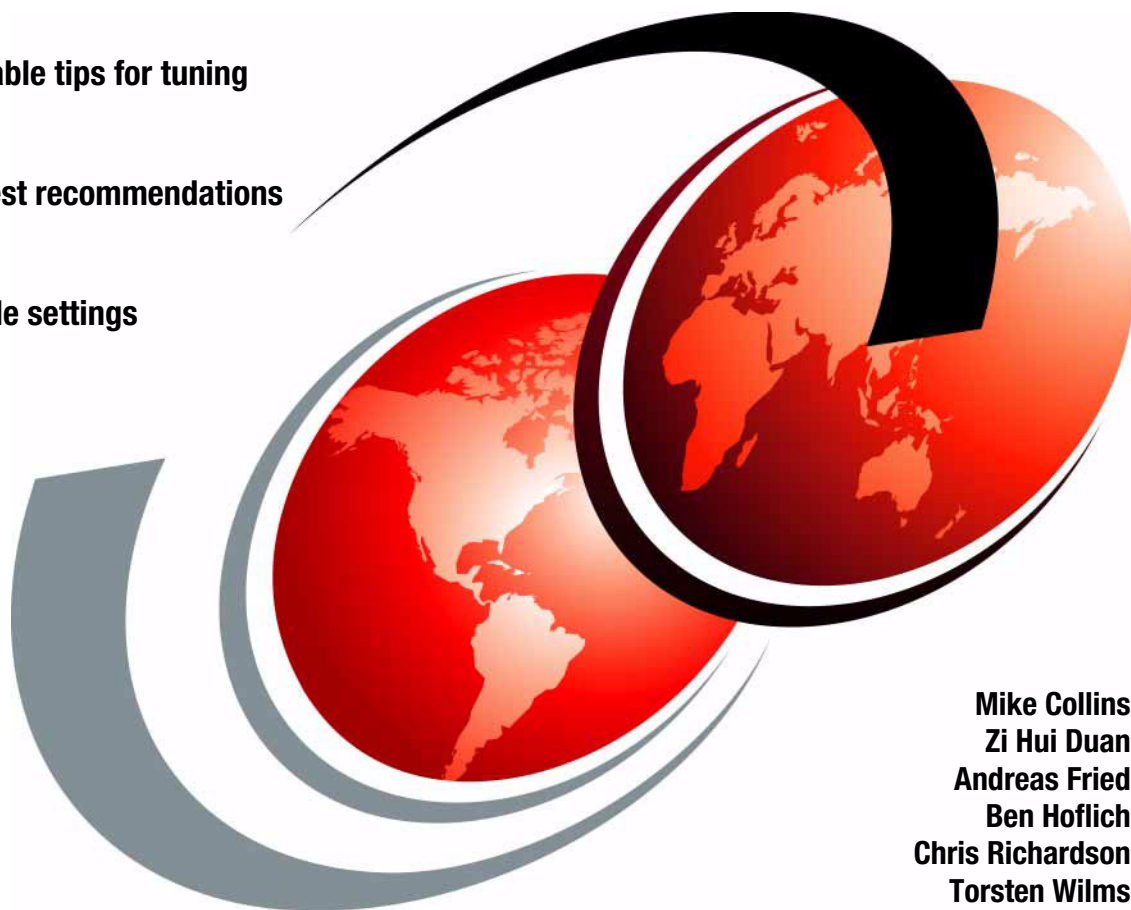# IBM Business Process Manager V8.5 Performance Tuning and Best Practices

Learn valuable tips for tuning

Get the latest recommendations

See example settings

Mike Collins
Zi Hui Duan
Andreas Fried
Ben Hoflich
Chris Richardson
Torsten Wilms

Redbooks

International Technical Support Organization

IBM Business Process Manager V8.5 Performance Tuning and Best Practices

February 2015

Note: Before using this information and the product it supports, read the information in "Notices" on page ix.

First Edition (February 2015)

This edition applies to Version 8.5.5 of IBM Business Process Manager and Version 8.5.5 of IBM Business Monitor.

Contents

Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

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

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

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

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

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

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

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

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

ix

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®	IMS™	Redbooks (logo) ®
alphaWorks®	InfoSphere®	System z®
CICS®	MQSeries®	Teamworks®
Cognos®	OpenPower™	Tivoli®
DB2®	Rational®	WebSphere®
developerWorks®	Redbooks®	
IBM®	Redpaper™	

The following terms are trademarks of other companies:

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

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

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

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Find and read thousands of IBM Redbooks publications

- ► Search, bookmark, save and organize favorites
- ► Get up-to-the-minute Redbooks news and announcements
- ► Link to the latest Redbooks blogs and videos

Get the latest version of the Redbooks Mobile App

iOS

Download Now

Android
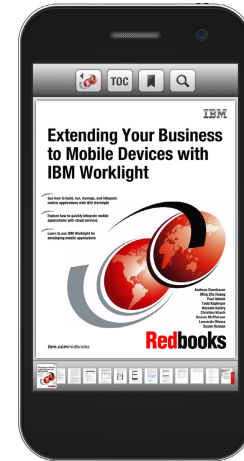
Promote your business in an IBM Redbooks publication

Place a Sponsorship Promotion in an IBM® Redbooks® publication, featuring your business or solution with a link to your web site.

Qualified IBM Business Partners may place a full page promotion in the most popular Redbooks publications. Imagine the power of being seen by users who download millions of Redbooks publications each year!

It's good to be noticed.

ibm.com/Redbooks
About Redbooks → Business Partner Programs

THIS PAGE INTENTIONALLY LEFT BLANK

Preface

This IBM® Redbooks® publication provides performance tuning tips and best practices for IBM Business Process Manager (IBM BPM) V8.5.5 (all editions) and IBM Business Monitor V8.5.5. These products represent an integrated development and runtime environment based on a key set of service-oriented architecture (SOA) and business process management (BPM) technologies. Such technologies include Service Component Architecture (SCA), Service Data Object (SDO), Business Process Execution Language (BPEL) for web services, and Business Processing Modeling Notation (BPMN).

Both IBM Business Process Manager and Business Monitor build on the core capabilities of the IBM WebSphere® Application Server infrastructure. As a result, Business Process Manager solutions benefit from tuning, configuration, and best practices information for WebSphere Application Server and the corresponding platform Java virtual machines (JVMs).

This book targets a wide variety of groups, both within IBM (development, services, technical sales, and others) and customers. For customers who are either considering or are in the early stages of implementing a solution incorporating Business Process Manager and Business Monitor, this document proves a useful reference. The book is useful both in terms of best practices during application development and deployment and as a reference for setup, tuning, and configuration information.

This book talks about many issues that can influence performance of each product and can serve as a guide for making rational first choices in terms of configuration and performance settings. Similarly, customers who already implemented a solution with these products can use the information presented here to gain insight into how their overall integrated solution performance can be improved.

This IBM Redbooks publication replaces the existing IBM Redpaper™ publication *IBM Business Process Manager V8.0 Performance Tuning and Best Practices*, REDP-4935.

Authors

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Austin Center.

Mike Collins is a Senior Software Engineer at IBM. He is currently a lead on the IBM Business Process Manager performance team in the Austin lab, a position he has held for approximately 10 years. He has been a performance analyst at IBM for 17 years, focusing on the Java based middleware stack. Past performance analysis roles include the IBM JVM and JIT, IBM WebSphere Application Server, WebSphere InterChange Server, and the WebSphere Adapters. He holds a B.S. in Computer Science from Purdue University. Mike has authored several IBM Redpaper publications, notably the series of Business Process Manager Performance Tuning and Best Practices publications.

Zi Hui Duan is an Advisory Software Engineer at IBM. He is currently a performance analyst in the global IBM BPM performance team. Before he took this role, Zi Hui has about 8 years of experience in developing, system testing, and stress testing for IBM BPM products. He holds a Masters degree in information science from Peking University. His interest areas include program design, SOA, system integration, and performance tuning, and so on. He has published several publications related to these areas.

Andreas Fried is a Senior Software Engineer at IBM. In 2012, he joined the IBM Business Process Manager architecture team as the lead architect of the database access layer. He has been a core developer of the database access layer of Business Process Manager and WebSphere Process Server since 2006. Past development roles include the WebSphere Business Process Choreographer Samples Web Page, WebSphere Portal, and Web Services for Remote Protlets (WSRP). He holds an M.S. (Diplom-Informatiker) in Computer Science from the University of Stuttgart, and a B.S. (Diplom-Betriebswirt (B.A.)) in Economics of the Berufsakademie Heidenheim.

Ben Hoflich is an Advisory Software Engineer with IBM in Austin, Texas. He is a member on the IBM Business Process Manager performance team. Ben has 15 years of experience working as a performance analyst on multiple IBM products and has contributed to several related performance publications. He holds a B.S. degree in Computer Science from the University of Texas at Austin.

Chris Richardson is a Software Architect for High Availability and Disaster Recovery with IBM in Austin, Texas. Chris has held this position for the past 2 years. Before this, Chris had been a Lead Performance Analyst on the BPM Performance team for many years. Chris has 16 years of experience working with IBM, almost all of that time focusing on Software Systems Performance. Chris holds a B.S. degree in Physics from the University of Texas at Austin and an M.S. in Physics from the University of Washington in Seattle.

Torsten Wilms is a performance analyst at the IBM Business Process Manager performance team in the Boeblingen lab, Germany. After graduation in Business Information Systems, Torsten joined IBM in 2001. He started at the SAP Customer Competence Center at IBM as an IT Specialist. In 2005, he joined the Lab Based Service Team in the Business Process Management organization where his responsibility was to consult customers to design and implement Business Process Manager scenarios with focus on performance and high availability. In 2009, Torsten joined the Business Process Manager Quality Assurance team where he designed customer-related test scenarios. Since 2011, Torsten has been in the IBM Business Process Manager performance team. His responsibilities are to analyze and to drive performance improvements into the product. Torsten has authored several IBM Redbooks in the area of IBM WebSphere and SAP products.

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

► Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

► Send your comments in an email to:

  redbooks@us.ibm.com

► Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

► Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

► Follow us on Twitter:

  https://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

Overview

In this chapter, we introduce the products covered in this book, and the overall document structure.

1

1.1 Products covered in this publication

This book describes the following products:

► IBM Business Process Manager (IBM BPM) V8.5.0 and V8.5.5 (all editions).

This product combines simplicity, ease-of-use, and task-management capabilities that support enterprise integration and transaction process management requirements as part of an overall SOA solution. Business Process Manager adds value in the following ways:

– Optimizes business processes by providing visibility to all process participants, fostering greater collaboration and enabling continuous process improvement across distributed platforms and IBM System z®.

– Increases efficiency with a federated view for performing tasks, managing work items, tracking performance, and responding to events, all in real time.

– Empowers users with real-time analytics to optimize business processes.

– Enhances time-to-value through business user-focused design capabilities, including process coaching to guide users easily through the steps of a process.

– Confidently manages change with a unified model-driven environment that makes the same process version visible to everybody.

– Combines simplicity, ease-of-use, and task management capabilities for IBM WebSphere Lombardi Edition with key capabilities from IBM WebSphere Process Server (Advanced Edition only). These capabilities support enterprise integration and transaction process management requirements as part of an overall SOA.

– Offers compatibility with earlier versions for the latest versions of WebSphere Process Server (Advanced Edition only) and WebSphere Lombardi Edition.

– Also, starting in V8.5.5 Case Authoring and Case Management are integrated with BPM Authoring and Process Management.

► IBM Business Monitor V8.5.5

This product provides comprehensive business activity monitoring (BAM). This feature enables user visibility into real-time, end-to-end business operations, transactions, and processes to help optimize processes and increase efficiency. Business Monitor adds value in the following ways:

– Provides a high-performance BAM solution for processes and applications running in disparate environments, which might or might not be implemented using any Business Process Manager technology.

- Offers built-in tools and runtime support for integrated BAM for Business Process Manager.

- Integrates IBM Cognos® Business Intelligence Server V10.2.1 for advanced analysis and reporting on historical data.

- Automatically generates dashboards for your Business Process Modeling Notation V2.0 (BPMN2) processes. This feature allows real-time visibility to process instances, key performance indicators (KPIs), Cognos reports, and annotated BPMN2 process diagrams.

- Includes fine-grained security to enable or prevent anyone from seeing a wide range of information depth or detail.

- Enables enhanced business user customization of data filtering and dashboard controls and reports.

- Enables views of KPIs, metrics, and alerts through web interfaces, Apple iPad, mobile devices, and corporate portals.

- Is available for distributed platform and System z.

1.2 Publication structure

The following list summarizes each chapter of this document:

► Chapter 1, "Overview" on page 1.

This current chapter presents the scope of the content of this book.

► Chapter 2, "Architecture best practices" on page 5.

This chapter provides guidance for architecture and topology decisions that produce well-performing and scalable Business Process Manager solutions.

► Chapter 3, "Development best practices" on page 29.

This chapter presents guidelines for solution developers that lead to high-performing systems.

► Chapter 4, "Performance tuning and configuration" on page 63.

This chapter explains the tuning and configuration parameters and settings to optimize the major components in a Business Process Manager solution.

► Chapter 5, "Database configuration, tuning, and best practices" on page 111.

This chapter explains the tuning methodology and configuration considerations to optimize the databases that Business Process Manager uses.

► Chapter 6, "Migration considerations" on page 139.

This chapter discusses performance considerations for both Business Process Manager Version to Version migration, and application's process instance migration when updated versions of applications are deployed into production.

► Chapter 7, "IT monitoring and tuning for performance testing and managing a production environment" on page 157.

This chapter discusses the methodology used for IT monitoring to obtain the relevant metrics to tune a production environment, and to conduct a performance or load test. For the latter topic, IT monitoring, tuning, and performance/load testing are all intrinsically linked, so the topics are interwoven accordingly.

► Chapter 8, "Initial configuration settings" on page 179.

This chapter provides details about the software configurations, including specific parameter settings, used for representative workloads that are evaluated by the IBM performance team during the development of IBM Business Process Manager and Business Monitor.

► "Related publications" on page 187.

This chapter provides links to best practices, performance information, and product information for both the products described in this publication and related products such as IBM WebSphere Application Server, IBM DB2®, and other applications.

2

Architecture best practices

This chapter provides guidance for how to design a high-performing and scalable Business Process Manager solution. The purpose of this chapter is to highlight the best practices that are associated specifically with the technologies and features that are delivered in the Business Process Manager and Business Monitor products covered in this book. However, these products are based on existing technologies (such as WebSphere Application Server and DB2). Each of these technologies has its own associated best practices.

This book does not enumerate these best practices outside Business Process Manager. See "Related publications" on page 187 for reference information and links to these other technologies.

2.1 Top tuning and deployment guidelines

This chapter details architectural best practices for Business Process Manager V8.5 solutions. Development best practices and performance tuning and configuration are covered in subsequent chapters.

If you read nothing else in this document, read and adhere to the following key tuning and deployment guidelines. They are relevant in virtually all performance-sensitive deployments:

► Use high-performance disk subsystems. In virtually any realistic topology, you must have a server-class disk subsystem (for example, a RAID adapter with multiple physical disks) on the tiers that host the Business Process Manager and Business Monitor data stores to achieve acceptable performance. This guidance applies to all databases (Process Server, Process Center, Message Engine, Business Monitor, and others), and the Business Process Manager Process Server cluster members also. *We cannot overstate this point.* In many cases, performance is improved by several factors by using appropriate disk subsystems.

► Use the most current Business Process Manager and Business Monitor release, with the most current fix pack. IBM improves the performance, scalability, serviceability, and quality of the Business Process Manager product line with every release and fix pack. With the most current level, you can avoid encountering issues that were already resolved by IBM.

► Set an appropriate Java heap size to deliver optimal throughput and response time. Memory usage data that is obtained through the JVM's verbose garbage collection option (verbosegc) helps determine the optimal settings. Further information is available in 4.2.2, "Java memory management tuning parameters" on page 67.

► Tune your database for optimal performance. Correct tuning and deployment choices for databases can greatly increase overall system throughput. For example, set the buffer pool or cache size to a minimum of 2 GB. For more details, see Chapter 5, "Database configuration, tuning, and best practices" on page 111.

