local\HP Inc\HPWebJetadmin\WjaService\config

When HP Web Jetadmin is running as a domain user, HP Web Jetadmin uses the file from the following corresponding user directory:

C:\Users\user_name\AppData\Local\HP Inc\HPWebJetadmin\WjaService\config

In the following section, the value of 10 for MoabDevice.MaxConcurrentThreadsInResolution must be increased:

```
<property name="MoabDevice.MaxConcurrentThreadsInResolution">
  <type>HP.Imaging.Wjp.Sdk.Core.Framework.ConfigurationItemString
  </type>
  <value>10</value>
</property>
```

If the maximum number of concurrent threads is increased, the total number of threads (MoabThreadPoolSize) might also need to be increased. The default value is 30 and can be changed in the following section of the PerformanceTuning.config.xml file:

```
<property name="MoabDevice.MoabThreadPoolSize">
  <type>HP.Imaging.Wjp.Sdk.Core.Framework.ConfigurationItemString
  </type>
  <value>30</value>
</property>
```

Adjust the threadpool values

If you view the performance threads page and nothing appears to be backed up, increasing the threadpools might not help much. If something is backed up, increasing the threadpools might decrease the CPU load if items are backing up behind those threadpools because a larger number of threads allows more work to finish.

The default values that HP Web Jetadmin provides in the PerformanceTuning.config.xml configuration file were tested and produced a performance improvement when the values were increased.

IMPORTANT: If a non-default value is required, any settings that are not in the PerformanceTuning.config.xml file must be added. For example, if the MoabDevice.MaxThreadPercentageUsedDuringStartup setting is not in the file, add the following XML property to the file with the appropriate value:

```
<property name=" MoabDevice.MaxThreadPercentageUsedDuringStartup ">
  <type>HP.Imaging.Wjp.Sdk.Core.Framework.ConfigurationItemString</type>
  <value>30</value>
</property>
```

The following table provides the default values for each settings.

Name in XML file *Name on Instrumentable page*	Default value	Description
ByUserTrackingCollection.Threads *WorkerThreadPool By User Alert Handler Thread Pool*	10	Increasing this value might help when performing By User data collections on many devices. This is limited by the MOAB threads.
DavAlertL2FComponent.Threads *WorkerThreadPool AlertsLogToFile ThreadPool*	5	Increasing this value might help when Alerts Log to File is used for large numbers of alert subscriptions.
DavEmailNotifierComponent.Threads *WorkerThreadPool Email Notifier ThreadPool*	5	Increasing this value might help when the email alert notifier is used for large numbers of alert subscriptions.
DavSnmpTrapGenerator.Threads *WorkerThreadPool SNMP Trap Generator ThreadPool*	5	Increasing this value might help when the SNMP Trap Generator notifier is used for large numbers of alert subscriptions.
DeviceComponent.BackgroundRefreshThreads *WorkerThreadPool DeviceComponent Device Refresh Thread Pool*	3	The number of threads used for handling device inactivated and re-activated events. Increasing this value probably will not help performance.
DeviceConfigComponent.Threads *WorkerThreadPool GetConfigItemsThreadPool*	5	Increasing this value might help when device configurations are being performed on many devices. This is limited by the MOAB threads.
DeviceDC Event Handler - AccessoryInventoryCollection.ThreadCount *WorkerThreadPool DeviceDC Event Handler – AccessoryInventoryCollection*	5	Increasing this value might help when accessory inventory data is being collected on many devices. Only used to handle incoming events.
DeviceDC Event Handler - ByUserTrackingCollection.ThreadCount *WorkerThreadPool DeviceDC Event Handler – ByUserTrackingCollection*	5	Increasing this value might help when By User data is being collected on many devices. Only used to handle incoming events.
DeviceDC Event Handler - DeviceInventoryCollection.ThreadCount *WorkerThreadPool DeviceDC Event Handler – DeviceInventoryCollection*	5	Increasing this value might help when device inventory data is being collected on many devices. Only used to handle incoming events.
DeviceDC Event Handler –DeviceUtilizationCollection.ThreadCount *WorkerThreadPool DeviceDC Event Handler – DeviceUtilizationCollection*	5	Increasing this value might help when device utilization data is being collected on many devices. Only used to handle incoming events.
DeviceDC Event Handler - EventLogHistoryCollection.ThreadCount *WorkerThreadPool DeviceDC Event Handler – EventLogHistoryCollection*	5	Increasing this value might help when event log data is being collected on many devices. Only used to handle incoming events.
DeviceDC Event Handler - PeakUsageCollection.ThreadCount *WorkerThreadPool DeviceDC Event Handler – PeakUsageCollection*	5	Increasing this value might help when peak usage data is being collected on many devices. Only used to handle incoming events.

