White paper

BMC Remedy AR System Server 7.6

Performance Tuning for Business Service Management

February 2011

ACTIVATE BUSINESS WITH THE POWER OF I.T.™

www.bmc.com

Contacting BMC Software

You can access the BMC Software website at http://www.bmc.com. From this website, you can obtain information about the company, its products, corporate offices, special events, and career opportunities.

United States and Canada

Address	BMC SOFTWARE INC 2101 CITYWEST BLVD HOUSTON TX 77042-2827 USA	Telephone	713 918 8800 or 800 841 2031	Fax	713 918 8000

Outside United States and Canada

Telephone	(01) 713 918 8800	Fax	(01) 713 918 8000

If you have comments or suggestions about this documentation, contact Information Design and Development by email at doc_feedback@bmc.com.

Restricted Rights Legend

Customer Support

You can obtain technical support by using the Support page on the BMC Software website or by contacting Customer Support by telephone or email. To expedite your inquiry, please see "Before Contacting BMC Software."

Support website

You can obtain technical support from BMC Software 24 hours a day, 7 days a week at http://www.bmc.com/support_home. From this website, you can:

- Read overviews about support services and programs that BMC Software offers.
- Find the most current information about BMC Software products.
- Search a database for problems similar to yours and possible solutions.
- Order or download product documentation.
- Report a problem or ask a question.
- Subscribe to receive email notices when new product versions are released.
- Find worldwide BMC Software support center locations and contact information, including email addresses, fax numbers, and telephone numbers.

Support by telephone or email

In the United States and Canada, if you need technical support and do not have access to the Web, call 800 537 1813 or send an email message to customer_support@bmc.com. (In the Subject line, enter SupID:<*yourSupportContractID*>, such as SupID:12345.) Outside the United States and Canada, contact your local support center for assistance.

Before contacting BMC Software

Have the following information available so that Customer Support can begin working on your issue immediately:

- Product information
  - Product name
  - Product version (release number)
  - License number and password (trial or permanent)
- Operating system and environment information
  - Machine type
  - Operating system type, version, and service pack
  - System hardware configuration
  - Serial numbers
  - Related software (database, application, and communication) including type, version, and service pack or maintenance level
- Sequence of events leading to the problem
- Commands and options that you used
- Messages received (and the time and date that you received them)
  - Product error messages
  - Messages from the operating system, such as file system full
  - Messages from related software

License key and password information

If you have a question about your license key or password, contact Customer Support through one of the following methods:

- E-mail customer_support@bmc.com. (In the Subject line, enter SupID:<*yourSupportContractID*>, such as SupID:12345.)
- In the United States and Canada, call 800 537 1813. Outside the United States and Canada, contact your local support center for assistance.
- Submit a new issue at http://www.bmc.com/support_home

Contents

White paper

Performance tuning for Business Server Management

About performance tuning for BSM

The Business Service Management (BSM) product suite can dramatically simplify an organization's IT environment by providing process automation and service management solutions. Though BMC software benchmarking greatly improve the performance of BSM products before they are released, understanding how best to tune your environment is critical.

This paper describes the best practices for BSM performance tuning. Environments and configurations for typical BMC enterprise customers are addressed, specifically the following:

- Evaluating the performance of a configuration running BSM applications
- Identifying potential bottlenecks that might cause performance issues
- Collecting and interpreting diagnostic data to identify the root cause of a performance issue

Product and configuration focus

Any BSM environment should contain an ITIL® -compliant product set, which includes:

- Configuration Management Database (CMDB)
- Automated discovery sources
- Service Impact Management and Service Level Management tools
- Functionality such as Incident, Problem, and Change Management.

The recommendations in this paper are based on tests conducted with products running on the BMC Remedy Action Request System (AR System) server, such as BMC Atrium CMDB, BMC Remedy IT Service Management Suite, and products such as BMC Service Impact Manager and BMC Configuration Discovery. However, the majority of the techniques discussed in this paper are generic best practices that can be applied to any product set.

To reduce the scope of BSM performance tuning, this document focuses primarily on the Oracle and Microsoft SQL Server databases and the Tomcat servlet engine.

AR System architecture

Understanding performance issues requires a basic understanding of the underlying application architecture. The following figure provides a simplified view of AR System architecture.

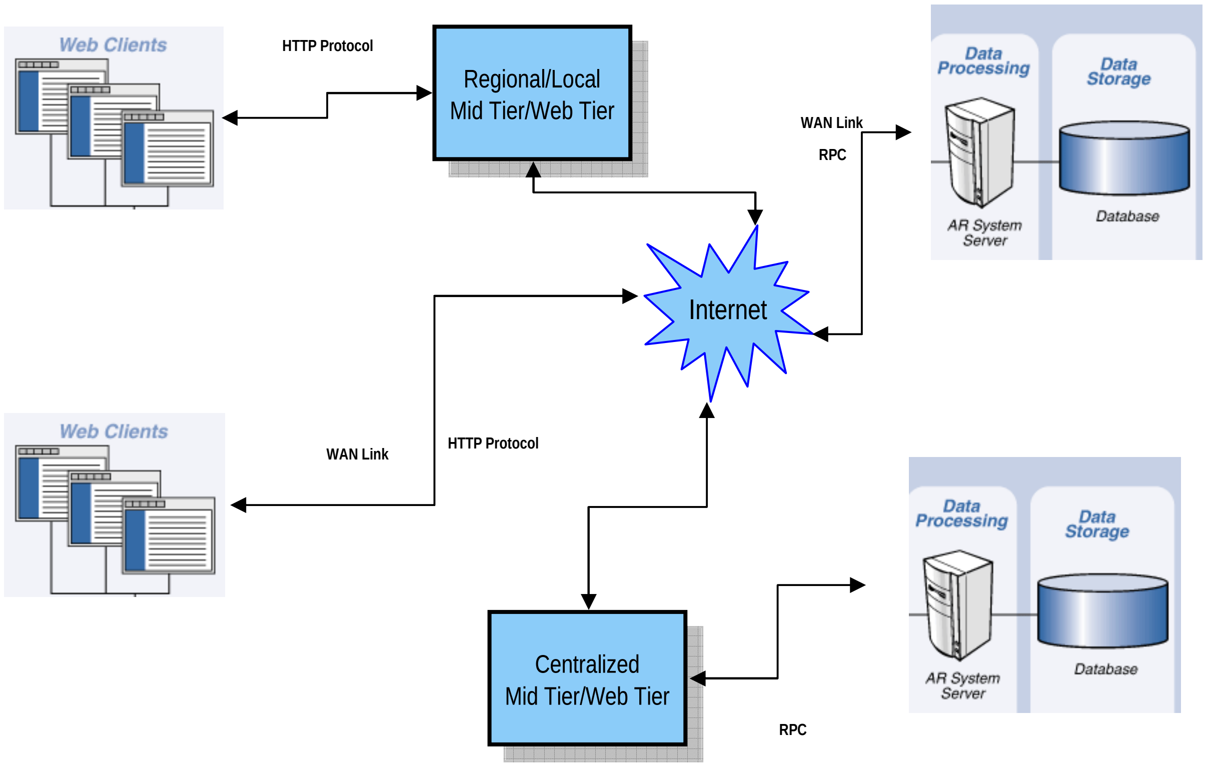


Figure 1: Simplified AR System architecture

AR System is implemented using a multitier architecture. At the backend, a database is used to store and retrieve data. AR System supports five databases: Oracle, SQL Server, DB2, Sybase and Informix. AR System server connects to the database and maintains a pool of persistent connections based on the thread settings as defined in the AR System configuration files, located at *`AR install dir`*`/conf/ar.cfg` or `ar.conf.`

The mid tier, a web application, connects to AR System server by using RPC protocol. An optional RPC connection pool can be maintained by the mid tier for optimal resource usage and performance. For a default installation, this RPC connection pool is on. The number of connections for an AR System server to which the mid tier connects is set to 80 by default, but can be configured using the web-based Mid Tier Configuration Tool. The mid tier can be located centrally, with the database and the AR System server, or regionally, which means close to end-users.

Performance considerations

Keep the following performance considerations in mind:

- Sizing database server for capacity is important as it may not be possible to scale DB tier horizontally unless you are using Oracle RAC.
- Database performance is highly dependent on fast IO subsystem. You must have an optimum I/O subsystem.
- Since the AR System server only caches metadata, it is highly dependent on the fast, reliable, low-latency connection to the database.

  BMC recommends a gigabit connection with close to zero latency between the AR System server and the database. Any additional hop, packet screener, or firewall has a negative impact on performance.
- For the AR System server tier, you can scale horizontally by configuring multiple AR System servers configured as a server group.
- For the web tier, you can scale horizontally by configuring multiple mid tiers connecting to the same AR System server or server group and by configuring a load balancer (or a reverse proxy) in front of the mid tiers to do the routing.
- BMC recommends having a dedicated environment for the BSM application. Performance is difficult to manage when multiple applications are running in the same environment.

Problem statement

Performance tuning initially requires a clear problem statement, including the following two common metrics:

- Throughput—Frequency of an operation over time, such as the number of tickets created in an hour, or the number of configuration items (CIs) processed in an hour.
- Response time—Seconds between a significant operation (for example, clicking Save to create an incident ticket) and the time that the operation is completed and system control is returned to the user. See “Collecting response time data” on page 9.

A problem statement should also include a list of recent changes that might have led to the current state. In production-related issues, change control or other sources can often provide a log of system changes that took place before a problem began. Such a change history might not point to the root cause of a problem, but provides valuable background data.

Collecting response time data

Response time might be described subjectively with a statement, such as “the system seems slow”. However, to effectively identify a problem, multiple users with different client configurations should manually time the operations.

To measure response time, collect event times from multiple users who are experiencing the problem. Capture various data points that impact performance to form a complete picture. Record the following data:

1. Client configuration and browser information for each user—Record the browser type and version. For more information, see “Optimum client configuration” on page 52.
2. Client location—Record the latency the user is experiencing at the application layer by using http-ping also along with tcp ping latency.
3. Timestamp for each timing test—Because caching effects might cause the first timing test to take longer than subsequent tests, make a separate note of these times for each user.

Figure 2 shows an example of a spreadsheet for capturing response time data. To instruct the testers exactly what to do and what to time, describe each test in detail. All testers should use a equivalent timing device, such as a stopwatch or similar tool.

Client configuration: (Number and speed of CPUs, RAM)					
Browser type and version					
Location					
TCP PING (msec) ping mid tier hostname (In case of regional mid tier, ping AR System server from the mid tier)					
HTTP PING (msec) http-ping to the mid tier login URL					
Use cases	**Response times**				
	Uncached access (*n.nn seconds*)	**Average of 3 cached accesses (*n.nn* seconds)**			
Incident Management (IM)					
Login					
Click IM Console					
Click New Incident					
Search Incident by Incident ID					
Search Incident by Customer Name					
Create (Save) Incident					
Modify Incident					
Change Management (CM)					
Click CM Console					
Click New Change					
Search Change by ID					
Create Change with Task					
Approve Change					

Figure 2: Sample spreadsheet for collecting response times

Best practices for problem statements

Use the following checklist to create a problem statement:

Characterize throughput or response time	☐ Before problem was observed ☐ After problem was observed
List any system changes that took place soon before the problem was observed	☐ Patch changes ☐ Configuration changes ☐ Workload changes ☐ User count changes
List observations	☐ Is the entire system slow or is the slow performance specific to particular user, particular location or particular transaction only? ☐ For global deployments, is the performance issue related to users accessing the system over WAN only? Is it related to specific site? ☐ Is the performance issue intermittent? If so, what is the frequency? When are issues occurring?

For a checklist that includes all performance tuning issues, see "Performance tuning checklist" on page 53.

Problem triage

Initial triage of performance problems involves observing resource consumption across the configurations. Resource profiles can point to the root cause of a problem, or they can help focus the search for a cause.

For each tier of your system, consumption of these resources should be analyzed:

- Central processing units (CPUs)—Include system time, user time, and I/O wait time. For benchmarks simulating online users, get CPU usage data for the computers driving the load. See "CPU consumption" on page 11.
- Memory—See "Memory consumption" on page 13.

CPU consumption

This section provides guidelines and best practices for CPU consumption.