► Disable tracing and logging. Tracing and logging are important when debugging, but the resources to do so severely affects performance. More information is available in 4.6.1, "Trace and monitor considerations" on page 85.

► Configure thread pools to enable sufficient concurrency. This configuration is important for high-volume, highly concurrent workloads because the thread pool settings directly influence how much work the server can concurrently process. Important thread pools include the Default, Web Container, ORB,

and Event Manager thread pools. For more information, see "Configure thread pool sizes" on page 88.

▶ Use fast, high-bandwidth network connections. There is significant network activity for many Business Process Manager activities, so minimizing network latency and ensuring sufficient bandwidth is essential between the following items:

– Process Designer and Process Center
– Process Center and its database
– Process Center and online Process Servers
– Process Portal and Process Server
– Process Server and its databases

▶ For business processes that use Business Process Modeling Notation (BPMN), tune the bpd-queue-capacity and max-thread-pool-size parameters to achieve optimal throughput and scaling. For more information, see 4.4.2, "Tune the Event Manager" on page 76.

▶ For business processes that use Business Process Execution Language (BPEL), follow these guidelines:

– Where possible, use non-interruptible processes (also known as *microflows* or *short-running processes*) instead of long-running processes (also known as *macroflows* or *long-running processes*). Many processes need macroflows (for example, if human tasks are employed or a state must be persisted). However, a significant amount of performance resources is associated with macroflows. If some portion of the solution requires macroflows, separate the solution into both microflows and macroflows to maximize use of microflows. For details, see "Choose microflows where possible" on page 13.

– For task and process list queries, use composite query tables. Query tables are designed to produce excellent response times for high-volume task and process list queries. For details, see "Choose query tables for task list and process list queries" on page 14.

– Use Work Manager-based navigation to improve throughput for long-running processes. This optimization reduces the number of allocated objects, the number of retrieved objects that are from the database, and the number of messages sent for Business Process Choreographer messaging. For more information, see 4.7.1, "Tune Work Manager-based navigation for business processes" on page 94.

– Avoid using asynchronous invocations unnecessarily. Asynchronous invocation is often needed on the edges of modules, but not within a module. Use synchronous preferred interaction styles, as described in "Set the Preferred Interaction Style to Synchronous when possible" on page 53.

– Avoid overly granular transaction boundaries in Service Component Architecture (SCA) and BPEL. Every transaction commit results in expensive database and messaging operations. Design your transactions carefully, as described in 3.3.9, "Transactional considerations" on page 51.

2.2 Modeling and developing applications

This section describes best practices for modeling and developing Business Process Manager applications. Business Process Manager V8.5 Advanced Edition offers two authoring environments:

► *Process Designer* is used to model, develop, and deploy Business Processing Modeling Notation (BPMN) business processes and cases, each of which often involve human interactions. The Process Designer is the only authoring tool for Business Process Manager V8.5 Standard Edition.

► *Integration Designer* is used to build and implement services that are automated or start other services. These services include web services, enterprise resource applications, or applications that run in IBM CICS® and IBM IMS™, which exist in the enterprise. Integration Designer is also the tool to use for authoring BPEL business processes.

These authoring environments both interact with the *Process Center*, which is a shared repository and runtime environment. Two individuals with the following separate roles with skill sets work together to develop business process management (BPM) applications by using these environments:

► The *business author* is responsible for authoring all business processes. The business author is able to use services but is not interested in the implementation details or how they work. The business author uses Process Designer to create business process diagrams (BPDs) that optionally use Advanced Integration services (AISs) created by the integration programmer.

► The *integration programmer* is responsible for doing all of the integration work necessary to support the processes that the business author creates. For example, the integration programmer implements all the AISs and produces mappings between back-end formats and the requirements of current applications. The integration programmer uses Integration Designer.

The remainder of this section is organized based on user type, with separate sections describing common best practices and best practices for Process Designer (for business authors) and Integration Designer (for integration programmers).

2.2.1 Common best practices

This section outlines general guidelines for designing and configuring elements of Business Process Manager V8.5.

Choose the appropriate granularity for a process

A business process and its individual steps should have business significance and not try to mimic programming-level granularity. Use programming techniques such as plain old Java objects (POJOs) or Java snippets for logic without business significance.

Use events judiciously

The purpose of event emission in Business Process Manager V8.5 is business activity monitoring. Because event emission uses a persistent mechanism, it can consume significant processor resources. Use Common Base Events for events that have business relevance only. Do not confuse business activity monitoring and IT monitoring. The Performance Monitoring Infrastructure (PMI) is far more appropriate for IT monitoring. This topic is discussed further in Chapter 7, "IT monitoring and tuning for performance testing and managing a production environment" on page 157.

The following principles generally apply for most customers:

► Customers are concerned about the state of their business and their processes. Therefore, events that signify changes in state are important. For long-running and human activities, this change in state is fairly common. Use events to track when long-running activities complete, such as when tasks change state.

► For short-running flows that complete within seconds, it is sufficient to know that a flow completes, perhaps with the associated data. Distinguishing events within a microflow or general system service that are only milliseconds or seconds apart usually does not make sense. Therefore, two events (start and end) are sufficient for a microflow or straight through process.

2.2.2 Process Designer architecture best practices

This section presents best practices for the Process Designer.

Use fast connection between Process Designer and Process Center

The Process Designer interacts frequently with the Process Center for authoring tasks. For this reason, minimize network latency to provide optimal response

times. Place the Process Center in the same physical location as the Process Designer users. Also, place the Process Center in the same physical location as its database. If you cannot relocate the Process Center, you can remotely connect to the environment where the Process Center is physically located and use the Process Designer through that mechanism.

Minimize use of Service Tasks

Where possible, call system lane tasks with Service No Task because Service Tasks have significant performance costs. The potential performance benefit of executing Service Tasks asynchronously is usually far outweighed by the additional overhead of Service Tasks because the Process Server persists the state and context to the database at transition points. In particular, avoid patterns such as the following pattern:

```
JavaScript 1 → Service Task 1 → JavaScript 2 → Service Task 2
```

If possible, avoid Service Tasks altogether, or switch to the following example to minimize the number of times that state and context are persisted:

```
JavaScript 1 → JavaScript 2 → Service Tasks 1 and 2 combined
```

Further, when Service Tasks are defined, select **Delete upon completion** to avoid unnecessary growth in database tables, which can impact performance. This setting is enabled by default in BPM 8.5.5, but is not the default value in earlier releases.

Use searchable business variables judiciously

Business variables add more overhead to the Process Server, so limit business data searches to only those fields that need to be searchable in the Process Portal. As a general guideline, rarely are more than 10 searchable variables required in a BPD. If your implementation has more searchable variables than this, reevaluate your design and refactor as needed to reduce the number of searchable variables.

Manage variable usage

Variables are persisted to the database when execution contexts are saved, which happens fairly frequently (for example, when changing from BPD to service execution, and when running each coach). These persistence operations are expensive. Minimize the persistence cost in the following ways:

► Minimize the number of variables used.

► Minimize the size of each variable.

► Set variables, such as DB result sets, to null when they are no longer needed.

► Minimize the number and size of variables that are passed to each task.

- Minimize the number and size of variables that are bound to each coach.
- If industry-standard schemas are being used (for example, ACORD or HIPAA), recognize that these schemas contain many fields, and use only the variables that are required from the schema. If necessary, convert to or from the industry standard schema on the edges of the application to avoid unnecessary persistence cost in the BPD processing.

Turn off auto-tracking in business process diagrams if not required

Auto-tracking is enabled by default for BPDs. This capability is important for many BPDs because it enables the gathering, tracking, and reporting of key business metrics. However, an additional cost exists as a result of auto-tracking because the events are processed by the Performance Data Warehouse and persisted in the database. Disable auto tracking for BPDs that do not require tracking and reporting business metrics.

Also consider creating tracking groups to only track key business events, and then disable auto-tracking. This ensures that the events persisted are only those required for business metrics.

Dashboard and auto-tracking

In version 8.5, the Process Performance and Team Performance dashboards do not require auto-tracking or custom tracking groups. The Process Portal Index is utilized as the data source for these dashboards.

If auto-tracking is enabled, however, additional optional features in the dashboard are enabled:

- The Process Performance dashboard can automatically annotate the instance diagram with the traversed instance path, and the projected path for the instance will be based on the historically most typical path for the process.
- If timing Intervals are added to custom tracking groups, a tab will automatically get added to the Process Performance dashboard to display the timing intervals for the process.

Read more about auto-tracking and dashboards in the IBM Knowledge Center at:

http://www.ibm.com/support/knowledgecenter/#!/SSFTDH_8.5.0/com.ibm.wbpm
.wle.editor.doc/modeling/topic/critical_path_A.html

Avoid business process diagrams that run perpetually

Some business process diagrams (BPDs) run forever. Although certain business processes must run perpetually, design this type of BPD only if the capability is strictly required. BPDs that run perpetually continually poll for new events, which

uses server processor resources. Consider using other communication mechanisms (such as Java Message Service queues) instead of polling. If polling is necessary, use an undercover agent (UCA) instead of a BPD to do the polling. Also, disable auto-tracking for these BPDs to avoid excessive traffic to the Performance Data Warehouse.

Develop efficient coaches

This topic is discussed in detail in the IBM Redbooks publication *Leveraging the IBM BPM Coach Framework in your organization*, SG24-8210, available at the following location:

http://www.redbooks.ibm.com/redbooks.nsf/RedpieceAbstracts/sg248210.html?Open

Following are key points extracted from this book:

► Use a modern browser. This is essential to achieve optimal response times. JavaScript processing and rendering are done within the browser; performing these operations efficiently requires a browser designed for Ajax solutions.

► Minimize the number of network requests to the server by combining resources such as JavaScript and images into single files, and using coarse grained APIs to combine multiple server actions into one request.

► Maximize browser concurrency (for example, using multiple channels effectively) by loading css files before JavaScript files to resolve dependencies, and loading both css and JavaScript files directly from html files.

► Minimize the number and size of business objects bound to a coach.

► Minimize the number of boundary events, since these result in network requests to the server and database to persist state.

► Execute long running requests asynchronously to provide optimal user response time.

► Avoid large, complex coaches.

► Use the Table and Tabs control judiciously.

► Avoid large, repeating tables. Page the results instead.

► Always wire coaches to end nodes.

► Use custom visibility sparingly.

Minimize use of large JavaScript scripts

Avoid large JavaScript blocks because JavaScript is interpreted and therefore is slower to process than other compiled mechanisms such as Java code. Large JavaScript blocks can also produce very large Document Object Model (DOM)

trees, which are expensive for browsers to process and render. Finally, large JavaScript blocks are often indicative of too much logic being placed in the Business Process Manager layer.

As a general guideline, limit a JavaScript block to 50 lines. If your implementation exceeds this value, reevaluate the design and refactor the implementation to use smaller JavaScript blocks.

Avoid direct SQL access to internal Business Process Manager tables

SQL tools provide the capability to access any database table, including internal Business Process Manager tables such as, LSW_TASK, LSW_PROCESS, and so on. Avoid accessing internal Business Process Manager tables directly because these are internal tables and the definition and content of the tables may change in future Business Process Manager releases. Instead, use published Business Process Manager APIs, such as JavaScript and REST APIs.

It is also important to avoid storing internal Business Process Manager information in application-specific persistent storage because it is difficult to maintain consistency between the internal Business Process Manager persistent storage and the application's own persistent storage.

2.2.3 Integration Designer best practices

This section describes best practices for using Integration Designer.

Choose microflows where possible

Use macroflows only where required (for example, for long-running service invocations and human tasks). Microflows exhibit significantly improved performance at run time. A non-interruptible microflow instance is run in one Java Platform, Enterprise Edition transaction with no persistence of state. However, an interruptible macroflow instance is typically run in several Java Platform, Enterprise Edition transactions, requiring that state persist in a database at transaction boundaries.

Where possible, use synchronous interactions for non-interruptible processes. A non-interruptible process is more efficient than an interruptible process because it does not use state or persistence in the backing database system.

To determine whether a process is interruptible, in the Integration Designer, click **Properties** → **Details**. A process is interruptible if the **Process is long-running** check box is selected.

If interruptible processes are required for some capabilities, separate the processes so that non-interruptible processes can handle the most frequent scenarios and interruptible processes handle exceptional cases.

Choose query tables for task list and process list queries

Query tables are designed to provide good response times for high-volume task lists and process list queries. Query tables offer improved query performance in the following ways:

► Improved access to work items, reducing the complexity of the database query

► Configurable high-performance filters for tasks, process instances, and work items

► Composite query tables to bypass authorization through work items.

► Composite query tables that allow query table definitions reflecting information shown in task lists and process lists

For more information, see the following references:

► PA71: Business Process Manager Advanced - Query Table Builder

   http://www.ibm.com/support/docview.wss?uid=swg24021440

► Query tables in Business Process Choreographer in the IBM Business Process Manager 8.5 IBM Knowledge Center

   http://www.ibm.com/support/knowledgecenter/#!/SSFTDH_8.5.0/com.ibm.w bpm.bpc.doc/topics/c6bpel_querytables_composit.html

Choose efficient metadata management

This section describes best practices for metadata usage.

Follow Java language specification for complex data type names

Business Process Manager Advanced Edition allows characters in business object (BO) type names that are permissible in Java class names, the underscore (_) character, for example. However, the internal data representation of complex data type names uses Java types. As such, performance is better if BO types follow the Java naming standards because valid Java naming syntax requires no additional translation.

Avoid use of anonymous derived types in XML schema definitions

Some XML Schema Definition (XSD) features (restrictions on the primitive string type, for example) result in modifications to the type that require a new subtype to be generated. If these types are not explicitly declared, a new subtype (a derived type) is generated at run time. Performance is generally better if this situation can

be avoided. Avoid adding restrictions to elements of primitive type where possible. If a restriction is unavoidable, consider creating a new, concrete SimpleType that extends the primitive type to include the restriction. Then, XSD elements might use that type without degraded performance.

Avoid references to elements in one XML schema definition from another

Avoid referencing an element in one XSD from another. For example, if `A.xsd` defines an element, AElement (shown in Example 2-1), it might be referenced from another file, `B.xsd` (shown in Example 2-2).

Example 2-1   AElement XSD

```
<xs:element name="AElement">
    <xs:simpleType name="AElementType">
                    <xs:restriction base="xs:string">
                            <xs:minLength value="0" />
                            <xs:maxLength value="8" />
                    </xs:restriction>
    </xs:simpleType>
 </xs:element>
```

Example 2-2   AElement referenced from another file

```
<xs:element ref="AElement" minOccurs="0" />
```

This practice often performs poorly. It is better to define the type concretely and make any new elements use this type. Thus, `A.xsd` takes the form shown in Example 2-3.

Example 2-3   AElementType XSD

```
<xs:simpleType name="AElementType">
        <xs:restriction base="xs:string">
            <xs:minLength value="0" />
            <xs:maxLength value="8" />
        </xs:restriction>
</xs:simpleType>
```

The form that `B.xsd` takes is shown in Example 2-4.

Example 2-4   BElement XSD

```
<xs:element name="BElement" type="AElementType" minOccurs="0" />
```

Reuse data object type metadata where possible

Within application code, it is common to refer to types, for example, when creating a BO. It is possible to refer to a BO type by name, for example in the method `DataFactory.create(String URI, String typeName)`.

You can also refer to the type by a direct reference, such as in the method `DataFactory.create(Type type)`. In cases where a type is likely to be used more than once, it is faster to retain the type, for example, through `DataObject.getType()`, and reuse that type for future use.

Choose between business state machines and business processes

Business state machines (BSMs) provide an attractive way of implementing business flow logic. For some applications, it is more intuitive to model the business logic as a state machine, making the resultant artifacts easier to understand. However, a BSM is implemented using the business process infrastructure, so there is always a performance impact when choosing BSM over business processes.

If an application can be modeled using either BSM or business processes, and performance is a differentiating factor, choose business processes. Also, more options are available for optimizing business process performance than for BSM performance.

Minimize state transitions in business state machines

Where possible, minimize external events to drive state transitions in BSMs. External event-driven state transitions are costly from a performance perspective. In fact, the total time it takes to run a BSM is proportional to the number of state transitions that occur during the life span of the BSM.