Name in XML file *Name on Instrumentable page*	Default value	Description
DeviceDC Event Handler - SupplyUtilizationCollection.ThreadCount *WorkerThreadPool DeviceDC Event Handler - SupplyUtilizationCollection*	5	Increasing this value might help when supply utilization data is being collected on many devices. Only used to handle incoming events.
DevicePollingReceiver.ForceSendEvent Alerts Poller Results HandlerThreads *WorkerThreadPool ForceSendEvent Alerts Poller Results Handler*	5	Increasing this value might help performance during alert subscription and might also help when receiving traps from devices.
DevicePollingReceiver.Normal Alerts Poller Results HandlerThreads *WorkerThreadPool Normal Alerts Poller Results Handler*	5	Increasing this value might help performance for background alerts polling.
DevicePollingReceiver.NumManualPollerThreads *WorkerThreadPool Alerts Manual Poller Threadpool*	5	Increasing this value might help when many alerts are polling-only alerts and might help the subscribe time for alerts.
DiscoveryManager.NumHostIpResolverThreads *WorkerThreadPool ResolverMethodIpHostnameThread*	30	Increasing this value might help reduce discovery times.
DiscoveryManager.NumResolverThreads *WorkerThreadPool ResolverMethodThread*	10	Increasing this value might help reduce discovery times.
DiscoveryManager.SLPListenerThreadPool *WorkerThreadPool SLPListenerThreadPool*	5	Increasing this value might help with SLP listen discoveries.
DiscoveryMethodSpecifiedAddress.PingThreadCount *WorkerThreadPool IpRangePingThreadPool1* **NOTE:** This threadpool is only visible during the discovery process.	30	Increasing this value might help when specified address or subnet range is being used for discovery.
FilterComponent.Threads *WorkerThreadPool DeviceComponent Device Refresh Thread Pool*	2	The threadpool used to evaluate auto-group membership. Increasing this value might help when there are a large number of auto-groups defined.
GroupsComponent.EventHandlerThreads *WorkerThreadPool GroupsComponent Event Handler Thread Pool*	5	Event handler for groups component. These refer to internal HP Web Jetadmin server events, such as GroupMembershipChangedEvent (someone removed a device from the group), GroupAttributeChangedEvent, EventMoabDataCacheChange (device data changed), EventMoabDeviceActivated (new device in the system).
DavAlertL2FComponent.Threads *WorkerThreadPool AlertsLogToFile ThreadPool*	5	