Monitoring CPU consumption

CPU usage substantially affects system response time, usually in a nonlinear way. Figure 3 shows a typical response-time curve. Response time starts at a baseline and grows slowly while CPU usage is low. As CPU usage increases, the response-time slope becomes steeper until it is almost vertical.

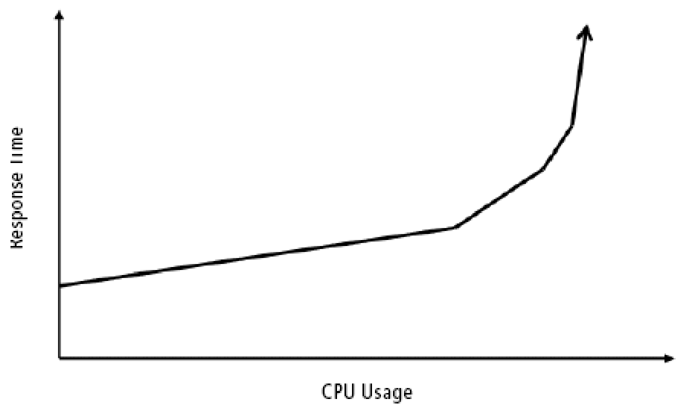


Figure 3: Typical response-time curve

Incremental response-time growth is expected until the significant response time increase in the curve is reached. The point at which the response time increase occurs is variable, but it typically occurs when CPU usage for at least one computer in the system is at least 60% and sometimes as high as 90%.

Running only one server at high load in a multi-server configuration can substantially affect response time. Favorable response time does not always occur when some servers in the configuration are largely idle.

If the sharp increase in response time in the configuration's response-time curve is similar to the bottleneck in Figure 3 but occurs while CPU usage on all systems is low, the consumption of a non-CPU resource is creating the system bottleneck.

CPU usage has several components:

- User time—Time consumed by applications performing useful activity on the system. Generally, user time should constitute the majority of CPU usage.
- System time—Time consumed by the operating system (OS) kernel for core processing. If system time is higher than 5%, an OS-level resource or locking problem exists, which is a typical memory management issue.
- I/O wait time—Time consumed waiting for a request to the I/O subsystem to return. If I/O wait time is substantial, the I/O subsystem cannot keep up with the CPU application processing. In this case, the system will likely have poor response time unless the I/O subsystem bandwidth is improved or the application’s data requirements are tuned. Generally, I/O concerns are relevant only on the database system of a multi-tiered configuration.

Typical tools used to track CPU usage are `perfmon` for Windows and `sar` for UNIX® and Linux®. In both cases, the collected data should be stored for later review. For UNIX systems, `top` can identify high-load processes, but its data is harder to store for review, so using the `ps` command might be a useful alternative. Monitoring relative CPU usage among running processes is also helpful, by using `top`.

Best practices for monitoring CPU consumption

Use the following checklist to monitor CPU consumption:

- ☐ Collect CPU usage data for each computer in the configuration.
- ☐ Make sure less than 70% of total CPU capacity is used on each computer in the configuration.
- ☐ Record which processes are consuming most of the CPU resources.
- ☐ Record whether I/O wait time is a substantial component of CPU usage. If so, the I/O subsystem is not keeping up with demand.

For a checklist that includes all performance tuning issues, see “Performance tuning checklist” on page 53.

Memory consumption

This section provides guidelines and best practices for memory consumption.

Monitoring memory consumption

Tracking memory consumption is not always straightforward because:

- Some systems assign unused memory to swap space or to other system resources.
- Memory allocation algorithms vary between operating systems.

Understanding approximately how much of a configuration’s physical memory is in use during application execution is important.

Modern operating systems always provide a virtual memory space that far exceeds the system’s physical memory. Thus, if an application consumes nearly all the system’s physical memory, processes in shared memory are written to disk to free up space for new memory requirements. This practice is informally called “swapping.” Although swapping can be useful, it degrades both CPU usage and response time. (Do not confuse process swapping with simple memory paging, which can be normal behavior and does not necessarily consume extra resources.)

The primary goal of evaluating memory consumption is to ensure that a system is not swapping (or close to swapping) and that an application can still allocate memory when needed. Because swapping increases CPU usage, the first symptom of running out of memory might be that CPU resources are fully used. Noticing how memory consumption grows over time is often useful. A process that continues to consume memory quickly might have a memory leak, which is likely to lead to swapping or failure.

When evaluating the performance impact of memory consumption with 32-bit BMC applications, consider whether the OS is 32-bit or 64 bit. For example:

- On 32-bit editions of Windows, applications have 4 GB of virtual address space available. This space is divided into 2 GB for the application and 2 GB for the kernel. However, you can use tuning features such as 4 GB tuning (4GT) and the `IMAGE_FILE_LARGE_ADDRESS_AWARE` value of the `LOADED_IMAGE` structure to reallocate 3 GB for the application and 1 GB for the kernel.
- On 64-bit editions of Windows, you can use the `IMAGE_FILE_LARGE_ADDRESS_AWARE` parameter to increase the 2 GB limit up to 4 GB for a 32-bit application. In addition, technology such as Physical Address Extension (PAE) can enable 32-bit Windows systems to use more than 4 GB of physical memory.

On UNIX and Linux systems, application memory management, including how shared memory is configured, differs from vendor to vendor. To learn how to use your system's virtual memory and shared memory management features to achieve the best performance for BMC applications, see your product documentation or consult your system administrators.

Best practices for monitoring memory consumption

Use the following checklist to monitor memory consumption:

☐ Make sure no system in the configuration is running out of physical memory and swapping.

☐ Be aware of 64-bit and 32-bit limitations on both the OS and applications.

☐ Track memory growth over time to identify any memory leaks.

For a checklist that includes all performance tuning issues, see "Performance tuning checklist" on page 53.

Configuring the mid tier, AR System and database server

After establishing a baseline for CPU, memory and IO resource utilization at the system level, the next step is to make sure each individual component—such as the mid tier, AR System and database— is configured for optimal performance as described in the following sections:

- Tuning the mid tier (page 15)
- Tuning AR Server (page 33)
- Tuning a database server (page 38)

Tuning the mid tier

The web client for the AR System platform is the BMC Remedy Mid Tier (the mid tier). The mid tier's main function is to transform any AR System application into a web application accessible by using a web browser. Thus the mid tier is a web application in the sense that it delivers web content to browser requests and is deployable on any J2EE-compliant servlet engine. However, it is not a standard web application because it does not contain any resources to fulfill use cases. All mid tier content is dynamically derived from the AR System applications that are deployed on the AR System server to which it connects.

This section explains how to configure the settings for the mid tier and the web infrastructure hosting the mid tier application to optimize browser response time for your user. In other words, this section provides instructions for optimizing the performance of the mid tier as a web application.

The mid tier is a web application conforming to the J2EE Servlet 2.3 specification. It can be deployed on a wide range of web application servers or servlet engines as specified in the AR System server compatibility matrix (http://www.bmc.com/support_home).

Note: In the remainder of this paper, the application server hosting the mid tier will be referred to as the web server.

In AR System 7.5 and later versions, Apache Tomcat is the default servlet engine included with the AR System installer. One advantage of this is that a licensed third-party product such as ServletExec does not need to be introduced into a production environment. This keeps deployment costs to a minimum. Tomcat performs as well as other application servers or servlet engines under most conditions. It also has a much smaller memory footprint because it is not a full-blown J2EE application server.

Fine tuning the mid tier web application includes the following:

- Fine tuning the web infrastructure hosting the mid tier application
- Fine tuning the mid tier web application.

The web infrastructure needs to be fine tuned separately because the JVM parameters, servlet engine thread configurations, and some HTTP protocol parameters are not controlled at the web application level. Instead, these parameters are set at the web infrastructure level.

Fine tuning the web infrastructure for mid tier

The following settings and parameters significantly impact browser response time:

- HTTP keep-alive (page 16)
- JVM settings (page 18)
- Threads configuration of the application server hosting the mid tier (page 21)

HTTP keep-alive

HTTP keep-alive is officially supported in HTTP 1.1. All current browsers, such as Firefox and Internet Explorer versions 6 or later, support HTTP keep-alive. Keep-alive is necessary because the original simple HTTP 1.0 protocol only requires the browser to open a TCP socket, send the request, receive the response, and then close the socket. This simple protocol works well but can tax browser performance for dynamic web applications. Performance is impaired due to the constant opening and closing of sockets by the browser, especially when SSL (that is, HTTPS protocol) is used or when a single URL has references to many other resources (such as images, CSS, and javascript) embedded in the HTML content corresponding to the URL. By using HTTP keep-alive, the browser maintains the opened sockets, keeping them alive. Subsequent requests are sent by using these already-opened sockets.

Most web servers and application servers have HTTP keep-alive on when they are installed out-of-the-box with the default options. However, the HTTP keep-alive parameters are set at a very short interval to reduce resource usage on the hardware that hosts the web server. This short keep-alive interval is generally sufficient for static web applications but is insufficient for AJAX type or dynamic web applications, as these applications employ more requests/responses interaction between the browser and the server.

After HTTP keep-alive is turned on, the following HTTP keep-alive parameters can be adjusted:

- Keep-alive count—Specifies how many requests can be submitted by the browser through an already-opened socket before the socket is closed by the web server.
- Connection timeout—Specifies how long the socket will be kept open by the web server when it is idle (in real usage, this approximately translates to browser user think time).

These two parameters are transmitted to the browser by using the HTTP response header. If necessary, the browser will know to establish a new socket connection when it sends out the next request.

A typical default installation has the keep-alive count set at 100 and the connection timeout set at 20 seconds (or less). These values are insufficient for AJAX type web applications, such as the mid tier, because AJAX web applications rely on a multitude of smaller HTTP requests to run a use case.

The following table specifies the recommended values for the HTTP keep-alive parameters of the web server that host the mid tier. If there are resource constraints, minimum recommended values are also provided. The syntax of the setting depends on your web server.

HTTP keep-alive parameter	**Recommended value**
Keep-alive count	infinite (minimum 5000)
Connection timeout	90000 ms (minimum 60000 ms)

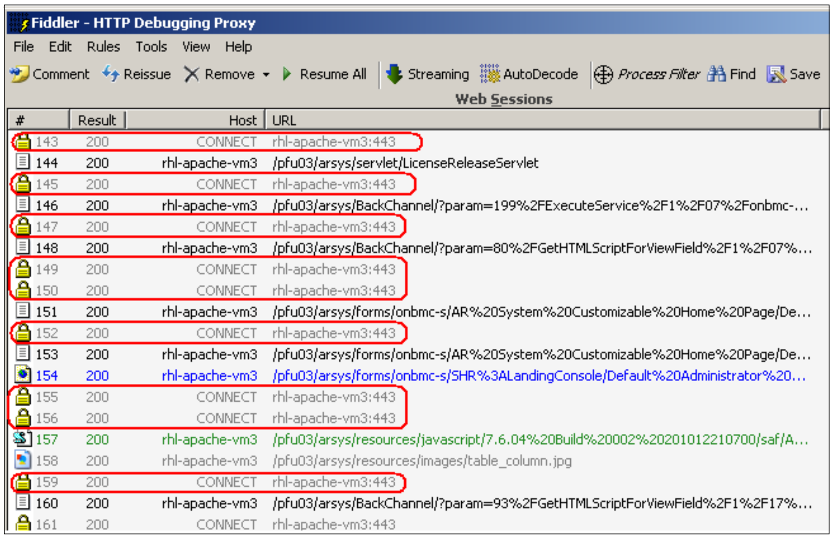
For Tomcat, locate the `Connector` entry in the file ***tomcat dir***`/conf/sever.xml`. The following shows the keep-alive count at infinite and the connection timeout at 90 seconds:

```
<Connector URIEncoding="UTF-8" acceptCount="100"
connectionTimeout="90000"
     maxHttpHeaderSize="8192" maxKeepAliveRequests="-
     1" maxThreads="500" port="80"
     protocol="HTTP/1.1" redirectPort="8443"/>
```

By allowing the browser to re-use sockets through HTTP keep-alive, you can expect a performance gain of approximately 10–30% depending on network latency—the larger the latency, the larger the gain. However, the perceived impact is huge. Correctly configuring the keep-alive settings on HTTPS (secure HTTP by using SSL) outperforms the non-keep-alive on normal HTTP (without SSL) by at least 10%.