For example, if a state machine transitions through states A → B → B → B → C (four transitions), making transitions through states A → B → C (two transitions) is twice as time-consuming. Also, automatic state transitions are much less costly than event-driven state transitions. Consider these principles when designing a BSM.

2.3 Topology

In this section, we provide information about choosing an appropriate topology for your solution.

2.3.1 Deploy appropriate hardware

It is important to pick a hardware configuration that contains the resources necessary to achieve high performance in a Business Process Manager environment. The following factors are key considerations in picking a hardware configuration:

► Processor Cores

Ensure that Business Process Manager servers (Process Servers and Process Centers) are installed on a modern server system with multiple processor cores. Business Process Manager scales well, both for single JVM servers through symmetric multiprocessing (SMP) scaling, and horizontally through clustering.

► Memory

Business Process Manager servers benefit from both a robust memory subsystem and an ample amount of physical memory. Ensure that the chosen system has server-class memory controllers and as large as possible L2 and L3 caches (optimally, use a system with at least a 4 MB L3 cache). Make sure that there is enough physical memory for all the combined applications (JVMs) that are expected to run concurrently on the system. For 64-bit JVMs (the recommended configuration), 4 GB per JVM is needed if the maximum heap size is 3 GB or less. Add additional physical memory for heap sizes larger than 3 GB. When using a 32-bit JVM, a rough guideline is 2 GB per JVM instance.

► Disk

Ensure that the systems hosting the message and data stores (The Process Server nodes, and the database tiers), and also the Process Server and Process Center cluster members, have fast storage. Fast storage typically means using Redundant Array of Independent Disks (RAID) adapters with writeback caches and disk arrays with many physical drives.

► Network

Ensure that the network is fast enough not to create a system bottleneck. For example, a dedicated 1 or 10 Gigabit Ethernet network is a good choice. Also, minimize latency across the topology (for example, between Process Designer and Process Center, between Process Center and the Process Center database, between Process Portal and Process Server, and between Process Server and the Process Server databases). Techniques to minimize network latency include physical collocation of hardware, and minimizing firewall separation between Business Process Manager components.

► Virtualization

When using virtualization such as IBM AIX® dynamic logical partitioning or VMware virtual machines, ensure sufficient physical processor, memory, and I/O resources are allocated to each virtual machine or logical partition (LPAR). Avoid over-committing resources.

2.3.2 Deploy local modules in the same server

If you plan to deploy modules on the same physical server, you can achieve better performance by deploying the modules to the same application server JVM. The server can then take advantage of this locality.

2.3.3 Best practices for Deployment Environments

See IBM Redbooks publication *Business Process Management Deployment Guide using Business Process Manager 8.5*, SG24-8175 for information about best practices for building a Deployment Environment. This book provides comprehensive guidance on selecting appropriate topologies for both scalability and high availability. You can retrieve this document from the following location:

http://www.redbooks.ibm.com/abstracts/sg248175.html

It is not the intent of this section to repeat content from that book. Rather, this book distills some of the key considerations when scaling up a topology for maximum performance.

Use the remote messaging and remote support deployment pattern

Use the remote messaging and remote support deployment environment pattern for maximum flexibility in scaling. For more information, see the section "Planning a network deployment environment" in the *IBM Business Process Manager Advanced Installation Guide* at the following location:

ftp://ftp.software.ibm.com/software/integration/business-process-manager/library/imuc_ebpm_dist_pdf.pdf

This topology (formerly known as the "Gold Topology") prescribes the use of separate clusters for applications and messaging engines. The topology also prescribes how clusters should be used for applications servers. This topology allows independent control of resources to support the load on each of these elements of the infrastructure.

> **Flexibility and cost:** As with many system choices, flexibility comes with some costs. For example, synchronous event emission between an application and the CEI server in this topology is a remote call, which is heavier than a local call. The benefit of this configuration is the independent ability to scale the application and support cluster. We assume that the reader is familiar with these kinds of system tradeoffs as they occur in most server middleware.

Process Portal indexing considerations

By default, there is one Process Portal index per node. On each node, at any given time, a single cluster member gets a lock on the index and performs indexing. All cluster members on the same node read from the node's index.

This configuration causes each cluster member to index all the cell's data. The end result is that each cluster member performs redundant work. When there are multiple nodes, consider to change the Process Portal index configuration so that there is a single shared index across nodes. In this case, only one of the cluster members across the entire cell gets a lock on the index and performs indexing. All cluster members across the cell read from the shared index. To perform this configuration, follow the instructions documented in the IBM Knowledge Center at the following location:

http://www.ibm.com/support/knowledgecenter/SSFPJS_8.5.0/com.ibm.wbpm.admin.doc/topics/ccfg_task_index.html

> **Important:** Put the shared Process Portal index on fast storage, and ensure there is a fast, low latency network connection between each cluster member and the cluster member where the shared Process Portal index is stored.

2.3.4 Evaluate service providers and external interfaces

One typical usage pattern for Business Process Manager V8.5 is as an integration layer between incoming requests and back-end systems for the business (target applications or service providers). In these scenarios, the throughput is limited by the layer with the lowest throughput capacity.

Consider the simple case where there is only one target application. The Business Process Manager V8.5 integration solution cannot achieve throughput rates higher than the throughput capacity of the target application. This inability to increase throughput applies regardless of the efficiency of the Business Process Manager V8.5 implementation or the size or speed of the system hosting it. Thus, it is critical to understand the throughput capacity of all target

applications and service providers and apply this information when designing the end-to-end solution.

Two key aspects of the throughput capacity of a target application or service provider are as follows:

► Response time, both for typical cases and exceptional cases

► Number of requests that the target application can process at the same time (concurrency)

If you can establish performance aspects of the target applications, you can calculate a rough estimate of the maximum throughput capacity. Similarly, if average throughput is known, you can also roughly calculate either one of these two aspects. For example, a target application that can process 10 requests per second with an average response time of one second, can process approximately 10 requests at the same time:

```
throughput / response time = concurrency
```

The throughput capacity of target applications is critical to projecting the end-to-end throughput of an entire application. Also, consider the concurrency of target applications when tuning the concurrency levels of the upstream Business Process Manager V8.5-based components. For example, if a target application can process 10 requests at the same time, tune the process server components that start this application. By tuning these components, the simultaneous request from Business Process Manager V8.5 at least matches the concurrency capabilities of the target.

Additionally, avoid overloading target applications because such configurations do not result in any increase in overall application throughput. For example, if 100 requests are sent to a target application that can process only 10 requests at the same time, throughput does not improve. However, throughput does improve by tuning in a way that the number of requests made matches the concurrency capabilities of the target.

Do not use synchronous invocation semantics when invoking a service provider that might take a long time to reply, either as part of mainline processing or in exception cases. Not using synchronous invocations avoids tying up the business process and its resources until the service provider replies.

2.4 Client environments

In this section, we document recommendations for optimizing Business Process Manager client environments, which include Process Portal, Process Designer, and Business Space solutions.

2.4.1 Optimize the topology

The performance of Asynchronous JavaScript and XML (Ajax) applications, such as those used in Business Process Manager, can be divided into a four-tiered model, as shown in Figure 2-1. Each tier must be optimized to deliver a high-performing solution. Several details for optimizing the topology are described later in this book, but a context for such a description is presented here.
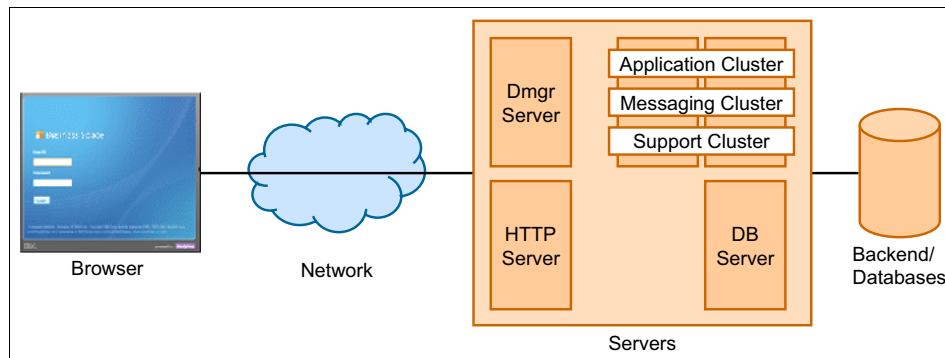
Figure 2-1   Four tiers of Ajax performance

The four tiers are browser, network, servers, and databases:

► Browser

 Ajax applications, by definition, perform work (to some extent) on the client side, inside the browser. All typical client work, such as building the user interface (UI), is done on the client side, which differentiates these applications from classical web applications, where only the rendering of HTML is done in the browser. Optimizing the browser and client system are key to delivering excellent performance (as described in 2.4.2, "Use a high-performing browser" on page 23 *through* 2.4.5, "Use modern desktop hardware" on page 23).

► Network

 In contrast to static web pages, Ajax applications load the content of a page dynamically as needed, instead of loading the complete page immediately.

Instead of one or several requests with significant payloads, Ajax applications often send many requests with smaller payloads. Therefore, delays in the network can significantly affect response times because they add time to each message. Ultimately, network delays can add up to the most significant factor for the overall page-response time.

Figure 2-1 on page 21 is simplified because the network also plays a role in communications between the servers and the databases and even between servers in a clustered setup. However, because of the nature of Ajax applications, the first point to analyze for delays is generally between the browser and the servers.

► Servers

The server infrastructure is responsible for handling the requests from the clients that are attached to it, running business applications (processes, state machines, web services, and other applications) and integrating back-end services. The configuration of this infrastructure heavily depends on the chosen topology.

The following servers play an important role for Business Process Manager solutions:

– HTTP server

An HTTP server is not part of all topologies. However, for environments that aim to serve thousands of users, an HTTP server is indispensable. Both static and dynamic requests from clients come to the HTTP server. This server can cache and then send the static (cached) content back to the client and route dynamic requests to the WebSphere Process Server REST API. Furthermore, an HTTP server can provide load balancing and support high-availability scenarios.

– Business Process Manager V8.5 servers (Process Server or Process Center)

Business Process Manager V8.5 servers execute various business logic (such as BPEL and BPMN business processes), querying task lists, creating and claiming tasks, and other functions. In addition, the Process Center executes authoring requests initiated by the Process Designer.

► Databases

There are multiple relevant databases depending on the client that is used (Process Designer, Process Portal, or Business Space). Each of these databases must be properly configured and tuned to support the anticipated load. See Chapter 5, "Database configuration, tuning, and best practices" on page 111 for a detailed description about this topic.

Use a network with low latency and high bandwidth. Each Business Process Manager client uses multiple HTTP requests per user action, so minimizing the

network delay per request is crucial. Network bandwidth is also crucial because some HTTP requests return a significant amount of data. Where possible, use a dedicated network that is both high-speed and high-bandwidth (1 Gb or faster) for connectivity between Business Process Manager clients and servers.

2.4.2 Use a high-performing browser

The choice of browser technology is crucial to Business Process Manager client performance. It is possible that more recent versions of a browser perform better than the older versions of the same browser.

2.4.3 Enable browser caching

Browsers generally cache static data after it is initially retrieved from the server, which can significantly improve response time for scenarios after the cache is primed. This improvement is especially true for networks with relatively high latency. Ensure that the browser cache is active and is effective. Cache settings are browser-specific; see 4.10, "Enable browser caching" on page 101 for further details.

2.4.4 Physical location of servers

One factor that influences network latency is the physical distance between servers and clients, and also the distance between the Business Process Manager servers in a cluster and their database (for example, between the Process Server and its databases). When practical, locate Business Process Manager V8.5 servers physically close to each other and to the Business Process Manager clients to minimize the latency for requests.

2.4.5 Use modern desktop hardware

For many Business Process Manager solutions, significant processing is done on the client system (for example, browser rendering). Thus it is imperative to deploy modern desktop hardware with sufficient physical memory and high-speed processors with large caches and fast front-side buses. Monitor your client systems with performance tools (Windows Task Manager or vmstat) to ensure that the client system has sufficient processor and memory resources to ensure high performance.

2.5 Large business objects

One issue that is frequently encountered by field personnel is identifying the largest object size that the Business Process Manager V8.5 server, corresponding document management functions, and corresponding WebSphere Adapters can effectively and efficiently process. A number of factors affect large business object processing in each of these products. This section presents both a description of the factors involved and practical guidelines for the current releases of these products. The issue of identifying the largest object size primarily applies to 32-bit JVMs because of the constraints on heap sizes in that environment.

The JVM is the single most important factor affecting large business object processing. Business Process Manager V8.5 uses the Java 6 JVM.

In general, objects that are 5 MB or larger might be considered "large" and require special attention. Objects of 100 MB or larger are "very large" and generally require significant tuning to be processed successfully.

2.5.1 Factors affecting large business object size processing

In general, the object size capacity for any installation depends on the size of the Java heap and the load placed on that heap (that is, the live set) by the current level of incoming work. The larger the heap, the larger the business object (BO) that can be successfully processed.

To apply this principle, you must first understand that the object size limit is based on three fundamental implementation facts about JVMs:

► Java heap size limitations

   The limit to the size of the Java heap depends on operating system, but it is not unusual to have a heap size limit of approximately 1.4 GB for 32-bit JVMs. The heap size limit is much higher on 64-bit JVMs and is typically less of a gating factor on modern hardware configurations than the amount of available physical memory. Further details about heap sizes are described in "Increase the maximum Java heap size" on page 86.

► Size of in-memory BOs

   BOs, when represented as Java objects, are much larger in size than when represented in wire format. For example, a BO that uses 10 MB on an input Java Message Service (JMS) message queue might result in allocations of up to 90 MB on the Java heap. This condition results from many allocations of large and small Java objects occurring as BO flows through the system.

A number of factors affect the in-memory expansion of BOs:

- The single-byte binary wire representation is generally converted to multi-byte character representations (for example, Unicode), resulting in an expansion factor of two.

- The BO might contain many small elements and attributes, each requiring a few unique Java objects to represent its name, value, and other properties.

- Every Java object, even the smallest, has a fixed need for resources because of an internal object header that is 12 bytes long on most 32-bit JVMs, and larger on 64-bit JVMs.

- Java objects are padded to align on 8-byte or 16-byte address boundaries.

- As the BO flows through the system, it might be modified or copied, and multiple copies might exist at any time during the end-to-end transaction. Having multiple copies of the BO means that the Java heap must be large enough to host all these BO copies for the transaction to complete successfully.

► Number of concurrent objects being processed

The size of an object that can be successfully processed decreases as the number of requests being processed simultaneously increases because each request has its own memory usage profile (live set) as it makes its way through the system. Simultaneously processing multiple large business objects dramatically increases the amount of memory required because the total of live sets for the request must fit into the configured heap.

2.5.2 Large object design patterns

The following sections describe the two proven design patterns for processing large objects successfully. In cases where neither design pattern can be applied, consider 64-bit mode. See 2.6, "Use 64-bit JVMs" on page 27 for details.

Also, for Mediation Flow Components, an IBM developerWorks® article is available that details best practices and tuning for large messages:

http://www.ibm.com/developerworks/webservices/library/ws-largemessaging

Batched inputs: Sending large objects as multiple small objects

If a large object must be processed, the solutions engineer must find a way to limit the number of allocated large Java objects. The primary technique for limiting the number of objects involves decomposing large business objects into smaller objects and submitting them individually.

If the large objects are actually collections of small objects, the solution is to group the smaller objects into conglomerate objects less than 1 MB. Several customer sites have consolidated small objects in this way, producing good results. If temporal dependencies or an all-or-nothing requirement for the individual objects exists, the solution becomes more complex. Implementations at customer sites demonstrate that dealing with this complexity is worth the effort as demonstrated by both increased performance and stability.

Certain WebSphere Adapters (such as the Flat Files adapter) can be configured to use a SplitBySize mode with a SplitCriteria set to the size of each individual object. In this case, a large object is split in chunks (of a size specified by SplitCriteria) to reduce peak memory usage.