Name in XML file *Name on Instrumentable page*	Default value	Description
MoabDevice.DependentMoeRefreshQueueThreads *WorkerThreadPool DependentMoeRefreshQueueThreadPool*	4	Whenever a MOE that other MOEs have a dependency on is changed, the threadpool is used to update the MOEs that have the dependency in the background after sending the event saying the original MOE has changed. This is used to ripple changes through the system for MOEs that depend on each other. As an example, the SupplyBlackCartridge MOE has dependencies on the SupplyBlackTonerCartridge and SupplyBlackInkCartridge MOEs.
MoabDevice.LockBucketSize	23	This value determines how many DeviceIDs can be locked at one time when reading/writing data to the database for a particular device.
MoabDevice.MaxConcurrentThreadsInResolution *WorkItemGroup MoabDeviceListResolution*	10	Increasing this value might help the MOAB portion of discovery resolution (shortening overall discovery times).
MoabDevice.MaxThreadPercentageUsedDuringDiscovery	50	The percentage of the overall MoabThreadPoolSize that can be used for discovery.
MoabDevice.MaxThreadPercentageUsedDuringStartup	25	The percentage of the overall MoabThreadPoolSize that can be used during startup.
MoabDevice.MaxThreadPercentageUsedForPolling	25	The percentage of the overall MoabThreadPoolSize that can be used for polling. Only the MOAB pollers are affected. This setting affects all MOAB operations (best bang for the buck). Any operation that requires device communication (either GET or SET) benefit.
MoabDevice.MoabThreadPoolSize *WorkerThreadPool MoabThreadPool*	30	Increasing this value might help improve performance when performing actions across multiple devices.
NotificationManager.NumNotificationThreads *WorkerThreadPool NotifyIfNeeded Event Handler Thread Pool*	6	Used for processing and sending alerts. Increasing this value might help when there are many active alert subscriptions.
OxpmComponent.AlertThreads *WorkerThreadPool WS Notification ThreadPool*	5	Increasing this value might help when the OXPm client is being used for alert notification for many devices.
SnmpTrapsReceiver.TrapsEventHandlerThreadCount *WorkerThreadPool SnmpTrapsReceiverEventHandler*	3	Increasing this value might help improve performance if the HP Web Jetadmin instance receives a large number of incoming traps (by user data collection, alert subscriptions).
Tftp.TFTPThreadPoolCount *WorkerThreadPool TFTPThreadPool*	10	Increasing this value might help when firmware upgrades are being performed on many HP Jetdirect NICs.
WebServer.Threads *WorkerThreadPool WebServer*	5	Increasing this value might help when many HP Web Jetadmin clients are being used simultaneously.

Hardcoded threadpools

Some threadpools values cannot be edited. Examples of threadpools that cannot be changed:

- WorkItemGroups GetDeviceUidThreadPool (hardcoded to 10)
- WorkerThreadPool DC Task Event Handler – DeviceInventoryCollection (hardcoded to 1)
- WorkerThreadPool DC Task Event Handler – DeviceUtilizationCollection (hardcoded to 1)
- WorkerThreadPool DC Task Event Handler – AccessoryInventoryCollection (hardcoded to 1)
- WorkerThreadPool ReportsMgr Event Handler (hardcoded to 1)

Example of updating the PerformanceTuning.config.xml configuration file

When looking at the instrumentable data, you might see something similar to the following:

WorkerThreadPool MoabThreadPool	30	8	0
WorkerThreadPool Normal Alerts Poller Results Handler	5	0	0
WorkerThreadPool NotifyIfNeeded Event Handler Thread Pool	5	0	0
WorkerThreadPool One Watt Event Subscription Thread Pool	6	0	0
WorkerThreadPool ReportsMgr Event Handler	1	0	0
WorkerThreadPool SLPListenerThreadPool	5	0	0
WorkerThreadPool SNMP Trap Generator ThreadPool	5	0	0
WorkerThreadPool TFTPThreadPool	10	0	0
WorkerThreadPool WebServer	5	2	0
WorkerThreadPool WS Notification ThreadPool	5	0	0
WorkItemGroup GetDeviceUuidThreadPool	10	0	0
WorkItemGroup MoabDeviceListResolution	20	1	0
WorkItemGroup MoabDiscovery	15	0	0
WorkItemGroup MoabPolling	7	7	14
WorkItemGroup MoabStartup	7	0	0

In this example, there are 14 threads waiting to execute in the MoabPolling threadpool. The following settings define this threadpool:

- MoabDevice.MoabThreadPoolSize

  This example has a value of 30, which is the default.

- MoabDevice.MaxThreadPercentageUsedForPolling

  This example has a value of 25, which is the default.