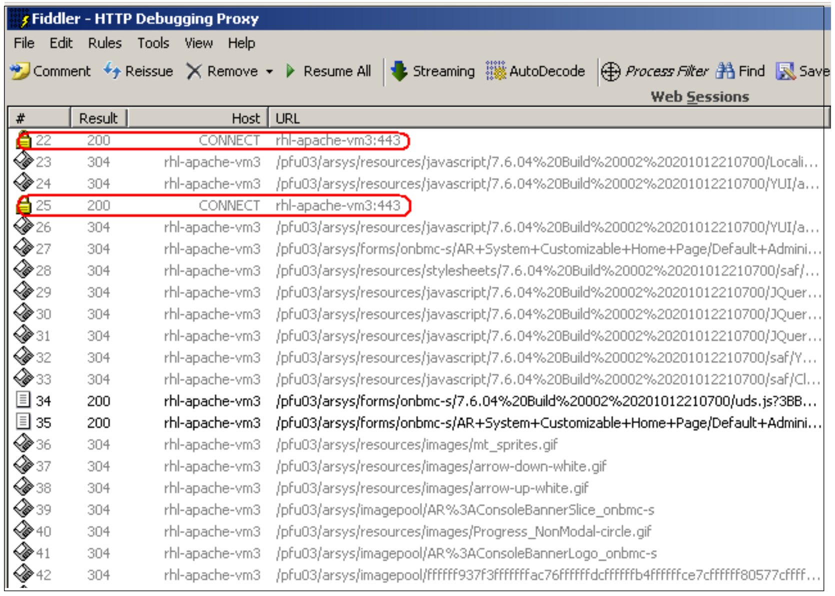
To verify that the keep-alive setting is working in the browser, use a web debugging proxy, such as Fiddler, to capture the exchanges between the browser and the server.

The following two figures compare the exchanges when keep-alive is off and when on. Observe the frequency of SSL socket establishment when keep-alive is OFF.


Fiddler - HTTP Debugging Proxy
File Edit Rules Tools View Help
Comment Reissue Remove Resume All Streaming AutoDecode Process Filter Find Save
Web Sessions

#	Result	Host	URL
143	200	CONNECT	rhl-apache-vm3:443
144	200	rhl-apache-vm3	/pfu03/arsys/servlet/LicenseReleaseServlet
145	200	CONNECT	rhl-apache-vm3:443
146	200	rhl-apache-vm3	/pfu03/arsys/BackChannel/?param=199%2FExecuteService%2F1%2F07%2Fonbmc-...
147	200	CONNECT	rhl-apache-vm3:443
148	200	rhl-apache-vm3	/pfu03/arsys/BackChannel/?param=80%2FGetHTMLScriptForViewField%2F1%2F07%...
149	200	CONNECT	rhl-apache-vm3:443
150	200	CONNECT	rhl-apache-vm3:443
151	200	rhl-apache-vm3	/pfu03/arsys/forms/onbmc-s/AR%20System%20Customizable%20Home%20Page/De...
152	200	CONNECT	rhl-apache-vm3:443
153	200	rhl-apache-vm3	/pfu03/arsys/forms/onbmc-s/AR%20System%20Customizable%20Home%20Page/De...
154	200	rhl-apache-vm3	/pfu03/arsys/forms/onbmc-s/SHR%3ALandingConsole/Default%20Administrator%20...
155	200	CONNECT	rhl-apache-vm3:443
156	200	CONNECT	rhl-apache-vm3:443
157	200	rhl-apache-vm3	/pfu03/arsys/resources/javascript/7.6.04%20Build%20002%20201012210700/saf/A...
158	200	rhl-apache-vm3	/pfu03/arsys/resources/images/table_column.jpg
159	200	CONNECT	rhl-apache-vm3:443
160	200	rhl-apache-vm3	/pfu03/arsys/BackChannel/?param=93%2FGetHTMLScriptForViewField%2F1%2F17%...
161	200	CONNECT	rhl-apache-vm3:443

Figure 4: HTTP keep-alive set to off


Fiddler - HTTP Debugging Proxy
File Edit Rules Tools View Help
Comment Reissue Remove Resume All Streaming AutoDecode Process Filter Find Save
Web Sessions

#	Result	Host	URL
22	200	CONNECT	rhl-apache-vm3:443
23	304	rhl-apache-vm3	/pfu03/arsys/resources/javascript/7.6.04%20Build%20002%20201012210700/Locali...
24	304	rhl-apache-vm3	/pfu03/arsys/resources/javascript/7.6.04%20Build%20002%20201012210700/YUI/a...
25	200	CONNECT	rhl-apache-vm3:443
26	304	rhl-apache-vm3	/pfu03/arsys/resources/javascript/7.6.04%20Build%20002%20201012210700/YUI/a...
27	304	rhl-apache-vm3	/pfu03/arsys/forms/onbmc-s/AR+System+Customizable+Home+Page/Default+Admini...
28	304	rhl-apache-vm3	/pfu03/arsys/resources/stylesheets/7.6.04%20Build%20002%20201012210700/saf/...
29	304	rhl-apache-vm3	/pfu03/arsys/resources/javascript/7.6.04%20Build%20002%20201012210700/JQuer...
30	304	rhl-apache-vm3	/pfu03/arsys/resources/javascript/7.6.04%20Build%20002%20201012210700/JQuer...
31	304	rhl-apache-vm3	/pfu03/arsys/resources/javascript/7.6.04%20Build%20002%20201012210700/JQuer...
32	304	rhl-apache-vm3	/pfu03/arsys/resources/javascript/7.6.04%20Build%20002%20201012210700/saf/Y...
33	304	rhl-apache-vm3	/pfu03/arsys/resources/javascript/7.6.04%20Build%20002%20201012210700/saf/Cl...
34	200	rhl-apache-vm3	/pfu03/arsys/forms/onbmc-s/7.6.04%20Build%20002%20201012210700/uds.js?3BB...
35	200	rhl-apache-vm3	/pfu03/arsys/forms/onbmc-s/AR+System+Customizable+Home+Page/Default+Admini...
36	304	rhl-apache-vm3	/pfu03/arsys/resources/images/mt_sprites.gif
37	304	rhl-apache-vm3	/pfu03/arsys/resources/images/arrow-down-white.gif
38	304	rhl-apache-vm3	/pfu03/arsys/resources/images/arrow-up-white.gif
39	304	rhl-apache-vm3	/pfu03/arsys/imagepool/AR%3AConsoleBannerSlice_onbmc-s
40	304	rhl-apache-vm3	/pfu03/arsys/resources/images/Progress_NonModal-circle.gif
41	304	rhl-apache-vm3	/pfu03/arsys/imagepool/AR%3AConsoleBannerLogo_onbmc-s
42	304	rhl-apache-vm3	/pfu03/arsys/imagepool/ffffff937f3fffffffac76ffffffdcffffffb4ffffffce7cffffff80577cffff...

Figure 5: HTTP keep-alive set to on

JVM settings

The next set of parameters to fine tune for the web infrastructure are the JVM settings:

- JVM heap size
- MaxPermSize

These values affect the frequency and pattern of garbage collection performed by the JVM, which can be CPU intensive.

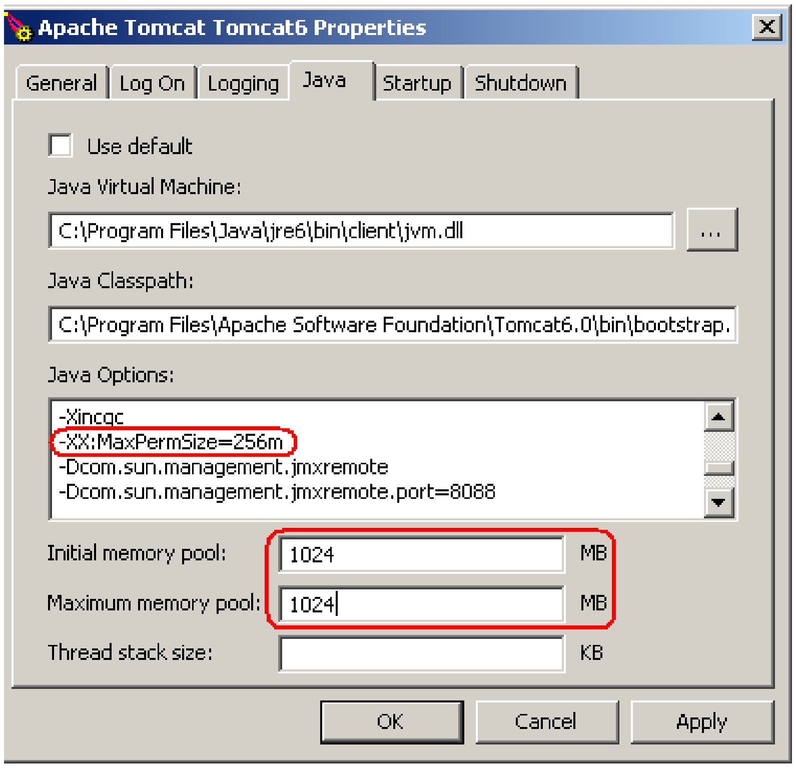
Non-optimal JVM settings might result in the following:

- The Garbage Collection (GC) cycle occurring frequently, thus consuming CPU cycles that would otherwise be available to service application users
- Memory allocation not being used completely, thus depriving other applications on the same machine of available memory

The following table provides recommended JVM parameter settings when the Tomcat servlet engine hosts only the mid tier web application. The following values are set in the Tomcat startup script:

JVM parameter	Recommended value
JVM heap	-Xms1024m –Xmx1024m
MaxPermSize	-XX:MaxPermSize=256m

(Optional) On a Windows platform, you can use the Tomcat configuration tool if Tomcat was installed as a Windows service. The following figure shows the Tomcat properties dialog box. The recommended JVM heap setting is based on profiling the memory footprint of the Tomcat servlet engine. An additional 850 MB is then added for the mid tier based on an out-of-the-box mid tier installation.



Use default

Java Virtual Machine: C:\Program Files\Java\jre6\bin\client\jvm.dll

Java Classpath: C:\Program Files\Apache Software Foundation\Tomcat6.0\bin\bootstrap.

Java Options:

```
-Xincgc
-XX:MaxPermSize=256m
-Dcom.sun.management.jmxremote
-Dcom.sun.management.jmxremote.port=8088
```

Initial memory pool: 1024 MB

Maximum memory pool: 1024 MB

Thread stack size: KB

Figure 6: Tomcat configuration tool on Windows

Consider the following when configuring your JVM settings:

- If your installed Tomcat servlet engine is also hosting another web application in addition to the mid tier, you must increase the maximum and minimum JVM heap settings based on the memory usage of this application. Again, the recommended values in the above table are specifically for a JVM running the Tomcat servlet engine hosting only the mid tier web application.
- If you are using another servlet engine or an application server other than Tomcat, you must determine the memory requirement for that application server when you configure the JVM heap setting. For the out-of-the-box application, you should then add an additional 850MB.
- If you want to allocate more memory to the JVM and your hardware has more physical memory available, increase the `arsystem.ehcache.referenceMaxElementsInMemory parameter in <MT>/WEB-INF/classes/config.properties` file so that the mid tier can use the additional memory to better accommodate the caching of AR System forms for higher performance. This configuration parameter is specific to the cache and the value is based on empirical data that BMC has collected while profiling the mid tier under load. For each additional 700 MB, you can increase the `arsystem.ehcache.referenceMaxElementsInMemory` by 1250 MB.

  You can determine the maximum number of in memory elements of each AR System type in the mid tier cache by multiplying the reference number by the weight factor as specified in `config.properties`. For example, the default maximum number of in memory active links is 6130 = 1250 * 4.904. This implies that you can adjust the weight factor based on your AR System application. For example, for applications with fewer active links, decrease the active links factor.