Claim check pattern: Only a small portion of an input message is used

When the input BO is too large to be carried around in a system and that process or mediation needs only a few attributes, you can use a pattern known as the *claim check pattern*. Using the claim check pattern, as applied to a BO, involves the following steps:

1. Detach the data payload from the message.

2. Extract the required attributes into a smaller control BO.

3. Persist the larger data payload to a data store and store the "claim check" as a reference in the control BO.

4. Process the smaller control BO, which has a smaller memory footprint.

5. When the solution needs the whole large payload again, check out the large payload from the data store using the key.

6. Delete the large payload from the data store.

7. Merge the attributes in the control BO with the large payload, taking into account the changed attributes in the control BO.

The claim check pattern requires custom code and snippets in the solution. A less developer-intensive variant is to use custom data bindings to generate the control BO. This approach is limited to certain export and import bindings. The full payload still must be allocated in the JVM.

2.5.3 Data management

The use of document management functions, such as document attachments and the integration capabilities of content that is stored in Enterprise Content Management repositories, might result in large objects. The capacity that is required for processing of documents depends on the size of the Java heap and

the load that is placed on that heap (that is, the live set) by the current level of incoming work. The larger the heap, the larger the data that can be successfully processed.

Document attachments and content integration artifacts are stored in the Process Server database. Over time and depending on the size and number of documents, the database might grow in size. Completed process instances should be archived or deleted.

2.6 Use 64-bit JVMs

Wherever possible, use 64-bit JVMs for both Process Server and Process Center because in 32-bit mode, the maximum heap size is limited by the 4 GB address space size. In most 32-bit operating systems, the practical heap size maximum varies in the range of 1.5 GB and 2.5 GB. In contrast, although maximum heap size is essentially limitless in 64-bit mode, standard Java best practices still apply (for example, ensure that sufficient physical memory exists to back the heap).

The sum of the maximum heap sizes and native memory use of all the Java processes running on a system should not exceed the physical memory available on the system. This total also includes additional memory required for the operating system and other applications. Java processes include threads, stacks, and just-in-time (JIT) compiled code.

Business Process Manager V8.5 servers run most efficiently on a 64-bit JVM instance because of the much larger amount of memory that is accessible in this mode. The performance and memory footprint of a 64-bit runtime server is about the same as the 32-bit version.

2.7 Business Monitor

This section describes best practices for performance with Business Monitor V8.5.5.

2.7.1 Event processing

A major factor in event processing performance is tuning the Business Monitor database. Ensure adequate buffer pool sizes to minimize disk reading activity

and placement of the database logs, which ideally is on a physically separate disk subsystem from the database table spaces.

By default, events are now delivered using the table-based delivery style for better performance.

See 5.4, "Suggestions for Business Monitor" on page 137 for a full description about this topic.

2.7.2 Dashboard

The platform requirements of Business Space and the Business Monitor widgets on the dashboard are relatively modest compared to the Business Monitor server and the database server. The most important consideration for good dashboard performance is to size and configure the database server correctly. Be sure that it has enough processor capacity for anticipated data mining queries, enough memory for buffer pools, and plenty of disk arms.

Dashboard performance has been observed to be better with more modern browsers. Be sure to consider using the most current browser supported.

2.7.3 Database server

Both event processing and dashboard rely on a fast, well-tuned database server for good performance. The design of the Business Monitor assumes that any customer using it has strong onsite database administrator skills. It is important to apply the database tuning suggestions as described in Chapter 5, "Database configuration, tuning, and best practices" on page 111.

2.7.4 Topology

While a stand-alone or single cluster topology is fine for proof of concept or development purposes, for a golden topology for IBM Business Monitor v8.5.5 we recommend a three cluster topology (Application, Support, and Messaging). A minimum of two cluster members each for high availability is strongly recommended.

In this release, all JVMs are 64 bit, and allocating memory for each JVM can be tuned. The Support cluster does much the heavy processing and should have more memory allocated to it. We recommend a starting point around 3 GB.

Development best practices

This chapter presents best practices that are relevant to solution developers. It addresses modeling, design, and development choices that are made while designing and implementing a Business Process Manager V8.5 solution. Business Process Manager Advanced Edition offers two authoring environments. These authoring environments both interact with the Process Center, which is a shared repository and runtime environment:

► In the *Process Designer* environment, you can model, develop, and deploy Business Process Modeling Notation (BPMN) business processes and Cases, each of which often have human interactions. The Process Designer is the only authoring tool for Business Process Manager V8.5 Standard Edition.

► In the *Integration Designer* environment, you can build and implement services that are automated or that start other services, such as web services, enterprise resource applications, or applications running in CICS and IMS. These services and applications exist in the enterprise. It is also the tool to use to author Business Process Execution Language (BPEL) business processes.

Two individuals with separate roles and skill sets work together when developing Business Process Manager applications. These roles correspond to the Process Designer and Integration Designer environments:

► The *business author* is responsible for authoring all business processes and Cases. The business author is able to use services but is not interested in the

implementation details or how they work. The business author uses Process Designer to create business process diagrams (BPDs), and utilizes Advanced Integration services (AISs) to collaborate with the integration programmer.

► The *integration programmer* is responsible for doing all of the integration work necessary to support the processes that the business author creates. For example, the integration programmer implements all the AISs and produces mappings between back-end formats and the requirements of current applications. The integration programmer uses Integration Designer.

The remainder of this chapter is organized based on the type of author, with separate sections describing authoring BPMN Business Processes and Cases (business author) and Integration Designer (integration programmer) best practices. Additionally, the chapter provides developer considerations for browser environments, and for WebSphere InterChange Server migration.

3.1 BPMN business process authoring best practices

The following best practices pertain to the development of high-performance business processes using the Process Designer:

► Clear variables in exposed human services
► Do not use multi-instance loops in system lane or batch activities
► Use conditional joins only when necessary
► Follow guidelines for error handling
► Use sequential system lane activities efficiently
► Ensure the Process Center is properly tuned and managed
► Use fast connection for Process Designer and Process Center
► Prevent WSDL validation from causing slow web service integration

3.1.1 Clear variables in exposed human services

Data from a taskless human service is not garbage-collected until the service reaches the endpoint. If a human service is developed that is not intended to reach an endpoint, such as a single page or redirect, then memory is not garbage-collected until the Enterprise JavaBeans (EJB) timeout occurs (two hours by default). To reduce memory use for these human services, set variables in the coach to null in a custom HTML block.

3.1.2 Do not use multi-instance loops in system lane or batch activities

Where possible, avoid using sub-BPDs as the activity of a multi-instance loop (MIL). This step is not an issue if the first activity is a user task instead of a system lane task. However, do not use MILs for batch or system lane activities. This pattern can generate an excessive number of tokens for the BPD to process. Also, activities in MILs in the system lane are run on a single thread, which is clearly not optimal on multiprocessor core servers.

Figure 3-1 shows an example of a poor BPD design pattern.
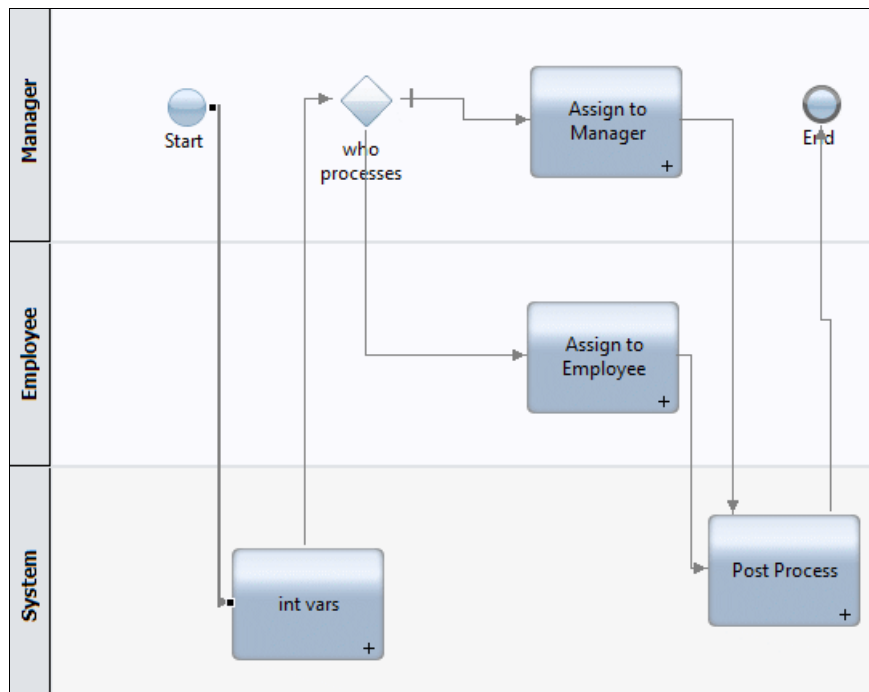
Figure 3-1   Poor BPD design pattern

Figure 3-2 shows an example of a good BPD design pattern.
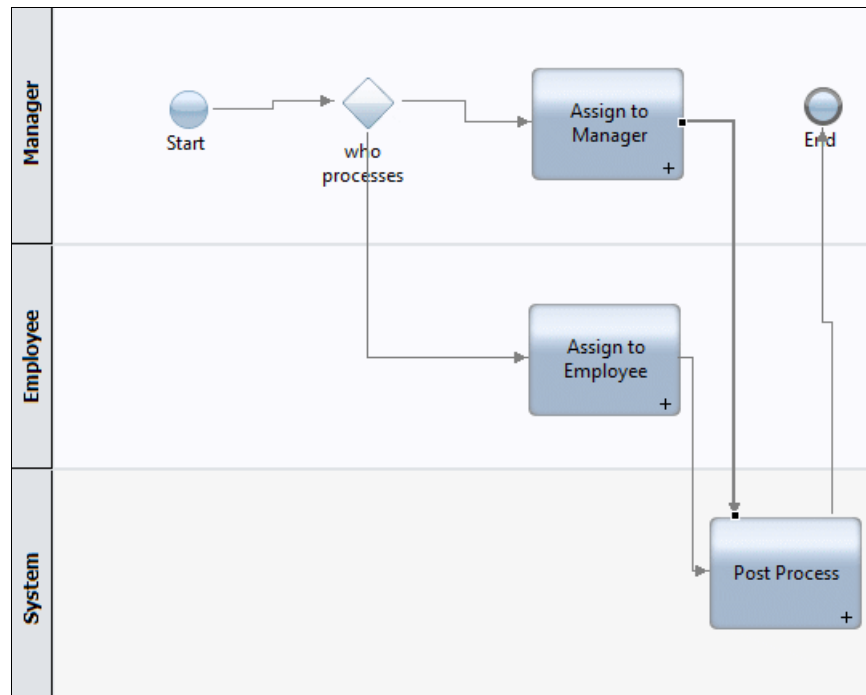
Figure 3-2   Good BPD design pattern

3.1.3 Use conditional joins only when necessary

Simple joins use an "and" condition; all lines that head into the join must have an active token for the tokens to continue forward.

By contrast, for conditional joins, all possible tokens must reach the join before they proceed. Thus, if you have a conditional join with three incoming lines, but only two of them currently have tokens (or $might$ have tokens by looking upstream), then those two tokens must arrive at the join to proceed. To determine this condition, the BPD engine must evaluate all possible upstream paths to determine whether the tokens can arrive at the join. This evaluation can be expensive for large, complex BPDs. Use this capability judiciously.

3.1.4 Follow guidelines for error handling

Avoid global error handling in a service, which can use an excessive amount of server processor utilization and can even result in infinite loops in coaches.

When catching errors on an activity in a BPD, do not route the error back to the same activity. Doing so causes the server to thrash between the BPD and service engine, using a large amount of Process Server processor time and also database processing.

3.1.5 Use sequential system lane activities efficiently

Each system lane activity is considered a new Event Manager task, which adds a task transition in the Process Server. These task transitions are expensive. If your BPD contains multiple system lane service tasks in a row, use one system lane task that wraps the others to minimize the extra resources needed for these transitions. Using one system lane task also applies for multiple consecutive tasks in a participant lane, although that pattern is much less common because generally an action is necessary between each task in a participant lane.

Figure 3-3 demonstrates a poor usage pattern (with multiple consecutive system lane activities).
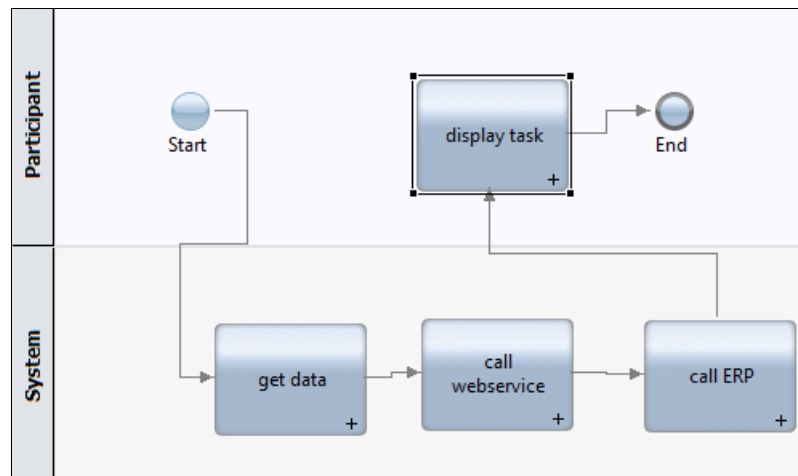
Figure 3-3   Poor BPD usage pattern

Figure 3-4 on page 34 shows a more optimal usage pattern (one system lane activity that incorporates the multiple activities in Figure 3-3).
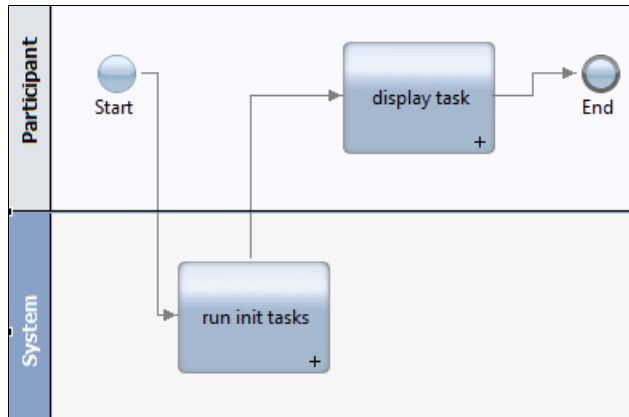
Figure 3-4   Good BPD usage pattern

3.1.6 Ensure the Process Center is properly tuned and managed

Because all Process Designer users access the Process Center, and for many cases use a significant amount of resources on the Process Center (and its database), optimizing its configuration is essential. See 4.5, "Process Center tuning" on page 84 for more information about this topic.

In addition, managing and tuning the Process Center repository is essential. As is the case for other BPM data stores, the database needs to be provisioned, monitored, and tuned. See Chapter 5, "Database configuration, tuning, and best practices" on page 111 for further information. In addition, the repository needs to be managed, particularly by keeping the snapshot population as low as is practical. One technique that is cruclal in this regard is to regularly delete unnamed snapshots via the provided utility. See 7.3.1, "Repository data" on page 174 for more information.

3.1.7 Use fast connection for Process Designer and Process Center

The Process Designer interacts frequently with the Process Center for authoring tasks. Take steps to expedite interaction between these components. See "Use fast connection between Process Designer and Process Center" on page 9 to learn more about optimizing the connection between Process Designer and the Process Center.

3.1.8 Prevent WSDL validation from causing slow web service integration

The Process Designer web service integration connector goes through several steps at run time to start a web service. First, the system generates the SOAP request from the metadata and the business objects (BOs). Then, the system validates the request against the Web Services Description Language (WSDL), makes the actual SOAP call over HTTP, and parses the results back into the BOs. Each of these steps potentially has some latency. Therefore, an important step is to make sure that the actual web service response time is fast and that the request can be quickly validated against the WSDL. Speed is especially important for web services that might be started frequently.

The two major causes of delays in validation are as follows:

► Slow responses in retrieving the WSDL
► Deeply nested WSDL includes structures

If the source of the WSDL is a remote location, the latency of retrieving that WSDL over HTTP adds to the overall latency. Thus, a slow connection, a slow proxy, or a slow server can all potentially increase the latency of the complete web service call. If that WSDL also nests additional WSDL or XML Schema Definition (XSD) files through imports or includes, then after the main WSDL is retrieved, the subfiles must also be retrieved. The validation continues to recurse through all WSDLs or XSDs that are nested within the subfiles. Therefore, in a case where there are multiple levels of nesting, many HTTP calls must make many calls to retrieve the complete WSDL document, and the overall latency can become high.