To prevent the MoabPolling threads from being queued, the number of available threads for MoabPolling must be increased. In this example, the number of available threads is 7, which is 25% of the total of 30 defined for the MoabDevice.MoabThreadPoolSize setting.

Increasing the MoabDevice.MaxThreadPercentageUsedForPolling setting decreases the number of available threads for other operations. Therefore, it is better to increase the MoabDevice.MoabThreadPoolSize setting, possibly combined with a higher percentage for the MoabDevice.MaxThreadPercentageUsedForPolling setting.

For example, if you change the MoabDevice.MoabThreadPoolSize setting to 60 and increase the percentage to 40%, there will be 24 threads for polling. Other HP Web Jetadmin operations will also benefit from an increased threadpool size.

However, the underlying hardware must be able to handle this change. This change must be tested. If the overall performance decreases, lower values must be used.

Increase the firmware upgrade performance

Under **Tools** > **Options** > **Device Management** > **Firmware**, the **Maximum concurrent upgrades** setting can be used to specify that multiple threads perform firmware upgrades in a parallel fashion rather than serially. The default setting is to upgrade eight devices simultaneously. In HP Web Jetadmin 10.3 SR 8 and later, this can be increased to 50 devices in a single task. Firmware installation is faster when multiple devices are upgraded at the same time, but more network traffic is generated. Operating limitations might also restrict how many devices can receive upgrades simultaneously. This setting applies to the number of devices inside a single task or multiple tasks.

WARNING: Increasing the **Maximum concurrent upgrades** setting to a value above 10 should only be done on powerful servers that have limited tasks other than the firmware upgrades.

NOTE: If the **Maximum concurrent upgrades** setting is set to 11 or higher, the FirmwareUpgradeThreadsize setting in the PerfomanceTuning.config.xml file must also be changed to match the value specified for the **Maximum concurrent upgrades** setting. For instructions on changing the FirmwareUpgradeThreadsize section, see Increase the firmware upgrade threadpool size on page 26.

If it is still taking too long to perform batch firmware upgrades, one additional technique can be used to significantly decrease the time it takes to complete a batch upgrade process. There is a trade-off in the accuracy of the reporting status, but the gains in time can be very significant.

The normal process for upgrading printer firmware involves opening a thread, sending the firmware file through either a port 9100 connection or a Web Services connection, monitoring the upgrade status on the device until it completes its power cycle and reports that the new firmware is in place, and then closing the thread. If the maximum thread count is set to 10, after all 10 threads complete the process, 10 more threads are attempted. This technique allows HP Web Jetadmin to very accurately report success or failure. However, the constant status checking after delivering the firmware file can take upwards of 90% of the total time to complete the process.

HP Web Jetadmin 10.3 SR2 and later allows for manipulating the FirmwareUpgrade.config.xml file to bypass the status-checking portion of the upgrade process. This means that HP Web Jetadmin delivers the firmware file to the device and reports success after the file is delivered in its entirety. This can eliminate potentially 90% of the transaction time per device, meaning many more devices are completed in a fraction of the previous time when status checking was being employed.

Use the following steps to bypass the status-checking portion of the upgrade process:

1. Use a text editor to open the existing FirmwareUpgrade.config.xml file. This file is available in the following directory on the HP Web Jetadmin host system:

   C:\Windows\ServiceProfiles\NetworkService\AppData\Local\HP Inc\HPWebJetadmin\WjaService\config\

2. Find the following settings:

```
<property name="IsJetdirectMonitorRequired">
  <type>HP.Imaging.Wjp.Sdk.Core.Framework.ConfigurationItemString
  </type>
  <value>True</value>
</property
<property name="IsPrinterMonitorRequired">
  <type>HP.Imaging.Wjp.Sdk.Core.Framework.ConfigurationItemString
  </type>
  <value>True</value>
</property>
```

3. Change the values of these settings from True to False to bypass status checking for HP Jetdirect and/or printer firmware and significantly speed up batch upgrades.