Many BMC customers are moving toward 64-bit platforms and running the 64-bit JVM. Be aware that the 64-bit JVM has performance overhead (see http://www.oracle.com/technetwork/java/hotspotfaq-138619.html#64bit_performance). BMC internal performance stress tests demonstrate that the 32-bit JVM outperforms the 64-bit JVM by at least 45% in terms of CPU utilization. However, if you need the 64-bit JVM, consider using hybrid mode and parallel GC as recommended by Oracle, that is, `-XX:+UseCompressedOops` and `-XX:+UseParallelGC`. The details and implications of using hybrid mode and parallel GC are beyond the scope of this document.

Threads configuration of the application server hosting the mid tier

For Tomcat, the last set of parameters to fine tune for the web infrastructure is the number of worker threads and the maximum accept count.

Note: If you are using a servlet engine other than Tomcat, see to the documentation for your particular application server to adjust the corresponding parameters.

The following are the two thread configuration parameters:

- `maxThreads`—The maximum number of HTTP requests that can be processed in parallel by Tomcat.
- `acceptCount`—The maximum number of HTTP requests that can be queued if no thread is available to service the request.

When the acceptCount queue reaches its maximum, additional HTTP requests will be dropped by Tomcat.

The following table provides the recommended thread configuration values specific to the Tomcat servlet engine:

Tomcat connector parameter	Recommended value
`maxThreads`	500
`acceptCount`	100

For Tomcat, locate the `Connector` entry in the file *`tomcat dir`*`/conf/sever.xml.` The following shows the max threads and accept count parameters configured as recommended:

```
<Connector URIEncoding="UTF-8" acceptCount="100"
connectionTimeout="90000"
     maxHttpHeaderSize="8192" maxKeepAliveRequests="-
     1" maxThreads="500" port="80"
     protocol="HTTP/1.1" redirectPort="8443"/>
```

In summary, the following values are recommended for fine tuning the web infrastructure hosting the mid tier application:

Keep-alive parameter	**Recommended value**
Keep-alive count	Infinite (minimum 5000)
Connection timeout	90000 ms (minimum 60000 ms)
JVM parameter	**Recommended value**
JVM heap	`-Xms1024m -Xmx1024m`
`MaxPermSize`	`-XX:MaxPermSize=256m`
Tomcat connector parameter	**Recommended value**
`maxThreads`	500
`acceptCount`	100
64-Bit JVM	**Recommended value**
JVM hybrid mode and GC	`-XX:+UseCompressedOops` `-XX:+UseParallelGC`

Fine tuning the mid tier

Fine tuning the mid tier application includes the following tasks:

- Configuring the mid tier so that the resources necessary to service the current request are already in memory
- Configuring the mid tier so that a resource that is not in memory can be located and loaded quickly. Memory is finite and thus cannot store all possible resources.
- Configure the mid tier to instruct the browser to cache and re-use resources that are infrequently changed.

Because the mid tier’s main function is to transform AR System applications into web applications, it essentially works by compiling the AR System definition of each AR System form into an HTML/JS pair. This process is analogous to compiling JavaServer Pages (JSP) into a servlet by the servlet engine. However, this process is CPU intensive for the mid tier because the AR System paradigm includes fine-grained roles and groups which access control by using the AR System form through the fields on the AR System form. This process of compiling AR System definitions into DHTML (HTML/JS) in the mid tier is referred to as prebuilding, precaching or preloading of AR System forms. For example, prebuilding an AR System form requires prebuilding of its subparts such as active links, fields, and so on. This process is independent of browser caching.

Whenever the browser user accesses an AR System form that is not yet prebuilt by the mid tier into DHTML, the mid tier compilation of the given AR System form into DHTML can be as high as 1 minute, depending on the complexity of the AR System form definition. If the AR System form is already prebuilt and cached in memory, the response time of the mid tier to service the http request is generally less than 10 msec. Therefore to achieve optimal response times in servicing the browsers requests, the mid tier relies heavily on prebuilding AR System forms into DHTML and caching the forms in memory.

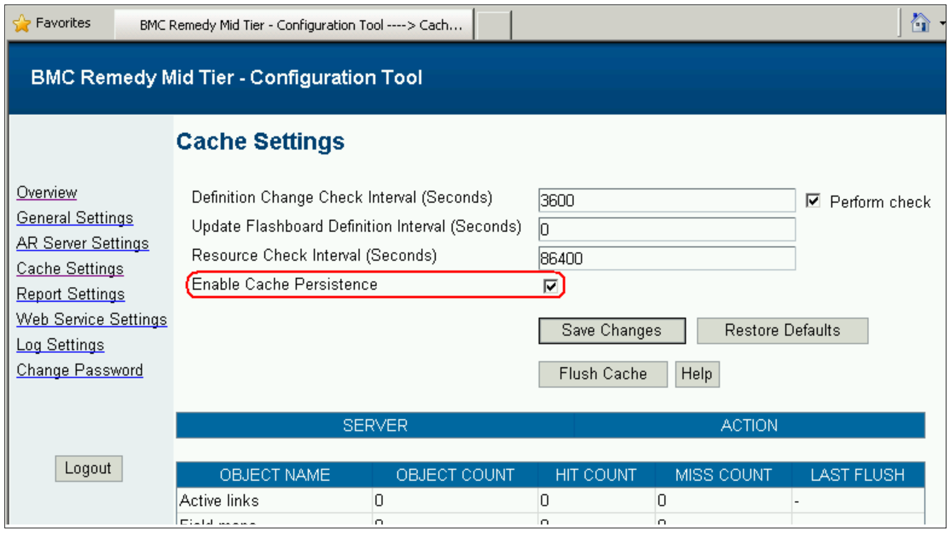
There are three performance related services in the mid tier to assist in this prebuilding process, however only two of these services can be manually configured. The third service performs automatically and requires no administration. The services are:

- Preload (page 24)
- Prefetch (page 25)
- Statistics service (page 26)

Additionally, you can configure the Enable Cache Persistence option in the Mid Tier Configuration Tool to serialize all objects that are prebuilt by the mid tier to disk. When this option is on, any object prebuilt by the mid tier is also serialized to disk. On restart of the mid tier (restarting Tomcat or the application server that hosts the mid tier web application), any AR System object, such as active links, fields, roles, groups, or forms, that is needed by the mid tier is loaded from the disk first if available. When this option is on, an object is not available in memory, but is located and loaded from the disk first, if available.

Note: BMC recommends that the Enable Cache Persistence option is turned on in a deployment environment.

The following figure shows Enable Cache Persistence activated on the Cache Settings page of the Mid Tier Configuration Tool.



Cache Settings

Definition Change Check Interval (Seconds)	3600	☑ Perform check
Update Flashboard Definition Interval (Seconds)	0	
Resource Check Interval (Seconds)	86400	
Enable Cache Persistence	☑	

SERVER	ACTION

OBJECT NAME	OBJECT COUNT	HIT COUNT	MISS COUNT	LAST FLUSH
Active links	0	0	0	-

Figure 7: Enable cache persistence

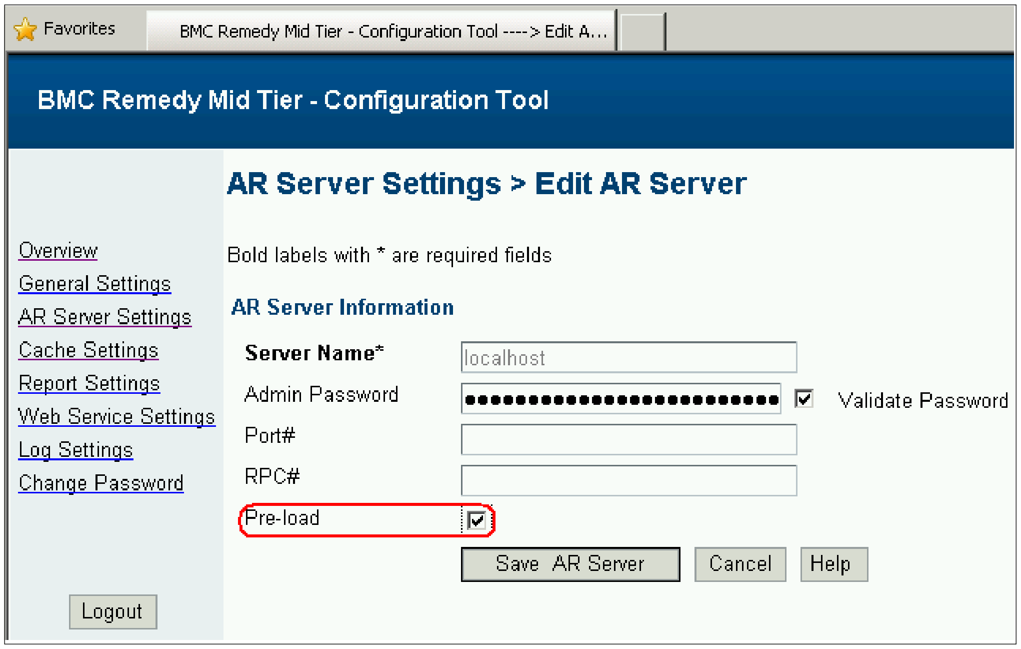
The following descriptions of the three performance related services in the mid tier assume that the Enable Cache Persistence option is on. When the mid tier loads a necessary object into memory with the Enable Cache Persistence on, the mid tier tries first to load the object from disk. If the object is not available from disk, then the mid tier loads the object from the AR System server, and then the object is serialized to disk. On subsequent loading of the same object, the object is now available from the disk unless the definition of the object has changed according to the Definition Change Check Interval parameter.

Preload

When the preload service is on for a particular AR System server as configured in the mid tier, then on mid tier startup the preload service will first preload all of the active links and menus, and then preload all of the AR System forms that contain active links. Any AR System form with active links must be a user-facing form for a use case, and thus will be accessed by the user at some point.

The final process of compiling the AR System form into HTML/JS is performed when an actual user accesses the mid tier by using the browser. At this point a view can be determined. This final process happens quickly after the AR System form object exists in memory.

The following figure shows Preload activated on the Edit AR Server page of the Mid Tier Configuration Tool.



AR Server Settings > Edit AR Server

Bold labels with * are required fields

AR Server Information

Server Name*	localhost
Admin Password	●●●●●●●●●●●●●●●●●●●●●●●●● ✓ Validate Password
Port#	
RPC#	
Pre-load	✓

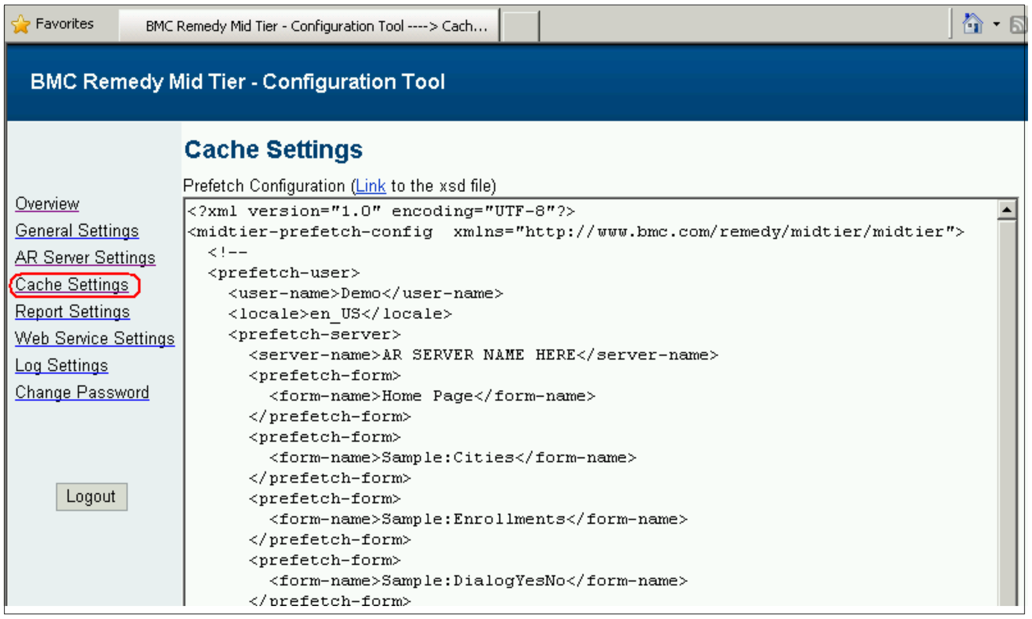
Figure 8: Preload

Prefetch

If the `MT dir/WEB-INF/classes/prefetchConfig.xml` file is not empty, all of the AR System forms as specified in the `prefetchConfig.xml` file are preloaded into memory. All of these AR System forms will also be serialized to disk with the Enabled Cache Persistence option on. The AR System forms that are contained in the `prefetchConfig.xml` file are necessarily a subset of the AR System forms that are preloaded by the preload service.