To alleviate this type of latency, you can store a local copy of the WSDL either in the local file system or somewhere where HTTP response is fast. For even better performance, this local copy can be "flattened," removing the nesting by manually replacing all of the include statements with the actual content.

3.2 Case authoring best practices

The following best practices pertain to the development of well-performing Case solutions through the Case Designer.

Notice: All BPMN Business Process authoring best practices apply to Case authoring.

3.2.1 Use required and automatically started case activities

When a case starts, all activities defined as required and automatically started with no precondition are intended to start in parallel. The first activity will start immediately in the caller's thread. For all other activities, an Event Manager task is created. In high volume workloads, this can lead to delayed activity starts.

Use activities that are defined as required and automatically started with no precondition judiciously.

3.2.2 Use case folder properties

A case folder property is a variable whose value is stored in the case folder. When the value of a case folder property is updated during runtime, it is saved to the case folder. In high volume scenarios, saving case folder properties can be expensive depending on the amount and frequency of the updates.

Carefully select which variables need to be defined as case folder properties.

3.3 Integration Designer best practices

The following best practices pertain to the development of well-performing solutions through the Integration Designer.

3.3.1 Use share-by-reference libraries where possible

Often, definitions of interfaces, BOs, data maps, mediation subflows, relationships, roles, and web service ports must be shared so that resources in several modules can use them. These resources are stored in the library. Libraries can be deployed in several ways:

► With a process application
► With the dependent module
► Globally

You can make your choice in the dependency editor. If you associate a library with a process application, then you select the application in the editor. The library is shared within the deployed process application, meaning only one copy is in memory. We call this type of library a *share-by-reference library*. The share-by-reference library is a feature that was introduced in Business Process Manager V7.5 Advanced Edition.

If you choose to deploy a library with the module, the deployment action creates a copy of the library for each module when the module is deployed. This type of library is known as a *share-by-value* library. Deploying the library with the module is the default setting when the library is not associated with a process application.

More details about Business Process Manager V8.5 library types can be found in "Libraries" in the IBM Knowledge Center for IBM Business Process Manager, Version 8.5, all platforms:

`http://www.ibm.com/support/knowledgecenter/#!/SSFTDH_8.5.0/com.ibm.wbpm`
`.wid.main.doc/newapp/topics/clibrary.html`

3.3.2 Ensure content in Toolkits is needed for multiple applications

Toolkits are copied into each process application that uses them. As such, ensure that a Toolkit's content is only artifacts that are needed by multiple process applications to reduce the size and complexity of the PC Repository. Include process application-specific content only in the process application itself.

3.3.3 Advanced Content Deployment considerations

Content included in a Process Application (PA) or Toolkit that is authored with the Integration Designer is considered Advanced Content. When the tip or snapshot of a PA or Toolkit is activated or deployed, Advanced Content is processed over a different path than Standard Content. Deployment of Advanced Content often takes much longer than Standard Content. This is because Advanced Content is packaged into Service Component Architecture (SCA) modules and libraries, and deployed as Business Level Applications (BLAs) and Java Platform, Enterprise Edition EARs on the Process Center or Process Server. Because each deployed BLA and enterprise archive (EAR) consumes a non-trivial amount of resource on the server (memory, disk space, processor cycles), Advanced Content deployments should be done the minimum number of times that is practical. Also, Snapshots and PA or Toolkit Tips that are no longer needed should be proactively cleaned up by deactivating and undeploying their Advanced Content. These topics are discussed further in the remainder of this section.

To check how many BLAs are installed on the server, use the administrative console on the Deployment Manager and go to **Applications** → **Application Types** → **Business-level applications.**

You will see the list of all installed BLAs, which includes customer PAs and Toolkits, and also Business Process Manager product applications. If there are

more than approximately 50 customer BLAs installed, consider how the number can be reduced. Snapshots can be deactivated and undeployed through the Process Center web page by going to the Snapshots page of the corresponding PA or Toolkit, as follows:

1. Access the Snapshots page

2. Click the name of the PA or Toolkit on the Process Apps or Toolkits tab of the Process Center web page.

3. Click the down arrow next to the Snapshot name.

4. If the Snapshot is Active and Deployed, click **Deactivate**.

5. Click **Undeploy**.

6. Confirm that the corresponding BLA was uninstalled through the deployment manager administrative console.

If you cannot remove a BLA through the Process Center web page, you can use the Deployment Manager administrative console to perform this action. Stop and uninstall the corresponding Java Platform, Enterprise Edition EARs, and delete the corresponding BLAs.

During iterative development of a solution, the amount of Advanced Content to manage can be minimized by integrating Advanced Content as late in the development cycle as is practical. The reason is because activating a snapshot, or deploying a tip, can take much longer when Advanced Content is involved. Thus, testing small incremental changes often becomes more time-consuming when a large amount of Advanced Content is involved. Also, Advanced Content should typically be relatively static after it is deployed because the implementation should meet the terms of a service level agreement (SLA) that was already agreed to as part of the solution development cycle. This recommendation should be balanced with testing needs and the overall schedule of the solution development cycle.

When designing the PA and Toolkit dependency graph, consider the server overhead that is attributable to Advanced Content. Include Advanced Content in Toolkits only if *every* PA that references the Toolkit needs all of the contained Advanced Content. Toolkits are copied "by value" into dependent process applications and Toolkits, so any Advanced Content will be duplicated on the server in every PA or Toolkit that references it. Including Advanced Content in Toolkits should be minimized because of the additional cost of Advanced Content.

Because deployment of Advanced Content uses BLAs and Java Platform, Enterprise Edition EARs, deployment can be disk-intensive. Deployment of Advanced Content occurs in multiple phases, with some processing and disk activity taking place on the deployment manager (dmgr) node, and some on each

AppTarget cluster member. The installation directory of the profiles for the dmgr and cluster members should be stored on fast disk subsystems, such as server-class hardware with RAID adapters with a write-back cache and multiple physical backing disks.

Deployment of Advanced Content can also be processor-intensive. For process servers, install snapshots with Advanced Content during periods of low activity so that users are not affected. For Process Center servers, ensure that the dmgr and cluster members are installed on servers with multiple processor cores. This enables the Process Center to remain responsive during periods of Advanced Content deployment. For clustered Process Centers, use a dynamic load balancer that can detect processor consumption on the cluster members to avoid sending work to cluster members that might be processing a long-running Advanced Content deploy operation.

3.3.4 Business object parsing mode considerations

This section describes how and why to choose the business object (BO) parsing mode. Two options for this mode exist: *eager* parsing mode and *lazy* parsing mode.

Overview

WebSphere Process Server V7.0.0.1 introduced BO lazy parsing mode. In this mode, when BOs are created, their properties are not populated until they are accessed by the application. Also, when reading XML input, the stream is incrementally parsed and its properties do not materialize until they are accessed.

In contrast, with eager parsing mode, the input XML stream is immediately and fully parsed (with some exceptions for specific mediation primitives) and materializes into a complete in-memory data object representation.

Lazy parsing uses the XML Cursor Interface (XCI) implementation of Service Data Objects (SDOs), provided by the WebSphere Application Server Feature Pack for SCA, which is delivered as part of Business Process Manager V8.5 Advanced Edition. In this mode, the input XML stream remains in memory throughout the lifetime of the corresponding BO. The cursor-based XML parser creates nodes in memory only when the nodes are traversed. Properties and attributes are evaluated only when the application requests them. Parsing the XML stream in this fashion delivers better performance than complete parsing for many applications in eager mode. This improvement is especially evident if only a small portion of the BO is accessed during the application execution.

Lazy parsing also reuses the XML stream when BOs are serialized, for example, for outbound service calls. The result is faster serialization because serializing the entire in-memory BO into XML strings is not necessary. Lazy parsing automatically detects namespaces in the output stream that differ from the namespaces in the original input stream, and corrects them. This form of serialization is used for all bindings and internal Business Process Manager runtime components, such as the Business Process Choreographer container, and when custom code calls BO serialization.

To further optimize performance, lazy parsing also employs a technology referred to as *copy-on-write*, where properties in BOs are lazily copied. When the copy operation initially takes place, the target BO points only to the node tree of the source BO. Lazy parsing copies no properties. Subsequent modifications of either the source or target BO triggers a split of the affected node in the BO tree, copying BO properties only as needed. Modifications occur in the local copy of the corresponding node. Copy-on-write is transparent to the application.

Set parsing mode
In Business Process Manager V8.5, the default parsing mode is the lazy mode for newly developed applications in Integration Designer V8.5. Applications migrated from WebSphere Process Server V7.0.0.3 and earlier releases continue to run in the eager parsing mode. However, for these older applications, you can manually enable lazy parsing mode through Integration Designer, which provides a property for each module and library to configure the XML parsing mode to either eager or lazy.

Figure 3-5 on page 41 shows the Integration Designer panel that the system displays during creation of a library. At this point, you can specify whether it can be included in an eager parsing or lazy parsing module, or both. The default is set to lazy parsing when the library is initially created. When both parsing modes are selected for the library, the module in which the library is located determines the actual parsing mode at run time.
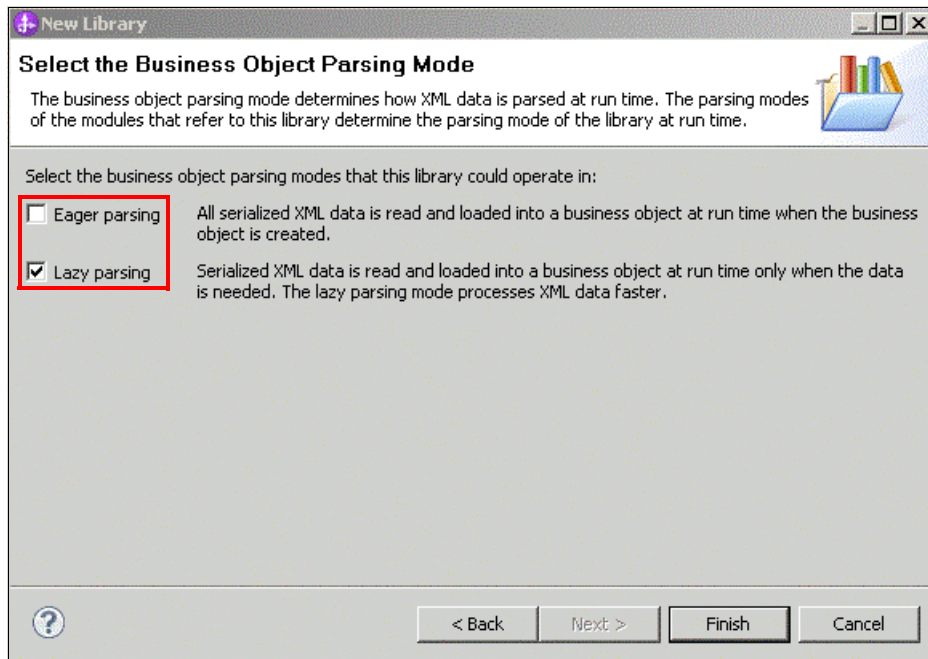
Figure 3-5   Lazy parsing for new library

When creating a module, you must choose eager or lazy parsing mode. The default mode is lazy parsing. Figure 3-6 on page 42 shows the Integration Designer panel for creating modules.
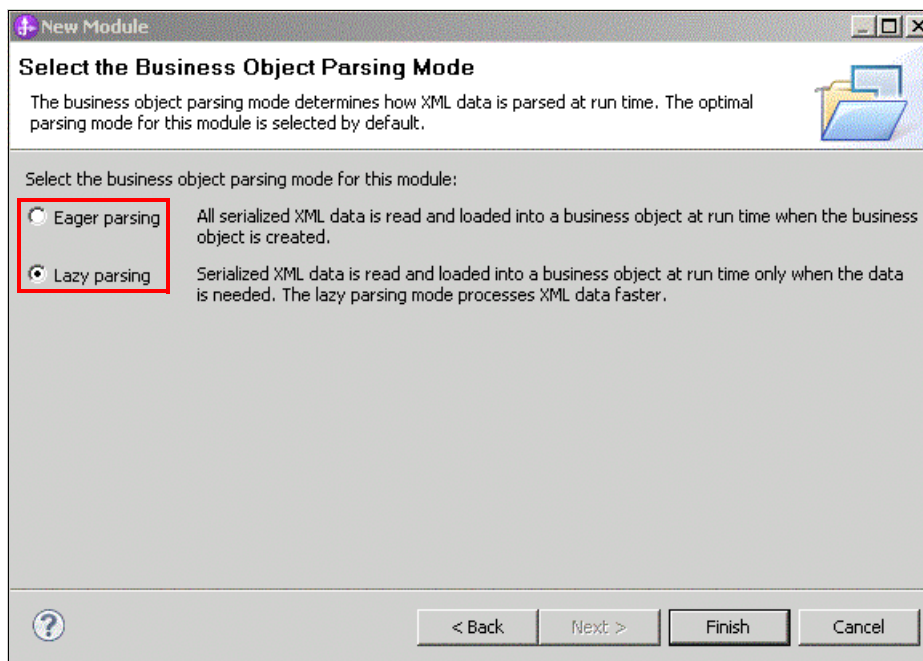
Figure 3-6   Lazy parsing for new module

In Business Process Manager V8.5, lazy parsing delivers excellent performance for XML-based applications that use BOs or messages approximately 3 KB or larger. The performance of lazy parsing is generally better than or about equal to eager parsing for other scenarios.

Applications benefiting from lazy parsing

This section provides general guidance for choosing the correct parsing mode, and best practices to follow when choosing one mode over the other. However, we suggest that you benchmark your application in both parsing modes to determine which mode best suits the specific characteristics of your application.

The primary performance benefits that are derived from lazy parsing are delayed parsing and parsing only the properties accessed by the application. Applications exhibiting any of the following characteristics are candidates for taking advantage of lazy parsing for better performance. The performance improvement when using lazy parsing can be significant:

► Only applications that use XML data streams can benefit from lazy parsing. For example, applications that use web services bindings, synchronous or asynchronous SCA bindings, or the Java Message Service (JMS) binding with the XML data handler can potentially benefit from lazy parsing mode.

► Applications with medium to large XML data streams are likely to see performance improvements when lazy parsing is used. The performance benefit increases as the size of the XML data stream increases. In the workloads we studied, lazy parsing performed better than eager mode when the BOs were approximately 3 KB or larger. The BO size differs depending on the composition of your application, so use these findings as a general guideline.

► Applications that access a small portion of an XML data stream are likely to see greater performance benefits from lazy parsing. Access to a BO is defined as reading the value of a property, updating a property value, or adding a property to a BO. Lazy parsing allows the unmodified portion of the data stream to pass through the system with minimal required resource usage. As opposed to eager parsing, the less an application accesses its BOs, the higher performance improvement it achieves by enabling lazy parsing.

► Mediation Flow Components (MFC) with multiple mediation primitives are likely to see performance improvements when you use lazy parsing. When using eager parsing, BOs are often serialized before starting the next mediation primitive and then deserialized after that mediation primitive is started. Lazy parsing mode reduces the cost of each serialization and deserialization by preventing unmodified properties from unnecessarily materializing in the data stream.

Applications that fall within these parameters likely benefit from lazy parsing. Applications that do not have those attributes produce little difference in performance and can run in either lazy or eager parsing modes. For example, applications that parse non-XML data streams, such as WebSphere Adapters and non-XML data handlers, might perform similarly, whether you choose lazy or eager parsing mode in Business Process Manager V8.5. This similarity is also true of applications that create BOs directly using the BOFactory service and applications that use small XML messages (as a general guideline, greater than 3 KB).

MFCs that access only the message header or contain a single mediation primitive, might see slightly better performance when using eager parsing. For example, a ServiceGateway with a single filter mediation routing based on the header content might perform better. However, adding multiple mediation primitives to a flow performance is likely to perform better in lazy parsing mode.

Minimize use of mixed parsing modes between modules
Sometimes it is necessary to direct some modules to use eager parsing while others use lazy parsing. For better performance, however, avoid using this mixed mode for application modules that frequently interact with each other. For example, if module A starts module B through the synchronous SCA binding on

each transaction, it is better to set modules A and B to both use either eager parsing or lazy parsing.