4. Save the file.

5. Restart the HP Web Jetadmin service (HPWJA Service). First notify all the users logged in to the system about the service restart and ensure that active HP Web Jetadmin tasks are stopped or rescheduled to avoid problems with the restart.

NOTE: Use the standard XML format when creating this file. Use a simple text editor such as Microsoft® Notepad. Line breaks should occur only after the appropriate tag at the end of a statement. Documentation viewers might add what appear to be line breaks in the XML content shown above.

HP Web Jetadmin now reports success only after the file deliveries are complete. To check the completion status on the devices, check the **Device Firmware Version** and **Device Firmware Date** columns to determine if the firmware upgrade was successful.

Increase the firmware upgrade threadpool size

The maximum number of concurrent firmware upgrades is defined by the firmware upgrade threadpool size. The value specified for the firmware upgrade threadpool size must match the value specified for the **Maximum concurrent upgrades** setting.

Use the following steps to increase the firmware upgrade threadpool size:

1. Make a backup copy of the PerfomanceTuning.config.xml file. This file is available in the following directory on the HP Web Jetadmin server:

   C:\Windows\ServiceProfiles\NetworkService\AppData\Local\HP Inc\HPWebJetadmin\WjaService\config

2. Use Notepad or a similar editor to open the PerfomanceTuning.config.xml file.

3. Change the <value>xx</value> entry in the FirmwareUpgradeThreadsize section. For example, specify 30.

   ```
   <property name="FirmwareComponent.FirmwareUpgradeThreadsize">
     <type>HP.Imaging.Wjp.Sdk.Core.Framework.ConfigurationItemString
     </type>
     <value>xx</value>
   </property>
   ```

   NOTE: The FirmwareUpgradeThreadsize section is available in the PerfomanceTuning.config.xml file only for new installations of HP Web Jetadmin 10.3 SR8 and later. For installations that have been upgraded to HP Web Jetadmin 10.3 SR8 or later, this section must be manually added to the PerfomanceTuning.config.xml file.

4. Close and save the file.

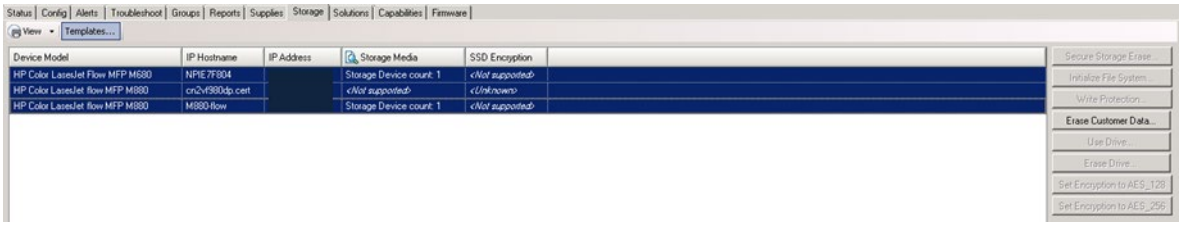
5. Restart the HP Web Jetadmin service (HPWJAService).

6. Launch the HP Web Jetadmin client.

7. Change the value of the maximum number of concurrent firmware upgrades.

Increase the number of threads for EraseCustomerDataTask

To select one or more devices, go to the **Config** tab and select **Erase Customer Data**.



When the process starts, it creates a threadpool (EraseCustomerDataTask) with a maximum of 5 threads hardcoded. The number of threads for this operation is hardcoded, and cannot be changed in the PerformanceTuningConfig.xml. However, it is possible to create multiple threadpools for more parallel processing. For every scheduled task for the Erase Customer Data operation, HP Web Jetadmin creates a new threadpool.

When two tasks are created, two threadpools are created for EraseCustomerDataTask.