> **Important**: Use prefetch only when you know the specific set of AR System forms for your use cases. Otherwise, use the preload service.
>
> Use either the preload service or the prefetch service but not both, because preload already loads a superset of the data as specified in the prefetch file.

Optionally, the `prefetchConfig.xml` file can be accessed through the Cache Settings page of the Mid Tier Configuration Tool.



Cache Settings

Prefetch Configuration (Link to the xsd file)

```
<?xml version="1.0" encoding="UTF-8"?>
<midtier-prefetch-config  xmlns="http://www.bmc.com/remedy/midtier/midtier">
  <!--
  <prefetch-user>
    <user-name>Demo</user-name>
    <locale>en_US</locale>
    <prefetch-server>
      <server-name>AR SERVER NAME HERE</server-name>
      <prefetch-form>
        <form-name>Home Page</form-name>
      </prefetch-form>
      <prefetch-form>
        <form-name>Sample:Cities</form-name>
      </prefetch-form>
      <prefetch-form>
        <form-name>Sample:Enrollments</form-name>
      </prefetch-form>
      <prefetch-form>
        <form-name>Sample:DialogYesNo</form-name>
      </prefetch-form>
```

Figure 9: prefetchConfig.xml

Statistics service

Whenever a browser user accesses a given AR System form, the statistics service collects the information necessary to build a view for the given AR System form. (A view of an AR System form from the browser perspective is the corresponding HTML/JS.) The statistics service builds a quintuple list (AR System server name, form name, view name, locale, user group combination), sorted by the frequency of access with the most frequently accessed AR System form listed first.

On restart of the mid tier after completion of the preload or prefetch service, the statistics service loads the AR System forms and the corresponding HTML/JS into memory. The AR System forms and HTML/JS are loaded into memory in the order given by the list using a thread with a lower than normal priority, allowing the mid tier to service browser users in parallel.

Note: The actual sequence executed by the mid tier in relation to preload, prefetch, and statistics services is more complex than as described here, and has been simplified for clarity.

After completion of the statistics service, the AR System forms and corresponding HTML/JS (as given by the collected data of actual usage) is loaded into memory. This allows the mid tier to quickly respond to browser requests.

The collected statistics are serialized to the `MT-dir/WEB-INF/classes/viewStat.dat` file. The collected data is constantly being validated. Data that is no longer applicable is removed (such as for an AR System server that has been disconnected from the mid tier). You can also view the collected data through the hidden JSP `config_cache_adv.jsp` file by using the Mid Tier Configuration Tool.

Running preload

Considering all of the services, the following is the optimal procedure to run preload once:

1. In the Mid Tier Configuration Tool, turn on Enable Cache Persistence in the Mid Tier Configuration Tool.
2. Turn on preload.
3. Allow preload to finish preloading all user facing AR System forms.
4. Turn off preload.

Using this optimal procedure, the statistics service loads only the objects that correspond to the actual usage of the system into the mid tier memory. After the preload service has run once, all of the relevant objects are written to disk (because Cache persistence is enabled). By turning off the preload service, the statistics service has full memory access to load only those objects which are collected in actual usage. You can repeat this procedure if the applications on the AR System server have changed.

Summary

The following summarizes the performance-related services in the mid tier:

- Select Enable Cache Persistence for a deployment environment.
- Use `the prefetchConfig.xml` file only if you know the specific forms for your use cases. Otherwise, use preload.
- Configure the mid tier to instruct the browser to cache and reuse the resources that are not often changed.

Regarding the last bullet item, the following two parameters in the `*MT dir*/WEB-INF/classes/config.properties` file control the browser cache directives issued by the mid tier:

- `arsystem.formhtmljs_expiry_interval`—Configures the browser cache expiry directives (in seconds) issued by the mid tier for the HTML/javascript pair. Each AR System form (together with a view, locale, and user permission group) is compiled into an HTML/JS pair. For example, if the value of this parameter is set to 3600, then the mid tier instructs the browser to use the HTML and JavaScript for the duration of 1 hour from the time of receipt without further checking for updates from the mid tier. To force an update, the browser user can click **Refresh** in the browser.
- `arsystem.resource_expiry_interval`—Configures the browser cache expiry directives issued by the mid tier for other mid tier resources, such as images, icons, CSS, ClientCore.js, and other platform resources.

Note: Any changes to the parameters above require a restart of Tomcat or the web server hosting the mid tier for the new value to take effect.

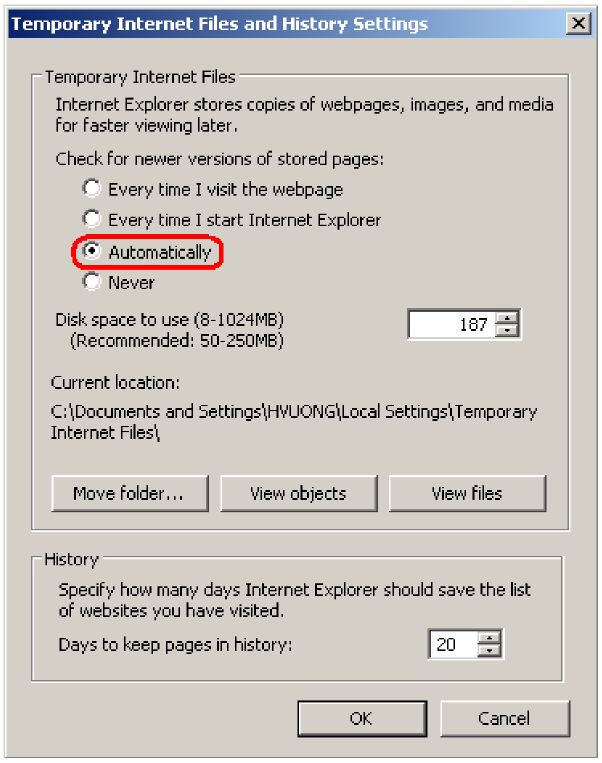
BMC recommends that you configure these parameters to be the same. The value should reflect how often you want the browser to check the mid tier for updates. In a typical deployment environment where the AR System applications are no longer being modified, this value can be set to a week or longer.

After the expiry interval has elapsed for a given resource, on subsequent browser refresh for the same resource, the browser will send a request to the mid tier with an -if-modified-since- HTTP header that contains the original time issued by the mid tier. On receipt of the request, the mid tier checks to see if the resource has changed. If not, the mid tier will return a 304 HTTP response (with no data payload) instructing the browser to reuse and update the expiry interval of the resource. Although the response in this case has zero data payload, on networks with bandwidth saturation or long latency, too many such requests can result in poor browser performance.

In production usage, suppose both of these values are set to 1 week when a change in an AR System form occurs. If the change is for a non-user-facing AR System form, then the change will takes effect as soon as the form definition is saved to the AR System server. If the change is for a user-facing form, then the users who accessed the form within the last week must refresh their browsers to see the change (assuming the mid tier had already updated the change in its cache).

Although the cache directives are issued by the mid tier, the browser may ignore them depending on the browser settings, thus resulting in poor performance. Make sure the browser setting observes the cache directives.

For Internet Explorer 6 or later, BMC recommends setting the Check for newer versions of stored pages option to Automatically in the Temporary Internet Files and History Settings dialog box, as shown in the following figure.



Temporary Internet Files and History Settings

Temporary Internet Files

Internet Explorer stores copies of webpages, images, and media for faster viewing later.

Check for newer versions of stored pages:

- Every time I visit the webpage
- Every time I start Internet Explorer
- [✓ Automatically]
- Never

Disk space to use (8-1024MB) (Recommended: 50-250MB): 187

Current location:
C:\Documents and Settings\HVUONG\Local Settings\Temporary Internet Files\

History

Specify how many days Internet Explorer should save the list of websites you have visited.

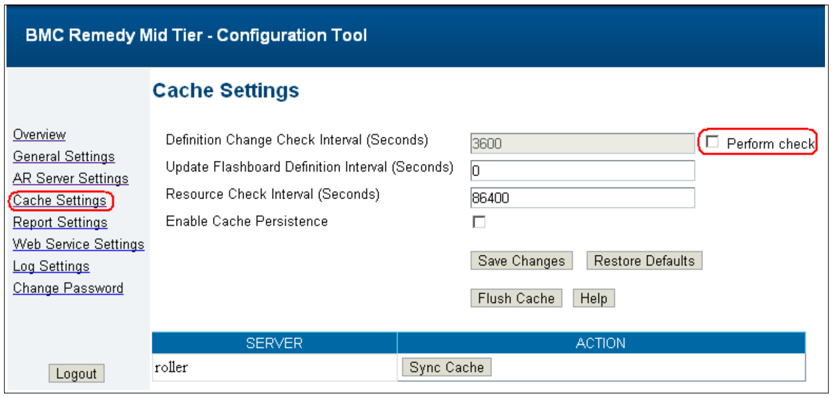
Days to keep pages in history: 20

Figure 10: Recommended browser cache setting

The following table lists the parameter values BMC recommends to fine tune the mid tier application.

Mid tier parameter or service	Recommended value
Enable Cache Persistence	Always on for a production environment
Prefetch or preload service	Use prefetch only when a specific set of AR System forms are known. Otherwise, use preload (recommended).
Recommended preload procedure	1. Turn on Enable Cache Persistence. 2. Turn on preload. 3. Allow preload to finish preloading all user facing AR System forms. 4. Turn off preload (allowing statistical service to take over).
`arsystem.formhtmljs_expiry_interval` `arsystem.resource_expiry_interval`	Set both parameters to the same value to reflect how often you want the browser to check with the mid tier for updates. In a deployment environment where the AR System applications are not modified, set to 604800 (1 week) or higher. The minimum recommended value is 86400 (1 day). For the new values to take effect, restart the mid tier.
Definition Change Check Interval	In a deployment environment where the AR System applications are not modified, turn this off. Otherwise, map this to the frequency of your AR System application modification. For example, if you push changes out every Sunday, set this frequency to 604800 (1 week).
`arsystem.log_level`	Severe. This can also be set through the Mid Tier Configuration Tool.

The following figure shows the Definition Change Check Interval option deactivated on the Cache Settings page in the Mid Tier Configuration Tool.


BMC Remedy Mid Tier - Configuration Tool

Cache Settings

Definition Change Check Interval (Seconds)	3600 ☐ Perform check
Update Flashboard Definition Interval (Seconds)	0
Resource Check Interval (Seconds)	86400
Enable Cache Persistence	☐

SERVER	ACTION
roller	Sync Cache

Figure 11: Definition Change Check Interval option set to off

Mid tier case studies

The following case studies illustrate performance improvements achieved by fine tuning the mid tier.

Case 1

Scenario: When the mid tier was under normal usage load, an AR System form took several minutes to load in the browser for some users. Some users received an HTTP 500 error.

Analysis: An HTTP 500 error is a generic web server error. Unless more specifics can be found, this error is difficult to resolve. Start investigating this in the web server logs, not in the web application logs. Look for any error at or near the time when the problems occur. For this case study, the factor of slow loading of forms plays a large role.

When the service is poor, there is usually either a CPU or resource contention. The natural course of action is to monitor the JVM.

Root cause: On monitoring the JVM hosting the Tomcat running the mid tier, it was observed that the JVM heap was over 90% utilized, so the GC was often running to reclaim memory. In the Tomcat log, out of memory exceptions generated by the JVM were found. These memory exceptions resulted in HTTP 500 errors, a generic web error to obfuscate details from potential hackers. Tomcat was found to be hosting an additional web application—BMC Remedy Knowledge Management (RKM).

Resolution: Memory usage of the RKM application was profiled and found to require approximately 200 MB. The JVM heap allocation was then increased to 1200 MB. The observed response time for loading an AR System form then returned to an acceptable level after restarting the JVM with the new heap allocation.