The interaction between modules with different parsing modes takes place through serialization and deserialization, which can negatively affect performance. If using a mixed mode is unavoidable, starting an application from an eager parsing module to a lazy parsing module is more efficient than the other way around. The output from the eager parsing module is serialized into an XML string, and the target module using lazy parsing still benefits fully from delayed parsing, producing this efficiency.

Share-by-reference libraries

Lazy parsing is optimized to use share-by-reference libraries. When a BO travels from one module to another as a parameter of a synchronous SCA service invocation, a lazy copy is used if the source and target modules share the BO schema through a share-by-reference library. Without the share-by-reference optimization, the caller serializes the BO and the callee deserializes the BO.

Differences in behavior for lazy parsing and eager parsing

If you are changing an application that was originally developed with eager parsing to use lazy parsing, be aware of the differences in behavior between each mode, as summarized here. Also, consider the behavior differences between each node if you are planning to switch an application between lazy and eager parsing mode.

Error handling

If the XML byte stream being parsed is ill-formed, parsing exceptions occur:

► With eager parsing, the exceptions occur as soon as the BO is parsed from the inbound XML stream.

► With lazy parsing, the parsing exceptions occur latently when the application accesses the BO properties and parses the portion of the XML that is ill-formed.

To properly handle ill-formed XML for either parsing mode, select one of the following options:

► Deploy a Mediation Flow Component on the application edges to validate inbound XML.

► Author lazy error-detection logic at the point where BO properties are accessed.

Exception stacks and messages

Because eager and lazy parsing have different underlying implementations, stack traces that are generated by the BO programming interfaces and services have the same exception class name. However, stack traces might not contain the same exception message or wrapped set of implementation-specific exception classes.

XML serialization format

Lazy parsing provides a fast serialization optimization that attempts to copy unmodified XML from the inbound data stream to the outbound data stream upon serialization. The fast serialization improves performance, but the resultant serialization format of the outbound XML data stream might be syntactically different than if the entire BO is updated in lazy parsing, or if eager parsing is used.

The output XML serialization format might not be precisely syntactically equivalent across these cases. However, the semantics of the XML data streams are equivalent, regardless of the parsing mode, and the resultant XML can be safely passed between applications running in different parsing modes with semantic equivalence.

Business object instance validator

The lazy parsing instance validator provides a higher fidelity validation of BOs, particularly for facet validation of property values. Because of these improvements in fidelity, the lazy parsing instance validator catches functional issues not detected in eager parsing mode and provides more detailed error messages.

Migration of XSL style sheets

When an application is authored using WebSphere Integration Developer V7.0.0.1 or earlier, the application is migrated using Integration Designer V8.5 and rebuilt. Extensible Stylesheet Language Transformation (XSLT) style sheets are regenerated from the associated map files. Extensible Style Sheet Language (XSL) style sheets built by Integration Designer V8.5 no longer contain white space or indent directives. This change improves performance in lazy parsing mode. However, if an XSLT primitive refers directly to manually edited style sheet, the style sheet is not regenerated when the application is built. In this case, performance improves by removing the white space directive (`<xsl:strip-space elements="*"/>`) if it appears, and by ensuring that indentation is disabled (`indent="no"`) unless required.

Private APIs

The BO programming model provides support for BO application programming interfaces (APIs) and a set of Service Data Object (SDO) APIs. In some cases,

applications might use additional implementation-specific APIs that are not part of the supported APIs for BOs. For example, an application might use the Eclipse Modeling Framework (EMF) APIs. Although these APIs might have worked in prior Business Process Manager releases and in eager parsing mode, they are not supported APIs for accessing BOs. EMF APIs are considered private to the implementation of BOs and should not be used by applications.

For these reasons, remove private APIs from the application and ensure that all BO access takes place using the supported API set.

Service Message Object Eclipse Modeling Framework APIs

A Mediation Flow Component makes it possible to manipulate message content using the Java classes and interfaces provided in the `com.ibm.websphere.sibx.smobo` package. For lazy parsing, the Java interfaces in the `com.ibm.websphere.sibx.smobo` package can still be used. However, methods that refer directly to Eclipse Modeling Framework (EMF) classes and interfaces or that are inherited from EMF interfaces are likely to fail. Also, the Service Message Object (SMO) and its contents cannot be cast to EMF objects when using lazy parsing.

Migration

All applications developed before Business Process Manager V7.5 used eager parsing mode, by default. When Business Process Manager runtime migration moves these applications, they continue to run in eager parsing mode.

To enable an application, that was originally developed using eager parsing, to use lazy parsing, first use the Integration Designer to move the artifacts of the application. After migration, use the Integration Designer to configure the application to use lazy parsing.

For information about moving artifacts in the Integration Designer, see "Migrating source artifacts" in the IBM Business Process Manager IBM Knowledge Center:

http://www.ibm.com/support/knowledgecenter/SSFPJS_8.5.0/com.ibm.wbpm.im uc.doc/topics/tmig_art_source_adv.html

For information about setting the parsing mode, see "Configuring the business object parsing mode of modules and libraries" in the IBM Business Process Manager IBM Knowledge Center:

http://www.ibm.com/support/knowledgecenter/SSFPJS_8.5.0/com.ibm.wbpm.wi d.main.doc/newapp/topics/tconfigbo.html

3.3.5 Service Component Architecture considerations

This section describes the SCA authoring considerations for Business Process Manager performance.

Cache results of ServiceManager.locateService()

When writing Java code to locate an SCA service, either within a Java component or a Java snippet, consider caching the result for future use, as service location is a relatively expensive operation. Code that is generated in Integration Designer does not cache the locateService result, so editing is required.

Reducing the number of Service Component Architecture modules when appropriate

When Business Process Manager V8.5 components are assembled into modules for deployment, many factors are involved. Performance is a key factor, but you must consider maintainability, version requirements, and module ownership also. In addition, more modules can allow for better distribution across servers and nodes. It is important to recognize that modularization also has a cost. When components are placed together in a single-server instance, package them within a single module for best performance.

Using synchronous Service Component Architecture bindings across local modules

For cross-module invocations, where the modules are likely to be deployed locally (that is, within the same server JVM), use the synchronous SCA binding because this binding is optimized for module locality and performs better than other bindings. Synchronous SCA is as expensive as other bindings when invocations are made between modules in separate Business Process Manager server instances.

Using multi-threaded Service Component Architecture clients to achieve concurrency

Synchronous components that are started locally (that is, from a caller in the same server JVM) run in the context of the thread of the caller. Thus the caller must provide concurrency, if wanted, in the form of multiple threads.

Adding quality of service qualifiers at appropriate level

You can add quality of service (QoS) qualifiers, such as Business Object Instance Validation, at the interface level or at an operation level within an interface. Because additional processor usage is associated with QoS qualifiers,

do not apply a qualifier at the interface level if it is not needed for all operations of the interface.

3.3.6 Business Process Execution Language business process considerations

This section presents considerations for authoring performance specific to BPEL business processes.

Modeling best practices for activities in a business process

For best performance, use the following guidelines when modeling a BPEL business process:

► Use the **Audit Logging property for Business Processes only** setting if you need to log events in the Business Process Choreographer database (BPEDB). You can set this property at the activity or process level. If you set it at the process level, the setting is inherited by all activities.

► For long-running processes, in Integration Designer, click **Properties** → **Server**, and clear the **Enable persistence and queries of business-relevant data** property for both the process and for each individual BPEL activity. Enabling this flag causes details of the execution of this activity to be stored in the Business Process Choreographer database, which increases the load on the database and the amount of data stored for each process instance. Use this setting only if you must retrieve specific information later.

► For long-running processes, a setting of **Participates** on all activities generally provides the best throughput performance. See "Avoid two-way synchronous invocation of long-running processes" below for more details.

► Human tasks can be specified in business processes (for example, process administrators), start activities, and receive activities. Specify human tasks only if necessary. When multiple users are involved, use group work items (people assignment criterion: Group) instead of individual work items for group members (people assignment criterion: Group Members).

Avoid two-way synchronous invocation of long-running processes

When designing long-running business process components, ensure that callers of a two-way (request/response) interface do not use synchronous semantics because this function ties up the caller's resources (such as threads and transactions) until the process completes. Instead, start such processes either asynchronously, or through a one-way synchronous call, where no response is expected. In addition, calling a two-way interface of a long-running business

process synchronously introduces difficulties when exceptions occur. Suppose that a non-interruptible process calls a long-running process using the two-way request/response semantics, and the server fails after the long-running process completes, but before the caller's transaction is committed. The following results occur:

► If the caller was started by a persistent message, upon server restart, the caller's transaction is rolled back and tried again. However, the result of the execution of the long-running process on the server is not rolled back because it was committed before the server failure. As a result, the long-running process on the server runs twice. This duplication causes functional problems in the application unless corrected manually.

► If the caller was not started by a persistent message, and the response of the long-running process was not submitted yet, the process ends in the failed event queue.

Minimize the number and size of Business Process Execution Language variables and business objects

Use the following guidelines when you define BPEL variables and BOs:

► Use as few variables as possible and minimize the size and the number of BOs used. In long-running processes, each commit saves modified variables to the database (to save context), and multiple variables or large BOs make this process costly. Smaller BOs are also more efficient to process when emitting monitor events.

► Specify variables as data type variables. This specification improves runtime performance.

► Use transformations (maps or assigns) to produce smaller BOs by mapping only fields that are necessary for the business logic.

3.3.7 Human task considerations

Follow these guidelines when developing human tasks for BPEL business processes:

► Use group work items for large groups (people assignment criterion: Group) instead of individual work items for group members (people assignment criterion: Group Members).

► Use native properties on the task object rather than custom properties where possible. For example, use the priority field instead of creating a custom property priority.

► Set the transactional behavior to `commit after` if the task is not part of a page flow. This setting improves the response time of task complete API calls.

3.3.8 Business process and human tasks client considerations

Consider the following information when developing effective BPEL business process and human task clients:

► Do not frequently call APIs that provide task details and process details, such as `htm.getTask()`. Use these methods only when required to show the task details of a single task, for example.

► Do not put too much work into a single client transaction.

– In servlet applications, a global transaction is typically not available. If the servlet calls the Human Task Manager (HTM) and Business Flow Manager (BFM) APIs directly, transaction size is typically not a concern.

– In Enterprise JavaBeans (EJB) applications, make sure that transactions are not too time-consuming. Long-running transactions create long-lasting locks in the database, which prevent other applications and clients to continue processing.

► Choose the protocol that best suits your needs.

– In a Java Platform, Enterprise Edition environment, use the HTM and BFM EJB APIs. If the client application is running on a Business Process Manager server, use the local EJB interface.

– In a Web 2.0 application, use the REST API.

– In an application that runs remote to the process container, the web services API is an option.

Client systems that follow a page-flow pattern should consider the following issues:

► Use the completeAndClaimSuccessor() API if possible. This API use provides optimal response time.

► Applications that assign the next available task to the user can use the following method on the Human Task Manager EJB interface. This method implements a performance-optimized mechanism to handle claim collisions.

`claim(String queryTableName, …)`

► Do not put asynchronous invocations between two steps of a page flow because the response time of asynchronous services increases as the load on the system increases.

► Where possible, do not start long-running subprocesses between two steps of a page flow because long-running subprocesses are started using asynchronous messaging.

Clients that present task lists and process lists should consider the following factors:

- ► Use query tables for task list and process list queries.

- ► Do not loop over the tasks shown in the task or process list and run an additional remote call for each object. This practice prevents the application from providing good response times and good scalability.

- ► Design the application so that all information is retrieved from a single query table during task list and process list retrieval. For example, do not make calls to retrieve the input message for task list or process list creation.

3.3.9 Transactional considerations

One of the strengths of the Business Process Manager BPEL platform is the precise control it provides for specifying transactional behavior. When modeling a process or mediation assembly, be sure that the modeler carefully designs the transaction boundaries as dictated by the needs of the application because boundaries are expensive in terms of system resources. The objective of this section is to guide the modeler to avoid unnecessary transaction boundaries. The following list details several guiding principles:

- ► The throughput of a particular usage scenario is inversely related to the number of transaction boundaries traversed in the scenario, so fewer transactions is faster.

- ► In user-driven scenarios, improving response time might require more granular transaction boundaries, even at the cost of throughput.

- ► Transactions can span across synchronous invocations but cannot span asynchronous invocations.

- ► Avoid synchronous invocation of a two-way asynchronous target. The failure recovery of a caller transaction can be problematic.

Take advantage of Service Component Architecture transaction qualifiers

In an SCA assembly, you can reduce the number of transaction boundaries by allowing transactions to propagate across components. For any pair of components where you want to reduce the number of transaction boundaries, use the following settings:

- ► For the reference of the calling component: `SuspendTransaction= false`
- ► For the interface of the called component: `joinTransaction= true`
- ► For implementation of both components: `Transaction= any|global`

These settings assume that the first component in such a chain either starts or participates in a global transaction.

Avoiding two-way synchronous invocation of an asynchronous target

If the target component must be started asynchronously and its interface is of two-way request/response style, then the target cannot be safely started through synchronous SCA calls. After the caller sends the request to the target, it waits for response from the target. Upon receiving the request, the asynchronous target starts a new transaction. Upon processing the request, the target returns the response asynchronously to the caller through the response queue. If a system failure occurs after the caller successfully sends the request but before receiving the response, the caller transaction is rolled back and tried again. As a result, the target is started a second time.

Take advantage of transactional attributes for activities in long-running processes

Although SCA qualifiers control component-level transactional behavior, additional transactional considerations in long-running business processes can cause activities to run in multiple transactions. You can change the scope of those transactions and the number of transactions using the transactional behavior settings on Java Snippet, Human Task, and start activities. For a detailed description of these settings, see "Transactional behavior of long-running BPEL processes" in the Business Process Manager 8.5 IBM Knowledge Center:

http://www.ibm.com/support/knowledgecenter/SSFPJS_8.5.0/com.ibm.wbpm.bpc.doc/topics/cprocess_transaction_macro.html

You have four setting choices for transactional behavior:

► Commit before

   Use the `Commit before` setting in parallel activities that start new branches to ensure parallelism. As noted in the IBM Knowledge Center, you must consider other constraints.

► Commit after

   Use `Commit after` for inline human tasks to increase responsiveness to human users. When a human user issues a task completion, the thread/transaction handling that action, is used to resume navigation of the human task activity in the process flow. The user's task completion action does not complete until the process engine commits the transaction.

   Starting with WebSphere Process Server 6.2.0, Receive and Pick activities in BPEL flow are allowed to define their own transactional behavior property values. If not set, the default value of initiating a Receive or Pick activity is `Commit after`.

Consider using `Participates` where possible because that behavior performs better:

► Participates

If the `Participates` setting is used, the commit is delayed and forces a longer response time for the user. Only the `Participates` setting does not require a new transaction boundary. The other three settings require the process flow container to start a new transaction before executing the activity, after executing the activity, or both.

In general, the `Participates` attribute provides the best throughput and should be used where possible. This suggestion is true for both synchronous and asynchronous activities. In the two-way asynchronous case, it is important to understand that the calling transaction always commits after sending the request. The `Participates` setting refers to the transaction started by the process engine for the response. When set, this setting allows the next activity to continue on the same transaction.

In special cases, transaction settings other than `Participates` might be preferable. Review the Business Process Manager 8.5 IBM Knowledge Center for more guidance about this issue.

► Requires own

The `Requires own` setting requires that a new transaction be started.

3.3.10 Invocation style considerations

This section explains invocation style considerations.

Use asynchronicity judiciously

Components and modules might be wired to each other either synchronously or asynchronously. The choice of interaction style can have a profound impact on performance. Exercise care when making this choice.

Set the Preferred Interaction Style to Synchronous when possible

Many Business Process Manager server component types (such as interface maps or business rules) start their target components based on the Preferred Interaction Style setting of the target interface. Because synchronous cross-component invocations perform better, set the Preferred Interaction Style to Synchronous when possible. Change this setting to Asynchronous only in specific cases. Such cases might include starting a long-running business process, or more generally, where the target component requires asynchronous invocation.