WorkerThreadPool DoDownloadThreadPool	10	0	0
WorkerThreadPool Email Notifier ThreadPool	5	0	0
WorkerThreadPool EraseCustomerDataTask-Threadpool-2f8aeaef	5	5	0
WorkerThreadPool EraseCustomerDataTask-Threadpool-a0de1d1f	5	5	0

By splitting up tasks that have a hardcoded thread limit in two or more tasks, more threadpools are created, each using the hardcoded threads limit per threadpool. When creating two tasks instead of one for the Erase Customer Data operation for a large number of devices, the processing time is cut in half, assuming the server has enough processing power. It is possible to create even more parallel tasks to reduce the overall processing time further.

Decrease the HP Web Jetadmin server startup time

HP Web Jetadmin also provides an option to increase the number of startup threads that it can use with the StartupThreads parameter in the server.config.xml file.

WARNING: By default, the StartupThreads setting is twice the number of logical cores. If the StartupThreads setting is increased, HP Web Jetadmin might not start if there are many scheduled tasks. Because of the high number of StartupThreads, starting the task manager and scheduled tasks at the same time might cause a circular dependency. This error message can be seen in the diagnostics page (http://Wja IP/diagnostics). If this happens, reduce the number of StartupThreads.

The server.config.xml file is available in the following directory:

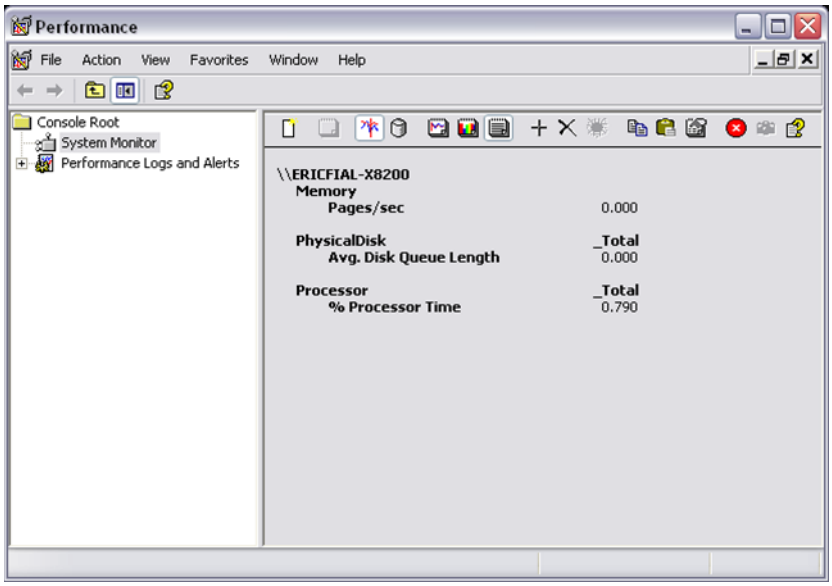
C:\Program Files\HP Inc\Web Jetadmin 10\WJABackupRestore\Settings\WjaService\config

Summary

Sizing an HP Web Jetadmin installation and optimizing it for performance can be challenging. The hardware where HP Web Jetadmin is installed, enabled features, and number of devices directly influences the performance of an HP Web Jetadmin installation. This white paper discusses techniques for how to potentially improve the performance of many tasks in HP Web Jetadmin by manipulating the number of threads used for each task. Configuring the Windows Paging File Size to provide reliable support for very large servers is also crucial to improving performance and eliminating memory out conditions (crashes).