Case 2

Scenario: When a deployment of mid tier was put under HTTPS, the average response time for browser loading of the same AR System form increased by over 35%.

Analysis: The only factor is the HTTPS protocol. When a problem concerns browser loading time, use a web debugging proxy, such as Fiddler, which provides the details of each request/response (including timing) for an entire use case. Such a tool also captures the same use case under plain HTTP. The capture of HTTP and HTTPS results can be compared to find where the additional time is spent.

Root cause: Using a web debugging proxy to capture the requests/responses of the browser, it was observed that an SSL socket was negotiated for every request sent by the browser.

Resolution: HTTP keep-alive was turned on, with keep-alive count set at infinite and max connection timeout set at 60 seconds. After restarting the web server, the AR System form loaded 10–15% faster than with plain HTTP. A10–15% gain can be expected with keep-alive on (with max connection timeout at 60 seconds or greater) than with keep-alive off.

Case 3

Scenario: When the mid tier is deployed as an internet application (not over VPN) in the United States, users in India experienced delays greater than 2 minutes to load an AR System form.

Analysis: Based on general networking knowledge, this is probably a latency problem. The latency at the TCP layer did not provide accurate information in terms of browser performance because the browser application works at the HTTP layer. Therefore the latency must be determined at the HTTP layer. To analyze browser performance relating to latency, use an http-ping tool, such as the one from Core Technologies (for Windows). Fiddler can also provide HTTP latency, but only when actual requests are made. In contrast, the http-ping tool is more like the ping tool.

For this case study, Fiddler was used to capture the same use case run from the United States and from India. Then a request was made by request comparison to see the accumulated effect of the HTTP latency.

Root cause: Using http-ping, the HTTP latency from India was measured. Then using Fiddler, the HTTP requests and responses of the given use case from the US and from India were captured. After comparing the cumulative effect of the HTTP latency, the browser cache directive was increased.

Resolution: The values of both `arsystem.formhtmljs_expiry_interval` and `arsystem.resource_expiry_interval` were changed to 86400 (1 day). After restarting the mid tier, loading of AR System forms took 5-7 seconds. With expiry for HTML/JS and resources set to 1 day as above, the browser will load from cache without sending requests to the mid tier for the duration of 1 day. After 1 day, requests for the same resources are sent to the mid tier with an If-Modified-Since header, allowing the mid tier to reply with an HTTP 304 (use browser cache and update expiry as resources have not changed) with 0 byte payload. This low payload allows for faster browser loading of the AR System forms.

Though the resolution cannot fix the latency issue, whenever a user loads a form, the user interface displays quickly while the data is slower to display. This is because the HTML/JS are already cached in the browser. The overall user experience improved because the browser was more responsive and less data was transmitted.

Tuning AR System server

Consider the following when tuning the AR System server:

- It is a multithreaded application.
- Its default thread count settings are very low to accommodate the lowest common denominator for hardware.

If the thread count is set too low, the AR System server will have low CPU use, poor throughput, and potentially poor response time under high loads. Defining too many threads can result in unnecessary thread administration. Suggested thread counts are 3 times the number of CPUs for fast and 5 times the number of CPUs for list. So a two CPU box might have 6 threads for fast and 10 threads for list. BMC recommends using the same value for the minimum and maximum settings.

AR System server–based applications work well on a variety of hardware platforms, including symmetric multiprocessor (SMP) systems that have many CPUs. The operating system for SMP systems might develop thread contention with very high thread counts for fast and list threads. Therefore, SMP systems with more than 8 CPUs might perform better with smaller thread counts, where the maximum for any thread pool is 30 or less.

Consider these suggestions as a starting point. Since there are several variables, including different hardware, CPU architecture, CPU speed, and so on, BMC recommends that you benchmark your environment to determine its optimum settings for fast and list threads.

Thread counts can also be used to allocate resources to different workload components. This is recommended if the workload is heavy and provides the following advantages:

- Limits the number of thread to a particular workload.
- Makes sure that the workload does not directly interfere with the regular workload running on fast/list threads by having separate pools.

To limit the computer resources allocated to reporting, you can associate reporting with a private thread that has a modest thread count.

Similarly, allocate a Command Automation Interface (CAI) plug-in with enough private threads to handle Storage Resource Manager (SRM) workload.

> **Note**: For Solaris, BMC lab benchmarking has shown that using `libumem` for memory management provided better AR System server performance than the standard Solaris memory management library. Therefore, in addition to the recommendations provided, BMC also recommends using `libumem` when deploying an AR System server on Solaris. Using `libumem` requires setting the `LD_PRELOAD` environment variable to `libumem.so` before starting the AR System server.

AR Server configuration parameters

The following are recommendations for setting AR System configuration parameters for performance optimization. These recommendations are for version 7.5 or later:

AR Server configuration parameter	Description
`Delay-Recache-Time`	The number of seconds before the latest cache is made available to all threads. Valid values are 0-3600 seconds. The default value is 5 seconds. The recommended value is 300 (5 minutes).
`When Delay-Recache`	Time is set to 0, every API call gets the latest cache (the cache is copied for every administrator call). Setting this option to 0 causes slower performance for cache operations because the AR System server needs to check with the database frequently for definition and group information changes.
`Max-Entries-Per-Query`	The maximum number of requests returned by a search. The default value is no server-defined maximum (entry is not defined). The recommended value is 2000. Because users can also specify the maximum number of requests returned through Search Preferences in the AR System User Preference form or the Options dialog box in BMC Remedy User, the actual maximum is the lower of these values. BMC recommends always setting a value for this parameter as unqualified searches can yield enormous results resulting in unpredictable system behavior.

`Next-ID-Block-Size`	Specifies the number of IDs that can be allocated to create operations at one time. The default value is 10. The recommended value is 100, meaning that AR System server will allocate unique IDs by 100 count blocks. A value less than 10 means that AR System server needs to verify with the DB more frequently for the next block of IDs. A larger value means the AR System server will allocate a larger block of IDs. Large values can cause the IDs to quickly go out of range, especially in a server group environment. Smaller values can cause frequent create operations to encounter contention IDs.
`Server-Side-Table-Chunk-Size`	For server-side table fields, the number of requests (or size of the chunk) that the server retrieves at one time from the database and stores in memory to process during filter or filter guide actions. The server then retrieves, stores, and processes the next chunk until all requests are processed. The recommended, and default, value is 1000. The value of 0 causes the server to retrieve an unlimited number of requests. Specifying a value lower than 1000 causes the server to process smaller chunks, which reduces server memory use but results in slower processing because the server must access the database many times, especially for large tables. Specifying a value higher than 1000 causes the server to retrieve and process large chunks of data and access the database fewer times, resulting in improved performance at the cost of increased memory use.
`Allow-Unqual-Queries`	Specifies whether unqualified searches can be performed on the AR System server. Valid values are T (allows unqualified searches) or F (disallows unqualified searches). The default value is T. The recommended value is F (disallow). Unqualified searches are ARGetListEntry or ARGetListEntryWithFields calls, in which the qualifier parameter is NULL or has an operation value of 0 (AR_COND_OP_NONE). Such searches might cause performance problems because they return all requests for a form. This is especially problematic for large forms.

`Cache-Mode`	Specifies whether a cache mode optimized for application development is on. In development mode, user operations might be delayed when changes are made to forms and workflow. Valid values are 1 (development cache mode is on) or 0 (development cache mode is off and the server is in production cache mode). The default value is 0. The recommended value for a production environment is 0 (development mode off).
`Debug-mode`	Specifies the server logging modes. See the *Configuration Guide* for all possible values. The recommended value is 0 (all logging off for a production environment).
`Submitter-Mode`	Specifies whether the Submitter field can be changed and whether the submitter of a request must have a write license to modify it. Valid values are 1 (locked mode— the Submitter field cannot be changed after a request is submitted) or 2 (changeable mode—the Submitter field can be changed after a request is submitted). The default value is 2. The recommended value is 1 (locked). In the locked mode, the AR System server needs to do less checking and is thus more efficient.
`Minimum-API-Version`	The oldest API version with which the server communicates. The default value is 0 (the server communicates with all API versions). BMC recommends setting a value appropriate for your production environment. Generally, your system should support two earlier versions. If the client's API version is less than the specified value, the server refuses to talk with the client, and the client receives a decode error. Supporting too many versions means there are excessive amounts of RPC parameter conversions that will degrade performance. For values corresponding to AR System release versions, see the *Configuration Guide*.
Private-RPC-Socket (Fast:390620) Private-RPC-Socket (List:390635)	BMC recommends setting this parameter for fast and list threads values only. Fast thread values equal 3 times the number of CPUs. List thread values equal 5 times the number of CPUs. The total number of threads should not exceed 8 times the number of CPUs (or cores).

`CMDB-Cache-Refresh-Interval`	The interval of time when CMDB calls the AR System server to update cache-related data. The default is 300 (5 minutes) if the entry does not exist in the AR System server configuration file. The recommended value is 600 (10 minutes).

AR System server case study

Issue: Certain forms in the Asset Management application slowed down at different times of the day during pre-production performance testing.

Symptoms:

- Different users using client machines with different RAM values randomly experienced poor performance.
- Testing was conducted with an empty database, so performance was a major concern especially when moving to a fully loaded database.

Diagnostic steps:

- A combined log for API calls, SQL and Filter processing was taken in a controlled environment for different forms.

```
10:56:51.7380 */+EXP    ARExport -- as user USER1 from
Remedy User (protocol 12) at IP address 555.205.45.25
<SQL > <TID: 0000004284> <RPC ID: 0000274133> <Queue:
List      > <Client-RPC: 390620   > <USER: USER1
> /* Mon Feb 26 2007 10:56:51.7380 */SELECT
fieldId,helpText FROM field WHERE schemaId=748
-------
10:56:55.1280 */-EXP              OK
<API > <TID: 0000000460> <RPC ID: 0000274136> <Queue:
List      > <Client-RPC: 390620   > <USER: USER1
> /* Mon Feb 26 2007

10:56:55.5340 */+EXP    ARExport -- as user USER1 from
Remedy User (protocol 12) at IP address 555.205.45.25
-------
10:56:58.6910 */-EXP              OK
```

Observations:

- During performance issues, the logs showed that most of the time (approximately 7 seconds) was spent on exporting the form definition (`.arf` file) and the view definition (`.arv` file).
- A form and its definition were exported because it was generated while the performance testing was underway.

Recommendation:

- The workaround was to separate the development environment from the performance test environment. The performance test environment was then set with developer cache mode turned off.

Tuning a database server

This section provides the best practices for tuning the following database servers:

- Tuning an Oracle server (page 38)
- Tuning a SQL Server database (page 48)

The server in your BSM configuration that requires the largest amount of I/O bandwidth is the AR System database server. BMC recommends external storage for this system because local disks might fail to keep up with the demands of a substantial implementation, and any I/O bottleneck will limit overall throughput and increase system response time.

Tuning an Oracle server

This section provides the following information for the Oracle database:

- Configuring your AR System database server for optimal performance
- Diagnosing and resolving issues.

Although the commands and syntax differ, similar methodologies should also be effective for other databases.

Initial database configuration

The basic memory structures associated with Oracle include the following:

- System Global Area (SGA)—Shared by all server and background processes. The SGA holds the following:
    - Database buffer cache
    - Redo log buffer
    - Shared pool
    - Large pool (if configured)
- Program Global Areas (PGA)—Private to each server and background process. There is one PGA for each process. The PGA holds the following:
    - Stack areas
    - Data areas

In Oracle 11*g*, the following parameters enable automatic memory management to control both SGA and PGA.

- `memory_max_target`—Governs total maximum memory for both SGA and PGA.
- `memory_target`— Governs existing sizes for both SGA and PGA.

Oracle 10*g* includes the following parameters for automatic memory management for SGA and a separate parameter for PGA:

- `sga_max_target`—Governs total maximum memory for SGA.
- `sga_target`—Governs existing memory for SGA.
- `pga_aggregate_target`—Defines the memory size for PGA.

For both Oracle 10*g* and 11*g* the regular memory parameters, such as `db_cache_size` and `shared_pool_size`, define the minimum size Oracle will maintain for each subarea in SGA.

The following table lists the recommended values for small, medium, and large Oracle databases.