Starting with WebSphere Integration Developer V6.2 (now Integration Designer), when a new component is added to an assembly diagram, its Preferred Interaction Style is set to Synchronous, Asynchronous, or Any, based on the component. In previous releases of the Integration Designer (then called WebSphere Integration Developer), the default initial setting of Preferred Interaction Style was set to Any unless explicitly changed by the user. If the Preferred Interaction Style of a component is set to Any, how the component is started is determined by the caller's context. If the caller is a long-running business process, a Preferred Interaction Style setting of Any is treated as asynchronous. If the caller is a non-interruptible business flow, a Preferred Interaction Style setting of Any is treated as synchronous.

See "Take advantage of transactional attributes for activities in long-running processes" on page 52 for more information about the invocation logic of processes.

The following list details additional considerations for invocation styles:

► When the Preferred Interaction Style of an interface is set to Asynchronous, it is important to realize the downstream implications. Any components started downstream inherit the asynchronous interaction style unless they explicitly set the Preferred Interaction Style to Synchronous.

► At the input boundary to a module, exports that represent asynchronous transports such as IBM WebSphere MQ, JMS, or Java EE Connector Architecture (with asynchronous delivery set), set the interaction style to Asynchronous. This setting can cause downstream invocations to be asynchronous if the Preferred Interaction Style is Any.

► For an SCA import, you can use Preferred Interaction Style to specify whether the cross-module call should be `Synchronous` or `Asynchronous`.

► For other imports that represent asynchronous transports such as WebSphere MQ or JMS, it is not necessary to set the Preferred Interaction Style to Asynchronous. Doing so introduces an unnecessary asynchronous hop between the calling module and the invocation of the transport.

Minimize cross-component asynchronous invocations within a module

Asynchronous invocations are intended to provide a rich set of qualities of service, including transactions, persistence, and recoverability. Therefore, think of an asynchronous invocation as a full messaging hop to its target. When the intended target of the invocation is in the same module, a synchronous invocation yields better performance.

Some qualities of services (such as event sequencing and store-and-forward) can only be associated with asynchronous SCA calls. Consider the performance impact of asynchronous invocations when setting these qualities of service.

Performance considerations for asynchronous invocation of synchronous services in a FanOut/FanIn block

Do not select asynchronous (deferred response interaction) service invocations for services with synchronous bindings (for example, web services) unless there is an overriding need and the non-performance implications for this style of invocation are understood.

Apart from the performance implications of calling a synchronous service asynchronously, you must consider reliability and transactional aspects. Generally, use asynchronous callouts only for idempotent query type services. If you need to guarantee that the service is called only once, do not use asynchronous invocation. Providing complete guidance on the functional applicability of using asynchronous callouts in your mediation flow is beyond the scope of this book.

More information can be found in the Integration Designer help documentation and Business Process Manager V8.5 IBM Knowledge Center:

http://www.ibm.com/support/knowledgecenter/SSFPJS_8.5.0/com.ibm.wbpm.ma in.doc/ic-homepage-bpm.html

Assuming that asynchronous callouts are functionally applicable for your configuration, there might be a performance reason for starting a service in this style. However, understand that asynchronous processing is inherently more expensive in terms of processor cycles because of the additional messaging resources incurred by calling a service this way.

Additional operational considerations in choosing synchronous or asynchronous mode might apply. For example, asynchronous invocations use the service integration bus messaging infrastructure, which in turn uses a database for persistence. Synchronous invocations perform well with basic tuning of the JVM heap size and thread pools, but for asynchronous invocations, SCA artifacts require review and tuning. This requirement includes tuning the SCA messaging engine, data sources (see 4.2.6, "Java Database Connectivity data source parameters" on page 70), and the database itself. For the data source, the tuning for JMS bindings in this book can provide guidance as the considerations are the same.

If multiple synchronous services with large latencies are called, asynchronous invocations can reduce the overall response time of the mediation flow, but at the expense of increasing the internal response time of each individual service call.

This reduction of overall response time and increase of internal response time assume that asynchronous callouts are configured with parallel waiting in the FanOut section of the flow under the following conditions:

► With iteration of an array when configuring the FanOut to check for asynchronous responses after all or N messages are fired

► With extra wires or FlowOrder primitive (by default)

If a number of services in a FanOut section of a mediation flow exists, calling these services synchronously results in an overall response time equal to the sum of the individual service response times.

Calling the services asynchronously (with parallel waiting configured) results in a response time of at least the largest individual service response time in the MFC. The response time also includes the sum of the time taken by the MFC to process the remaining service callout responses on the messaging engine queue.

For a FanOut/FanIn block, the processing time for any primitives before or after the service invocations must be added in both cases.

To optimize the overall response time when calling services asynchronously in a FanOut/FanIn section of a mediation flow, start the services in the order of expected latency (highest latency first), if known.

You must consider the trade-off between parallelism and additional asynchronous processing. The suitability of asynchronous processing depends on the size of the messages being processed, the latency of the target services, the number of services being started, and any response time requirements expressed in service level agreements (SLAs). We suggest running performance evaluations on mediation flows, including FanOuts with high latency services, if you are considering asynchronous invocations.

The default quality of service-on-service references is `Assured Persistent`. You can gain a substantial reduction in asynchronous processing time by changing this setting to `Best Effort` (non-persistent). This change eliminates I/O to the persistence store, but the application must tolerate the possibility of lost request or response messages. This level of reliability for the service integration bus can discard messages under load and might require tuning.

3.3.11 Large object considerations

This section presents best practices for performance (particularly memory use) when using large business objects. Very large objects put significant pressure on Java heap use, particularly for 32-bit JVMs. 64-bit JVMs are less susceptible to

this issue because of the large heap sizes that you can be configure, although throughput, response time, and overall system performance can still be impacted by excessive memory usage. Follow these practices to reduce the memory pressure and avoid OutOfMemory exceptions and improve overall system performance.

Avoid lazy cleanup of resources

Lazy cleanup of resources adds to the live set required when processing large objects. If you can clean up any resources (for example, by dropping object references when no longer required), do so as soon as is practical.

Avoid tracing when processing large business objects

Tracing and logging can add significant memory resources. A typical tracing activity is to dump the BO payload. Creating a string representation of a large BO can trigger allocation of many large and small Java objects in the Java heap. For this reason, avoid turning on tracing when processing large BO payloads in production environments.

Also, avoid constructing trace messages outside of conditional guard statement. For example, the code in Example 3-1 creates a large string object even if tracing is disabled.

Example 3-1   Creating a large string object

```
String boTrace = bo.toString();
```

Although this pattern is always inefficient, it impacts performance even more if the BO size is large. To avoid unnecessarily creating a BO when tracing is disabled, move the string construction inside an if statement, as shown here:

```
if (tracing_on) System.out.println(bo.toString());
```

Avoid buffer-doubling code

Study the memory implications of using Java data structures that expand their capacity based on input (for example, StringBuffer and ByteArrayOutputStream). Such data structures usually double their capacity when they run out of space. This doubling can produce significant memory pressure when processing large objects. If possible, always assign an initial size to such data structures.

3.3.12 Mediation Flow Component considerations

This section describes Mediation Flow Component (MFC) considerations for performance.

Use Extensible Stylesheet Language Transformation primitives versus business object maps

The Extensible Stylesheet Language Transformation (XSLT) primitive offers two alternate transformation approaches in a mediation flow. If no XSLT-specific function is required, it is generally better to use the Business Object Map primitive, which can be faster. The exception is when a Mediation Flow Component is trivial in nature and the transformation is taking place at the `/body` level of the service message object (SMO). In this case, XSLT is faster because the native XML bytes can be passed straight to the XSLT engine. Native XML bytes are available if the XSLT transform primitive is the first in the flow or only preceded by one or more of the following primitives:

► Route on Message Header (Message Filter primitive)
► XSLT primitive (Transforming on `/body` as the root)
► EndpointLookup without XPath user properties.
► Event emitter (event header only)

Starting in Business Process Manager 8.0, the Integration Designer enhances the XML Mapper to use Business Object Maps for more cases, which results in better runtime performance. This is also the case for BPM 8.5. This enhancement is described further in the presentation at the following location:

http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/topic/com.ibm.iea.iid/iid/8.0/EnhancementsOverview/BPM80_IID_Enhancements.pdf

Aggregation blocks (FanOut and FanIn)

Aggregation is an important MFC pattern. It enables a single inbound request to map into multiple outbound service invocations, the responses from which can be aggregated into a single response to the original request. Before you develop mediations using aggregation design patterns, you need to consider several performance factors. See the developerWorks topic about best practices, "Aggregation design patterns and performance considerations in WebSphere Enterprise Service Bus V7.5" at the following location (this article is still relevant for BPM 8.5):

http://www.ibm.com/developerworks/websphere/library/techarticles/1111_norris/1111_norris.html

Configure Mediation Flow Component resources

When creating resources using Integration Designer, the application developer might choose to use preconfigured MFC resources or let the tool generate the mediation flow-related resources that it requires. Both approaches have their advantages and disadvantages.

Preconfigured resources help in the following circumstances:

▶ Existing resources are to be used

▶ External creation and tuning scripts are to be applied

Preconfigured resources also allow easier post-deployment adjustment.

Tooling-created resources are suitable if no further need exists for creating resources using scripts or the administrative console. Most performance tuning options can be changed because they are now exposed in the tooling.

In our performance tests, we used preconfigured resources because segregating the performance tuning from the business logic makes it possible to maintain the configuration for different scenarios in a single script.

3.4 Browser environment considerations

When using a browser for developing or evaluating Business Process Manager client solutions through the Process Portal or Business Space, developers often alter the configuration of the browser to perform debugging, tracing, and other browser functions. However, some of these configuration changes can dramatically affect the performance of the Business Process Manager solution. Before rolling out a solution to production or obtaining performance measurements, review this section and observe its guidelines.

Disable or uninstall add-ins or extensions for production use

An important consideration is using add-ins or extensions for your browser environment. If you experience performance problems while using the browser, make sure that you disable or uninstall all add-ins or extensions unless they are necessary.

Avoid Developer Tools render or compatibility mode in Internet Explorer

In Internet Explorer (IE), placing the browser in compatibility mode or in a render mode equivalent to an older browser version by using the Developer Tools is possible. This option can be useful for developing or debugging solutions (for example, making an IE 9 browser behave as an IE 8 browser for compatibility testing). However, make sure that you do not use these settings in production.

Use browser tools

The following observations pertain to browser tools that are available for obtaining response time measurements, counting requests, and analyzing caching:

- ► The Net tab in Firebug is useful for obtaining request timings, analyzing request/response headers, and counting the number of requests. However, it reports requests that are satisfied from the browser cache as code 200 responses. You can still determine that the request is cached from the Cache tab shown on the request, which indicates that the request was satisfied from the browser cache. If you copy and paste the results into another document (for example, into an email), the Cache tab is not copied. Thus, it is possible to be misled by the 200 responses and draw an incorrect conclusion that caching is not effective when it actually is.

- ► Fiddler is another powerful tool and has the advantage of supporting both IE and Firefox browsers. However, because Fiddler acts as a proxy between the browser and the server and cached requests are handled internally by browsers, these requests are never shown in Fiddler. This absence of result reporting prevents you from determining which requests are fulfilled from the browser cache, but Fiddler is still useful for analyzing the requests that actually are sent to the server.

- ► HttpWatch does not have the limitations of Fiddler because it is supported on both IE and Firefox browsers. Its results copy and paste easily into either a spreadsheet or a document, and it displays cached requests in a straightforward manner.

3.5 WebSphere InterChange Server migration considerations

The following considerations pertain to those migrating from WebSphere InterChange Server (WICS) to Business Process Manager V8.5:

- ► Migrated workloads using custom IBM WebSphere Business Integration (WBI) adapters or older WebSphere Business Integration Adapters result in interaction with Business Process Manager V8.5 through JMS, which is slower than the JCA adapters. Use WebSphere adapters to replace WebSphere Business Integration Adapters when possible.

- ► Some technology adapters (such as HTTP and web services) are migrated by the WebSphere InterChange Server migration wizard into native Business Process Manager V8.5 SCA binding, which performs better. For adapters that are not migrated automatically to available SCA bindings, development effort spent on migrating manually to an SCA binding removes the dependency on an older adapter and provides better performance.

- ► The WebSphere InterChange Server Migration wizard in Integration Designer offers a feature to merge the connector and collaboration module. Enable this

option, if possible, because it increases performance by reducing cross-module SCA calls.

► WebSphere InterChange Server collaborations are migrated into Business Process Manager V8.5 BPEL processes. You can further customize the resultant BPEL processes so they become more efficient.

– Migrated BPEL processes enable support for compensation by default. If the migrated workload does not use compensation, you can disable this support to gain performance. Find the relevant flag in Integration Designer by selecting the process name and then clicking **Properties** → **Details** → **Require a compensation sphere context to be passed in**.

– The generated BPEL flows still use the WICS APIs to perform BO and collaboration-level tasks. Development effort spent cleaning up the migrated BPEL to use Business Process Manager APIs instead of the ICS APIs results in better performance and better maintainability.

– You might be able to replace BPEL processes produced by migration with other artifacts. All WebSphere InterChange Server collaborations are currently migrated into BPEL processes. For certain scenarios, other Business Process Manager server artifacts (for example, business rules) might be better choices. Investigate the BPEL processes produced by migration to ensure that the processes are the best fit for your scenario.

► Disable Message Logger calls in migration-generated Mediation Flow Components (MFCs). The WebSphere InterChange Server Migration wizard in Integration Designer generates an MFC to deal with the mapping details of a connector. This MFC contains the code for handling synchronous and asynchronous calls to maps that transform application-specific BOs to generic BOs and generic BOs to application-specific objects.

The generated MFC contains embedded MessageLogger calls that log the message to a database. Disable these calls if they are not required in your business scenario to reduce writes to the database and thus improve performance. (Select the MessageLogger instance, choose the Details panel, and clear the **Enabled** check box.)

► Reduce memory pressure by splitting the shared library generated by the migration wizard. The migration wizard creates a single shared library and puts all migrated BOs, maps, and relationships into it. All the migrated modules share this library by copy. This sharing can cause memory bloat for cases where the shared library is large and many modules are present. The solution is to manually refactor the shared library into multiple libraries based on functionality or usage and modify modules to reference only the shared libraries that they need.

► If the original WebSphere InterChange Server maps contain many custom map steps, the development effort spent in rewriting those map steps results

in better performance. The WebSphere InterChange Server Migration wizard in Integration Designer V8.5 generates maps that use the ICS APIs at a translation layer above Business Process Manager V8.5 server technologies. Removing this layer by making direct use of Business Process Manager V8.5 server APIs avoids the cost of translation and produces better performance.

4

Performance tuning and configuration

To optimize performance, it is necessary to configure a system differently than the default settings. This chapter lists several areas to consider during system tuning, including Business Process Manager products and other products in the system. Documentation for each of these products can answer questions about performance, capacity planning, and configuration and offers guidance about achieving high performance in various operational environments.

> **Note:** Database tuning is also critical for high performing Business Process Manager solutions. Database tuning is discussed in Chapter 5, "Database configuration, tuning, and best practices" on page 111.

Several configuration parameters are available to the system administrator. Although this chapter identifies several specific parameters observed to affect performance, it does not address all available parameters. For a complete list of configuration parameters and possible settings, see the relevant product documentation.

The first section of this chapter is a basic tuning checklist that enumerates the major components and their associated tuning parameters. Subsections follow the checklist that address tuning in more detail, describing several tuning

63

parameters and their suggested settings (where appropriate) and providing advanced tuning guidelines for key areas of the system.

In addition, Chapter 8, "Initial configuration settings" on page 179 talks about important initial configuration settings and Chapter 7, "IT monitoring and tuning for performance testing and managing a production environment" on page 157 discusses the methodology for using IT monitoring tools to obtain the necessary data to iteratively adjust these parameters to achieve optimal performance. These chapters should be read in conjunction with this chapter because they are intrinsically related.

> **Important:** There is no guarantee that following the guidance in this chapter will provide acceptable performance for your application. However, if you set these parameters incorrectly, you can expect degraded performance.

4.1 Tuning checklist

This checklist serves as a guide when tuning a Business Process Manager solution. Each topic is covered in more detail in the remainder of this chapter.

► Common basic tuning considerations

– Disable tracing, logging, and monitoring when possible.