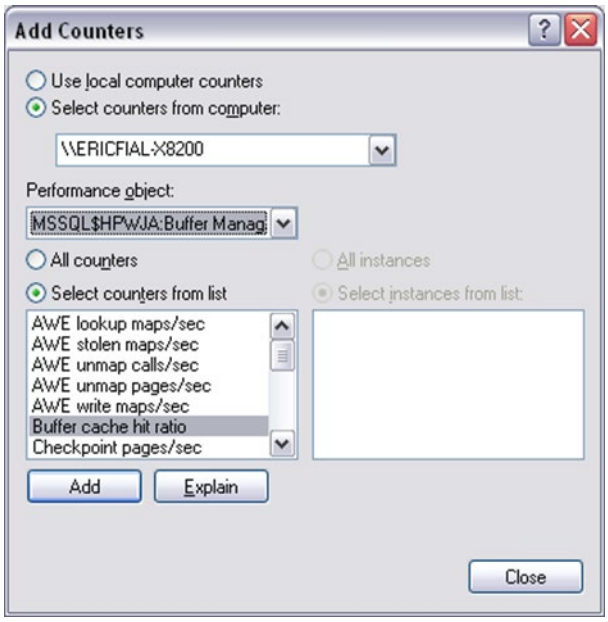
Appendix A–Check memory consumption

1. Run Perfmon (**Start**, **Run**, **perfmon**, **OK**), and then switch to **View Report** (Ctrl+R).



   Having an Avg. Disk Queue Length that is higher than 2 for anything longer than a short period of time is the first sign that more memory might be needed.

2. Right-click and select **Add Counters**. From the **Performance object** list, select **MSSQL$HPWJA:Buffer Manager**.



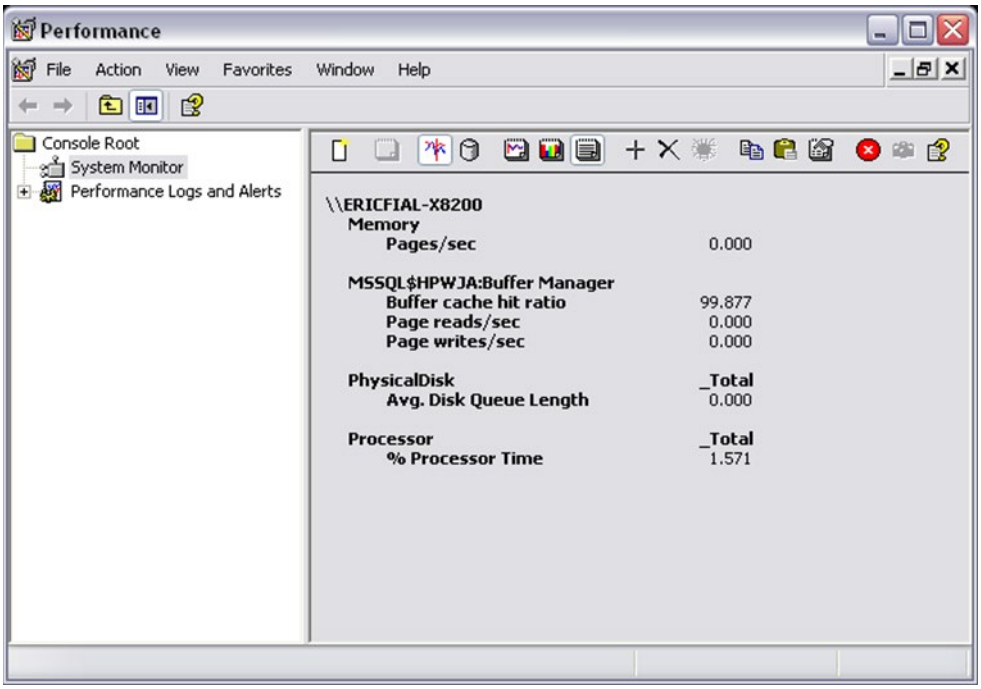
3. From the **Select counters from list** section, add the following objects: **Buffer cache hit ratio**, **Page reads/sec**, **and Page writes/sec**.

4. Click **Close**.

If any of the following are true, more memory might be needed:

- Buffer cache hit ratio < 97%
- Page reads/sec > 10 for an extended period of time
- Page writes/sec > 10 for an extended period of time

These numbers jump from time to time during normal operations. However, if they are consistently out of the normal, it usually indicates that the database is spending time moving data out of its cache and might benefit from more memory. Typically, when a machine is behaving badly, the Page reads/sec and/or the Page writes/sec are in the 100s and the Avg. Disk Queue Length is in double or triple digits.



hp.com/go/getconnected

Current HP driver, support, and security alerts
delivered directly to your desktop

pdf_6881505_en-US-1, May 2023