System components	Database size		
	Small	Medium	Large
Recommendations			
CPUs (typical configuration)	8	16	32
Memory (typical configuration)	8 GB	16 GB	32 GB
Concurrent users	800	2000	5000
Sessions	300	500	1000
Processes	300	500	1000
Temporary tablespace	1 GB	1–2 GB	2–4 GB
Log files	3 (500 MB each)	3 (500 MB each)	3 (500 MB each)
Requirements			
`memory_target` (11*g*)	2-3 GB	5-6 GB	12-15 GB
`sga_target` (10*g*)	2 GB	4 GB	10 GB
`db_cache_size`	1.2 GB	3 GB	8 GB
`shared_pool_size`	800 MB	1 GB	2 GB
`pga_aggregate_target`	1 GB	2 GB	4-5 GB

Cursor sharing

The most important parameter setting in Oracle for BMC Remedy based applications is `cursor_sharing`, which determines what kind of SQL statements can share the same cursor. Its default value is EXACT, which means that Oracle allows statements with identical text to share the same cursor.

Because AR System issues SQL statements with literals, there could be thousands of SQL statements issued by the application. Each statement is treated as a different statement and is parsed. This frequent parsing of the SQL statement uses significant resources such as shared pool and library cache latch, which will severely limit performance and scalability. Setting `cursor_sharing` to FORCE or SIMILAR is a way to mitigate this issue. With `cursor_sharing` set to FORCE or SIMILAR, Oracle replaces literal values with system generated bind variables in the SQL statement. This increases soft parsing but significantly reduces hard parsing.

BMC recommends setting `CURSOR_SHARING` to FORCE. The performance of FORCE and SIMILAR is similar, although field experience and internal testing of the AR System platform indicates that FORCE is slightly better than SIMILAR. An excessive number of child cursors is an issue with SIMILAR.

BMC recommends setting `_b_tree_bitmap_plans` to false. BMC Remedy based applications do not provide any bitmap indexes out of the box. However, the optimizer can choose a bitmap access path without the presence of bitmap indexes in the database by using the CPU-intensive BITMAP CONVERSION FROM/TO ROWID operation. Setting `_b_tree_bitmap_plans` to false avoids this issue.

The following Oracle *10g*/*11g* database settings are recommended:

Parameter	**Recommended value**
`Cursor_sharing`	FORCE
`_b_tree_bitmap_plans`	False

Cost Based Optimizer

When used with Oracle 10*g* or later, AR System applications depend on Oracle's Cost-Based Optimizer (CBO) for performance. When a SQL statement is issued by the AR System server, the CBO uses database statistics to determine an optimal execution plan to fetch the required data. By default, Oracle automatically gathers statistics using a scheduled job GATHER_STATS_JOB, which runs locally within a maintenance window between 10 P.M. and 6 A.M. weeknights and all day on weekends.

Database administrators (DBAs) can turn off automatic statistics gathering to avoid conflicts with other business processes running on the database. In such cases, statistics should be collected manually on a regular basis. Statistics should also be gathered after large data loads on updates to reflect correct volume and distribution.

Query the database to determine when statistics were last gathered. If statistics were not gathered recently, make sure that automatic statistics gathering is enabled. Lack of statistics can lead to poor SQL execution plans and inefficient database processing.

The only time you might need to gather statistics manually in an Oracle database is during batch jobs that begin with an empty destination table. Statistics on such a table reflect that it is empty. If the table subsequently becomes very large, operations against the table might begin to perform poorly because the statistics no longer reflect its true size. If this occurs, you can gather statistics during the execution of the batch job.

Consider this only for batch operations that take hours rather than minutes.

Oracle CLOB storage

The default for Character Large Object (CLOB) column storage during an AR System server installation is Out Row. For example, the actual data is stored outside the actual row that contains the CLOB column. If high database space growth is a concern, BMC recommends converting CLOB storage to In Row. For more information, see the *Database Reference*.

Oracle case-insensitivity

If you need to make the Oracle database case-insensitive, set the following parameters in Oracle 10*g*:

Parameter	Description
`NLS_COMP`	Specifies how the predicates in a SQL statement will be compared. Valid values are: • BINARY—Comparisons according to the binary value of the characters. • ANSI—Still available (from Oracle 9*i*) but only for backwards compatibility. • LINGUISTIC—Honors the setting of `NLS_SORT`.
`NLS_SORT`	Specifies the collating sequence for ORDER_BY queries. Valid values are: • BINARY—The collating sequence is based on the numeric value of the characters. • Named Linguistic Sort—Sorting is based on the order of the defined linguistic sort.

To make Oracle use an existing index on a column that is present in one or more queries, do the following:

1. Set `NLS_COMP` and `NLS_SORT` at the session level or the database level.
2. Drop and recreate all ARADMIN indexes as function-based indexes.

```
Setting NLS_SORT in a session
SQL> alter session set NLS_SORT=BINARY_ CI;

Drop and recreate an index as a function-based
index
SQL> drop index <index name>:
SQL> create index <index name> on

<table_name>(NLSSORT(<column_name>,
‘NLS_SORT=BINARY CI’)):
```

If you still have a Full Table Scan against the table after following this procedure, then force the Oracle CBO to pick up the index by doing the following:

- Set the Oracle parameter `optimizer_index_cost_adjust` to 1 at the session level.

```
SQL> alter session set
optimizer_index_cost_adjust=1:
```

The default value for `optimizer_index_cost_adjust` is 100.

Note: This setting forces the Oracle CBO to choose index lookups over Full Table Scans. BMC recommends thoroughly assessing the impact of these settings in your development and quality assurance environments before implementing this setting in Production.

Oracle database diagnostics

Typically, Oracle 10*g*/11*g* diagnostics come in the form of a report called Automatic Workload Repository (AWR). To create an AWR report, a snapshot is taken before and after the period of interest. The report is then generated to show how the system counters (V$ views) changed between the two snapshots. Oracle AWR snapshots are automatically taken every hour unless changed by the DBA. Reports are most valuable when they focus on a period of high activity.

The AWR reports include a high-level summary of system usage, specific observed wait events, and a list of high-load SQL statements. Oracle documentation explains how to interpret the reports. The following points provide additional guidelines.

- Make sure the Buffer Cache and Shared Pool are well used.

Instance efficiency percentages (target 100%)			
Buffer NoWait %:	100.00	Redo NoWait %:	100.00
Buffer Hit %:	99.99	In-memory Sort %:	100.00
Library Hit %:	98.99	Soft Parse %:	99.51
Execute to Parse %:	3.51	Latch Hit %:	100.00
Parse CPU to Parse Elapsd %:	86.68	% Non-Parse CPU:	93.57

A section near the top of the AWR report addresses Shared Pool usage. If the amount of consumed Shared Pool memory is above 80% and the use of statements is low (for example, less than 40% of memory is used by statements executed more than once), your system is not reusing SQL statements efficiently. This could happen if `cursor_sharing` is set to `EXACT`.

Shared pool statistics	Begin	End
Memory Usage %:	86.17	86.27
% SQL with executions>1:	93.33	94.74
% Memory for SQL w/exec>1:	97.83	98.25

The best practice is to set `cursor_sharing` to `FORCE` for Oracle 10*g* or later.

Important: For AR System server settings, the `cursor_sharing` setting in the `ar.cfg` file must match the setting in the `init.ora` file. See the *Oracle's Cursor Sharing for BMC Remedy Products* white paper.

The overall sizing for the Buffer Cache and the Shared Pool in Oracle depends on whether the OS is 32-bit or 64-bit. If you encounter a problem, add memory to the cache or the pool, but do not make them larger than the size shown in the table in "Initial database configuration" on page 38.

- If the Buffer Cache and Shared Pool are well used, check for blocking wait events.

  In the wait event summary at the top of the AWR report, the CPU time event should be near or at the top of the list, ideally 70% or more. Typically, CPU time might be low if you are I/O bound. If the top wait events are I/O-related, you might be getting poor SQL execution plans.

  The high-load SQL statements ordered by Buffer Gets are listed further down the report. If you see statements with very high Buffer Gets per Execute, an index might be missing on a table accessed by the statement. `db_file_scattered_read` wait events are associated with Full Table Scans or Index Fast Full Scans and could indicate a need for additional indexes (or up-to-date statistics for Oracle).

The following table lists the top timed events:

Event	Waits	Time (sec)	Avg wait (msec)	% total call time	Wait class
CPU time	n/a	94	n/a	95.7	n/a
LNS wait on SENDREQ	713	51	71	51.8	Network
Log file parallel write	15,869	10	1	10.6	System I/O
Log file sync	14,965	10	1	10.4	Commit
Control file sequential read	21,781	10	0	10.0	System I/O

- Use the Oracle SQL trace utility.

  For more information about the executed SQL statements and their execution plans, use the Oracle SQL trace utility.

  When SQL tracing is on, raw trace files are generated in an Oracle dump directory. The `tkprof` utility can use these trace files to create reports on executed SQL statements. Multi-threaded applications might produce multiple trace files at the same time, and the trace files might get large quickly, so managing the generated trace files can be challenging.

  Other ways to evaluate SQL execution plans include directly using the SQL plan tables, such `as V$SQLAREA, V$SQL_TEXT`, and `V$SQL_PLAN`. If the `EXECUTION PLAN` has been aged out of `V$SQL_PLAN`, use the `EXPLAIN PLAN` and `AUTOTRACE` commands.

  When tuning SQL statements, getting the runtime execution plan for the SQL statements in question is important. Oracle supplies a script, available in `$ORACLE_HOME/rdbms/admin`, called `awrsqrpt.sql`. This SQL script takes the Start and End AWR Snapshot Ids and the SQLId of the statement to be examined as arguments. If a runtime plan is not available, then use the Explain Plan to get an execution plan for the statement.

  When two plans differ in the Cost column, the plan with the lower Cost values may not necessarily be better. The actual execution for the SQL with the higher Cost might be better than the one with the lower Cost. Run the SQL in a tool, such as SQL*Plus, with timing turned on.

Recommended init.ora settings

If you use an Oracle database with your AR System–based products, BMC recommends using the following `init.ora` settings:

Init.ora parameter	Recommended value
`_b_tree_bitmap_plans`	FALSE
`cursor_sharing`	FORCE
`cursor_space_for_time`	FALSE
`db_block_checking`	FALSE
`db_block_checksum`	TRUE
`db_block_size`	8k
`log_buffer`	10485760
`open_cursors`	500
`optimizer_dynamic_sampling`	2
`session_cached_cursors`	100
`statistic_level`	TYPICAL
`timed_statistics`	TRUE
`undo_retention`	14400
`work_area_policy`	AUTO

Best practices for tuning Oracle database servers

When tuning an AR System Oracle database server, use this checklist:

- ☐ Make sure that `alert.log` is clean.
- ☐ Use an appropriately sized SGA.
- ☐ Set `cursor_sharing` to `FORCE`.
  - ☐ Use AWR reports to collect diagnostics data. Review top timed events for potential bottlenecks.
  - ☐ Monitor the database for top SQL statement. An AWR report can also be used for this purpose. Make sure that the SQL execution plans are optimal.

For expensive SQL statements, also do the following:

- ☐ Make sure that the database statistics are current and representative of the data volume and data distribution.
- ☐ Review the execution plan and look for common red flags such as Merge Join Cartesian operation, full table scans, index full scan, bitmap from/to conversion, and so on.
- ☐ Implement additional indexes if appropriate. BMC provides indexes out of the box. However, there might be cases where additional indexes are required based on the way the application is configured and used. Typically these requirements are documented in product manuals.

For a checklist that includes all performance tuning issues, see "Performance tuning checklist" on page 53.

Oracle case study

Issue: ITSM: Incident Ticket submission took 18 seconds for a single user test.

Symptoms:

- Database trace accounted for less than 2 seconds of time.
- The `arserverd` process accounted for 2 to 3 seconds of CPU time.
- The SQL log from AR System shows SQL time approximately 1 second.

Diagnostic steps:

1. Reduce the number of connections to the database
2. Measure the CPU time of all the relevant processes on the AR System server and Oracle database by doing the following:
   a. Bring up the environment (with limited connections from the AR System server to Oracle).
   b. Execute the transaction once to warm up the environment.
   c. Start the transaction again up to the point before submission.
   d. Record the process status (ps) time for `arserverd` and Oracle processes.
   e. Click SUBMIT.
   f. Verify the `top` command to see which Oracle process is consuming time.
   g. Record the ps time for `arserverd` and Oracle processes.

Observations and analysis:

- The `arserverd` took little time. Most of the CPU (close to 18 seconds) was consumed by the Oracle process.
- A combined log for API calls, SQL and Filter processing was informative. There was a gap of 0.8 seconds on every LOB write. If all LOB writes at different places in the log file were added, the total time accounted for was over 15 seconds.

```
* Thu Dec 07 2010 08:13:54.2535 */OK
* Tue Dec 07 2010 08:13:54.2535 */SELECT C456 FROM
T1253 WHERE C1 = '000000000004902' FOR UPDATE
* Tue Dec 07 2010 08:13:54.2536 */Set LOB into the
above row...
* Tue Dec 07 2010 08:13:54.2550 */OK
* Tue Dec 07 2010 08:13:55.0727 */UPDATE T1253 SET
C459 = EMPTY_CLOB() WHERE C1 = '000000000004902'
* Tue Dec 07 2010 08:13:55.0759 */OK
```

- The Oracle raw trace file also showed a similar pattern during the direct write operation**.**

```
WAIT #4: nam='SQL*Net message from client' ela= 155
driver id=1650815232 #bytes=1 p3=0 obj#=-1
tim=295710595994
WAIT #9: nam='direct path write' ela= 0 file number=7
first dba=396004 block cnt=1 obj#=-1 tim=295710596586
WAIT #9: nam='direct path write' ela= 1 file number=7
first dba=396004 block cnt=1 obj#=-1 tim=295710596738
WAIT #9: nam='direct path write' ela= 52 file number=7
first dba=396004 block cnt=1 obj#=-1
tim=295710596787WAIT #0: nam='SQL*Net message to
client' ela= 5 driver id=1650815232 #bytes=1 p3=0
obj#=-1 tim=295711477135
WAIT #0: nam='SQL*Net message from client' ela= 146
driver id=1650815232 #bytes=1 p3=0 obj#=-1
tim=295711477427
STAT #4 id=1 cnt=1 pid=0 pos=1 obj=0 op='FOR UPDATE
(cr=6 pr=0 pw=0 time=85 us)'
```

- Oracle Support had a note with a similar issue.

    Note 393780.1 Poor Performance Writing Blobs SymptomsA job that is writing BLOBs are taking long time and consumes a lot of CPUs.

    The 'pstack' output of the spinning process shows :

    __fdsync ksfdsyncdata kcflsync kcblsy kcblcn kcblrr kcbldrcls kdlpdba ktsplbfmb ktsplbrecl ktspgsp_cbk1 kdlgsp kdlsfb kdl_write1 kokliccb koklcre kokleva evaopn2 insolev insbrp insrow insdrv inscovexe insExecStmtExecIniEngine insexe opiexe kpoal8 opiodr ttcpip opitsk opiino opiodr opidrv sou2o opimai_real main _start

    CauseEspecially the __fdsync() system calls indicate that the problem is OS-related.

- Technical note from Veritas

    Oracle Import takes longer when using buffered VxFS than using unbuffered VxFS.

    Loading data into database using the "import" utility may be slower with buffered VxFS. Double buffering could be easily avoided by enabling Quick I/O for VERITAS File System (VxFS).

    Quick I/O allows regular files built on VxFS to be accessed as a raw device, bypassing normal file system buffering and allowing direct I/O. There is no question of double-buffering when Quick I/O is used. …

Recommendation: Move to Quick I/O or Raw Devices

Tuning a SQL Server database

Based on field experience and performance tests conducted by the BMC ESM performance team, BMC recommends the Microsoft SQL Server tuning practices provided in this section.

Set PARAMETERIZATION database option to FORCED

In SQL Server 2005/2008, forced parameterization promotes the reuse of SQL execution plans, thereby reducing the time spent parsing SQL queries. The effect is similar to that achieved by Oracle cursor sharing.

To enable forced parameterization by setting the PARAMETERIZATION option to FORCED in the ALTER DATABASE statement, do the following:

1. Connect to the SQL server instance as a user with ALTER permission on the database.
2. Assuming database name is `arsystem`, run the following SQL commands:

   ```
   Alter database ARSystem set PARAMETERIZATION
   FORCED
   go
   ```

3. To verify the parameterization setting of the database to verify, run the following SQL commands:

   ```
   select name, is_parameterization_forced
   from sys.databases
   where name='ARSystem'
   ```

For more information, see SQL Server 2005 Books Online at http://technet.microsoft.com/en-us/library/ms175037.aspx.

Set the READ_COMMITTED_SNAPSHOT database option to ON

BMC recommends turning on the ALLOW_SNAPSHOT_ISOLATION and READ_COMMITTED_SNAPSHOT options for the AR System database. Turning on these options enables the row versioning-based isolation level, which provides the following benefits:

- Read operations retrieve a consistent snapshot of the database.
- SELECT statements do not lock data during a read operation (readers do not block writers, and vice versa).
- SELECT statements can access the last committed value of the row, while other transactions update the row without being blocked.
- Fewer deadlocks and lock escalations occur.
- Fewer locks required by a transaction occur, which reduces the system overhead required to manage locks.

Turning on the READ_COMMITTED_SNAPSHOT option

To turn on the READ_COMMITTED_SNAPSHOT option, do the following:

1. Connect to the SQL server instance as a user with ALTER permission on the database.
2. Make sure there are no active connections to the database except for the connection executing the ALTER DATABASE command.
3. Assuming database name is `arsystem`, run the following SQL commands:

```
ALTER DATABASE arsystem SET
ALLOW_SNAPSHOT_ISOLATION ON
ALTER DATABASE arsystem SET
READ_COMMITTED_SNAPSHOT ON
go
```

4. To get the isolation level of the target database to verify, run the following SQL commands:

```
select name, is_read_committed_snapshot_on
from sys.databases
where name='ARSystem'
go
```

Note: The DBA must make sure that `tempdb` has ample space to support the version store after enabling row versioning-based isolation levels.

SQL Server maximum degree of parallelism

Important: This section applies only to transaction-oriented systems. Skip this section if your system is for reporting or is a Decision-Support System.

When configuring your SQL Server database server, use the max degree of parallelism (MDOP) option to limit the number of processors used in parallel plan execution.

By default, this option is set to 0, which uses all available processors. One long running, CPU-intensive SQL statement could monopolize all processors and create long wait times for other users.

If your transaction environment shows long running, CPU-intensive SQL statements, set MDOP as follows:

Number of cores or CPUs on SQL server	MDOP setting
1–7	0 (default)
8–15	6
16 or more	8

The following example sets the MDOP to 8:

```
sp_configure 'show advanced options', 1;
GO
RECONFIGURE WITH OVERRIDE;
GO
sp_configure 'max degree of parallelism', 8;
GO
RECONFIGURE WITH OVERRIDE;
GO
```

Best practices for tuning SQL Server database servers

Use the following checklist when tuning an AR System SQL Server database server:

☐ Set PARAMETERIZATION option to FORCED for AR System database.

☐ Set the READ_COMMITTED_SNAPSHOT database option to ON for AR System database.

☐ Set the SQL Server MDOP option to the appropriate value.

For a checklist that includes all performance tuning issues, see “Performance tuning checklist” on page 53.

Troubleshooting SQL Server performance

For general information about troubleshooting performance problems, see http://www.microsoft.com/technet/prodtechnol/sql/2005/tsprfprb.mspx for SQL Server 2005, and http://msdn.microsoft.com/en-us/library/dd672789(SQL.100).aspx for SQL Server 2008.

Overall system tuning

This section provides guidelines for optimizing the network, client configuration and overall system.

Optimizing the network

To optimize the network, BMC recommends the following:

- Use a gigabit network with minimal latency between AR System server and database server.
- Do not to install a firewall between the AR System server and the database server because the firewall has a significant negative impact on the performance.
- If there is a load balancer or firewall between the mid tier and the AR System server, configure it to not sever idle connections from the mid tier. Reestablishing severed connections takes additional time.

Optimum client configuration

The optimum client/browser configuration is as follows:

- Dual Core 2+ GHz CPU with 2 MB L2 cache or single core 3+ GHz CPU with 2 MB L2 cache.
- 2-4 GB RAM.
- Firefox 3.6 or later or Internet Explorer 8 or later.
  - Disable all browser add-on toolbars and plug-ins except Adobe Flash.
  - Configure the browser to open pop-ups in a new tab.
  - Configure the browser cache setting to Automatic.

Note: While Virus scan is running all applications will run slower, especially the browser.

Overall tuning strategy

Follow these steps to improve system performance:

1. Create a precise statement of your performance problem. See “Problem statement” on page 8.
2. Evaluate the performance of the following components and determine if troubleshooting is necessary:
   a. CPU and memory resources. See “CPU consumption” on page 11 and “Memory consumption” on page 13.
   b. Mid tier. See “Tuning the mid tier” on page 15.
   c. AR System server. See “Tuning AR Server” on page 33.
   d. Database server. See “Tuning a database server” on page 38.

   Important: Evaluate all components before making any changes.
3. Prioritize any proposed changes based on your problem statement.
4. Perform and evaluate the impact of the highest priority change.

   Important: Make any production environment changes one at a time. Each time a substantial change is made, quantitatively evaluate the effect of that change before making another change.
5. Repeat step 4 for each change in order of priority.

Best practices for tuning the overall system

Use the following checklist when tuning your overall system:

- ☐ Survey each area of your system for possible tuning issues before making any changes.
- ☐ Prioritize all changes by their likely impact on the identified problem.
- ☐ Implement changes one at a time in a prioritized manner.
- ☐ Quantitatively evaluate the impact of every change before considering whether another change should be made.

For a checklist that includes all performance tuning issues, see “Performance tuning checklist” on page 53.

Performance tuning checklist

The following checklist includes all of the checklists in this document.

Performance tuning checklist		
Problem statement	Characterize throughput or response time.	☐ Before problem was observed ☐ After problem was observed
	List any system changes that occurred just before the problem was observed.	☐ Patch changes ☐ Configuration changes ☐ Workload changes ☐ User count changes
CPU consumption monitoring	☐ Collect CPU usage data for each computer in the configuration. ☐ Make sure less than 70% of the total CPU capacity is used on each system in the configuration. ☐ Record which processes are consuming most of the CPU resources. ☐ Record whether I/O wait time is a substantial component of CPU usage. This indicates that the I/O subsystem is not keeping up with demand.	
Memory consumption monitoring	☐ Make sure no system in the configuration is running out of physical memory and swapping. ☐ Be aware of 64-bit and 32-bit limitations on the OS and applications. ☐ Track memory growth over time to identify any memory leaks.	

Mid tier tuning	☐ Use precaching to avoid high latency for first-time access to a form. ☐ Set `resource_check_interval` to one day. ☐ Set minimum heap size to 1 GB and maximum heap size to 1 GB.
Oracle server tuning	☐ Use an appropriately sized SGA. ☐ Set `cursor_sharing` to SIMILAR in Oracle 10*g*. ☐ Use Oracle AWR reports to collect diagnostics. ☐ If system is I/O bound or db_file_scattered_read events exist, consider whether high-load SQL statements need tuning or additional indexes should be added.
SQL server tuning	☐ Set the SQL Server ALTER DATABASE statement's PARAMETERIZATION option to FORCED. ☐ Set the SQL Server MDOP option to the appropriate value. ☐ Set the AR System server Select-Query-Hint option to NOLOCK.
AR System server tuning	☐ Set thread counts for FAST and LIST to appropriate values. ☐ Set `NEXT_ID_BLOCK_SIZE` to 10. ☐ Use AR System server logging to obtain additional diagnostic information. ☐ Set Max Entries Returned by GetList, and disable unqualified searches. Set Server Table Field Chunk Size to reduce the impact of queries with large return sets.
Overall system tuning	☐ Survey each area of your system for possible tuning issues before making any changes. ☐ Prioritize all changes by their likely impact on the identified problem. ☐ Implement changes one at a time in a prioritized manner. ☐ Quantitatively evaluate the impact of every change before considering whether another change should be made.

199037