– Database performance is crucial for all Business Process Manager solutions, so ensure that all databases are well-tuned. For example, use a minimum of 2 GB for buffer pools or cache sizes, place logs and containers on separate physical disk subsystems, and define appropriate indexes.

– If security is required, use application security, not Java2 security.

– Use an appropriate hardware configuration for performance measurement (for example, notebooks and desktops are not appropriate for realistic server performance evaluations).

– If hardware virtualization is used, ensure that adequate processor, memory, and I/O resources are allocated to each virtual machine. Avoid overcommitting resources.

– Minimize network latency, and ensure sufficient network bandwidth, between all systems in the configuration, which includes the following items:

  • Between the Process Portal clients and the Process Server
  • Between the Process Server and its databases
  • Between the Process Designer clients and the Process Center

- Between the Process Center and its database
  - Do not run production servers in development mode or with a development profile.
  - Tune external service providers and external interfaces to ensure that they do not cause a system bottleneck.
  - Configure message-driven bean (MDB) activation specifications.
  - Configure thread pool sizes, including the Web Container, Default, ORB, and Event Manager (for Business Processing Modeling Notation (BPMN) solutions) and Work Manager (for Business Process Execution Language (BPEL) solutions) thread pools.
  - Configure settings of data sources for connection pool size and prepared statement cache size. Consider using non-XA data sources for Common Event Infrastructure data when that data is non-critical.
  - Increase the maximum number of connections in the data pool to greater than or equal to the sum of all maximum thread pool sizes.

► Business Processing Modeling Notation (BPMN) business processes
  - Set bpd-queue-capacity to 10 times number of physical processors, capped at 80.
  - Set the max-thread-pool size to 30 plus 10 times the number of physical processors, capped at 110.

► Business Process Choreographer (for Business Process Choreographer business processes)
  - Use Work Manager-based navigation for long running processes, and optimize the message pool size and intertransaction cache size.
  - Use query tables to optimize query response time.
  - Optimize Business Flow Manager resources:
    - Database connection (Business Process Choreographer database)
    - Activation specification (`BPEInternalActivationSpec`)
    - Java Message Service (JMS) data source connection (BPECF and BPECFC)
  - Optimize the database configuration for the Business Process Choreographer database (BPEDB).
  - Turn off state observers that are not needed (for example, audit logging).

► Messaging and message bindings
  - Optimize activation specification (JMS, MQJMS, WebSphere MQ).
  - Optimize queue connection factory (JMS, MQJMS, WebSphere MQ).
  - Configure connection pool size (JMS, MQJMS, WebSphere MQ).

– Configure service integration bus data buffer sizes.

► Java

– Set the heap and nursery sizes to manage memory efficiently.
– Choose the appropriate garbage collection policy (generally,
  `-Xgcpolicy:gencon`).
– Enable verbosegc to obtain Java memory statistics for later analysis.
  There is essentially no overhead attributable to enabling verbosegc.

► Business Monitor

– Configure Common Event Infrastructure.
– Set message consumption batch size.
– Enable key performance indicator (KPI) caching.
– Use table-based event delivery.
– Enable the data movement service.

4.2 Common tuning parameters

This section lists performance tuning parameters commonly used for tuning
Business Process Manager solutions, for both BPMN and BPEL business
processes.

4.2.1 Tracing and logging flags

Tracing and logging are often necessary when setting up a system or debugging
issues. However, these capabilities require performance resources that are often
significant. Minimize their use when evaluating performance or in production
environments. This section lists tracing parameters used in the products covered
in this book. Some settings are common to all or a subset of the products; others
are specific to a particular product. Unless stated otherwise, you can set all of
these parameters using the administrative console.

To enable or disable tracing, click **Troubleshooting** → **Logs and Trace** in the
properties of the subscription. Select the server on which you want to change the
log detail levels and click **Change Log Detail Levels**. Set both the
Configuration and **Runtime** fields to the following value:

`* =all=disabled`

To change the Performance Monitoring Infrastructure (PMI) level, click
Monitoring and Tuning → **Performance Monitoring Infrastructure**. Select the
server on which you want to change the log detail levels and click **none**.

In addition, Cross-Component Tracing (XCT) is useful for problem determination, enabling correlation of Service Component Architecture (SCA) component information with log entries. However, do not use XCT in production or while obtaining performance data. Two levels of XCT settings are possible:

► Enable
► Enable with data snapshot

Both incur significant performance resource usage. Enable with data snapshot is costly because of the additional I/O involved in saving snapshots in files.

To enable or disable XCT, click **Troubleshooting** → **Cross-Component Trace**. Select the **XCT** setting from three options under the Configuration or Runtime tab:

► Enable
► Disable
► Enable with data snapshot

Changes made on the Runtime tab take effect immediately. Changes made on the Configuration tab require a server restart to take effect.

Further information is provided in the "Managing Log Level Settings in TeamWorks" technote:

http://www.ibm.com/support/docview.wss?uid=swg21439659

4.2.2 Java memory management tuning parameters

This section lists several frequently used Java virtual machine (JVM) memory management tuning parameters. For a complete list, see the JVM tuning guide offered by your JVM supplier. For a more detailed discussion about JVM tuning for BPM applications, see 4.12, "Advanced Java heap tuning" on page 103.

To change the JVM parameters, complete the following steps:

1. Go to the JVM administrative window by first clicking **Servers** → **Application** → **Performance Monitoring Infrastructure**.
2. Select the server on which you want to change the JVM tuning parameters.
3. Click **Server Infrastructure** → **Java and Process Management** → **Process Definition** → **Additional Properties** → **Java Virtual Machine**.
4. Change the JVM parameters on this panel.

Java garbage collection policy

The default garbage collection (GC) algorithm on platforms with an IBM JVM is a generational concurrent collector (specified with `-Xgcpolicy:gencon` under the

Generic JVM arguments on the JVM administrative console panel). Our internal evaluation shows that this garbage collection policy usually delivers better performance with a tuned nursery size, as described in the next section.

Java heap sizes

To change the default Java heap sizes, set the initial heap size and maximum heap size explicitly on the JVM window in the administrative console. Use this capability to relieve memory pressure in the Java heap, but always ensure that there is sufficient physical memory to back the JVM heap size and all other memory requirements.

If you click **Generational Concurrent Garbage Collector**, the Java heap is divided into a new area (*nursery*), where new objects are allocated, and an old area (*tenured space*), where longer-lived objects are located. The total heap size is the sum of the new area and the tenured space. You can set the new area size independently from the total heap size. Typically, set the new area size in the range of 1/4 and 1/2 of the total heap size. The relevant parameters are as follows:

► Initial new area size: `Xmns<size>`
► Maximum new area size: `Xmnx<size>`
► `Fixed` new area size: `Xmn<size>`

4.2.3 Message-driven bean ActivationSpec

To access the MDB ActivationSpec tuning parameters, more than one shortcut is available in the administrative console:

► Click **Resources → Resource Adapters → J2C activation specifications**. Select the name of the application specification you want to access.

► Click **Resources → Resource Adapters → Resource adapters**. Select the name of the resource adapter you want to access. Then, click **Additional properties → J2C activation specifications.** Select the name of the activation specification you want.

The following custom properties in the ActivationSpec message-driven bean (MDB) have considerable performance implications. These properties are described further in "Tune message-driven bean ActivationSpec properties" on page 87.

► maxConcurrency
► maxBatchSize

4.2.4 Process Engine navigation thread pools

Navigation activities for both the BPEL Engine and the BPMN Engine are typically configured to use a Work Manager. To tune these thread pools, go to the Admin Console and select **Resources** → **Asynchronous beans** → **Work managers**. Select the Work Manager name and then click **Thread pool properties**.

You typically must tune the following Work Managers:

► DefaultWorkManager (BPEL)

► BPELNavigationWorkManager (BPEL)

► BPMEventManagerWorkManager (BPMN) - this is the default name of the Work Manager, configurable using the property `<was-work-manager>`

4.2.5 Java Message Service connection pool sizes

You can access the JMS connection factories and JMS queue connection factories from the administrative console in several ways:

► Click **Resources** → **Resource Adapters** → **J2C connection factories** and select the factory name.

► Click **Resources** → **JMS** → **Connection factories** and select the factory name.

► Click **Resources** → **JMS** → **Queue connection factories** and select the factory name.

► Click **Resources** → **Resource Adapters** → **Resource adapters** and select the resource adapter name (for example, SIB JMS Resource Adapter). Then, click **Additional priorities** → **J2C connection factories** and select the factory name**.**

► From the connection factory admin panel, click **Additional Properties** → **Connection pool properties**. Set the **Maximum connections** property for the maximum size of the connection pool.

4.2.6 Java Database Connectivity data source parameters

Data sources can be accessed through either of these paths:

► Click **Resources** → **JDBC** → **Data sources** and select the data source name**.**

► Click **Resources** → **JDBC Providers** and select the Java Database Connectivity (JDBC) provider name, followed by clicking **Additional Properties** → **Data sources** and selecting the data source name**.**

Connection pool size

The maximum size of the data source connection pool is limited by the value of the Maximum connections property, which can be configured by clicking the **Additional Properties** → **Connection pool properties** in the data source window.

Increase the maximum number of connections in the data pool to greater than or equal to the sum of all maximum thread pool sizes.

The following data sources typically must be tuned:

► BPMN data sources for the BPMN databases and associated message engine databases (this applies both to both Process Server and Process Center).

► Performance Data Warehouse (PDW) data sources for event tracking data emitted by BPMN business processes.

► The Common Database data source, which contains data for many aspects of Business Process Manager solutions, including the repository where solution artifacts are stored.

► Business Process Choreographer (BPC) data sources for the BPEDB and associated message engine database (for BPEL business processes).

► SCA application bus messaging engine data source.

► SCA system bus messaging engine data source.

Prepared statement cache size

You can configure the cache size of the data source prepared statement from the data source. Click **Additional properties** → **WebSphere Application Server data source properties**.

For BPEL business processes, ensure that each data source's prepared statement cache is sufficiently large. For example, set the BPEDB data source prepared statement cache size to at least 300.

4.2.7 Tuning the Virtual Member Manager LDAP cache

Tune the search results and attributes cache to improve the performance of Virtual Member Manager (VMM) searches.

Because the WebSphere Dynamic Cache (DynaCache) is used as the implementation of the virtual member manager cache, you can use the WebSphere DynaCache Monitor to monitor the virtual member manager cache usage and cache hits, as is described at the following links:

http://www.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.nd.multiplatform.doc/ae/tdyn_servletmonitor.html?cp=SSAW57_8.5.5&lang=en

http://www.ibm.com/developerworks/websphere/downloads/cache_monitor.html

The search results and attributes cache sizes and cache timeouts can be modified using the WebSphere Integrated Solutions Console as follows:

► Navigate to the WebSphere Integrated Solutions Console **Security** → **Global security**.

► Under Available realm definitions ensure that **Federated Repositories** is selected.

► Select **Configure**.

► Select the **LDAP Repository Identifier**.

► Select **Performance** under Additional Properties.

VMM attribute cache settings

Select the Cache attributes option and specify the maximum number of search attribute entries. This option enables WebSphere Application Server to save the LDAP entries so that it can search the entries locally rather than making multiple calls to the LDAP server. This cache contains attributes of an LDAP entity.

► Verify the optimum size of the cache by using the WebSphere DynaCache Monitor. The VMM Attribute Cache should be set to at least:

`entries = amountOfGroups + 2 * distinctUsersPerDay`

Each cache entry roughly consumes 1,6kb of the heap.

– Example 1: 1000 distinct users per day, belonging to 50 groups with 20 users each.

Set the maximum number of search attribute entries to:

`50 + 2 * 1000 = 2050`

- – Example 2: 5000 distinct users per day, belonging to 250 groups with 20 users each.

  Set the maximum number of search attribute entries to:

  ```
  250 + 2 * 5000 = 10250
  ```

► Select the *cache timeout* option that is associated with the *cache the attributes* option to specify the maximum number of seconds that the Application Server can save these entries. The recommendation is to set the cache timeout to 14400 seconds (4 hours). Depending on the frequency of changes to users and groups in LDAP, set the timeout to a higher or lower value.

VMM search results cache settings

Following are basic cache considerations:

► Select the *cache the search results* option and specify the maximum number of search result entries. This option enables WebSphere Application Server to save the results of a search inquiry instead of making multiple calls to the LDAP server to search and retrieve the results of that search.

This cache contains entries for user and group queries. There can be multiple entries for users and groups. Therefore, set the number of search result entries to a larger number than the number of concurrent users and groups. Verify the optimum size of the cache by using the WebSphere DynaCache Monitor. The VMM Search Results Cache should be set to at least:

```
entries = 3 * amountOfGroups + 6 * distinctUsersPerDay
```

Each Cache entry will roughly consume 1,6kb of the heap.

- – Example 1: 1000 distinct Users per day, belonging to 50 Groups with 20 Users each.

  Set the maximum number of search attribute entries to:

  ```
  3 * 50 + 6 * 1000 = 6150
  ```

- – Example 2: 5000 distinct Users per day, belonging to 250 Groups with 20 Users each.

  Set the maximum number of search attribute entries to:

  ```
  3 * 250 + 6 * 5000 = 30750
  ```

► Select the *cache timeout* option that is associated with the *cache the search results* option to specify the maximum number of seconds that the Application Server can save the results. The recommendation is to set the cache timeout to 14400 seconds (4 hours). Depending on the frequency of changes to users and groups in LDAP, set the timeout to a higher or lower value.

4.2.8 Messaging engine properties

Two messaging engine custom properties might affect the messaging engine performance:

- ► sib.msgstore.discardableDataBufferSize
  - – The property stays in the data buffer for best effort nonpersistent messages.
  - – The default value is 320 KB.
  - – After the buffer is full, messages are discarded to allow newer messages to be written to the buffer.
- ► sib.msgstore.cachedDataBufferSize
  - – The property stays in memory cache for messages other than best-effort nonpersistent.
  - – The default is 320 KB.

To access these properties in the console, use the following steps:

1. Click **Service Integration** → **Buses** and select the bus name.
2. Click **Messaging Engines** and select the messaging engine name.
3. Click **Additional properties** → **Custom properties**.

4.2.9 Run production servers in production mode

Business Process Manager servers can be run in development mode to reduce start time for the server by using JVM settings to disable bytecode verification and reduce just-in-time (JIT) compilation time. Do not use this setting on production servers because it is not designed to produce optimal runtime performance. Make sure to clear the **Run in development mode** check box for the server. This setting is found in the configuration window of the server in the administrative console. Click **Servers** → **Application Servers**. Click the server whose setting you want to change and click **Configuration**.

You might also create server profiles with production or development templates. Use production profile templates for production servers.

4.3 Process Portal tuning and usage

This section contains tuning advice and usage guidelines for the Process Portal.

4.3.1 Use a high-performing browser

The choice of browser technology is crucial to Process Portal performance. In many cases, more recent versions of the browser will perform much better than older versions.

4.3.2 Enable browser caching

Browsers generally cache static data after it is initially retrieved from the server, which can significantly improve response time for scenarios after the cache is primed. This improvement is especially true for networks with relatively high latency, or when a limited number of browser channels are available for communication with the server. Ensure that the browser cache is active and is effective. Note that cache settings are browser-specific; see 4.10, "Enable browser caching" on page 101 for further details.

4.3.3 Locate the Process Portal physically close to the Process Server

One factor that influences network latency is the physical distance between systems. Where practical, locate Business Process Manager V8.5 servers physically close to each other and to the Process Portal users to minimize the latency for requests.

4.3.4 Use the WORK tab to refresh the Task List

Instead of refreshing the entire Process Portal browser page through F5 or a similar mechanism, click the WORK tab on the Process Portal browser page. This refreshes only the tasks in the task list, and does not refresh the entire browser page.

4.3.5 Use modern desktop hardware

For many Business Process Manager solutions, much of the processing is done on the client system (for example, browser rendering and Client Side Human Services in Business Process Manager 8.5.5). Therefore, be sure to deploy modern desktop hardware with sufficient physical memory and high-speed processors with large caches and fast front-side buses. Monitor your client systems with performance tools (Windows Task Manager or `vmstat`) to ensure that the client system has sufficient processor and memory resources to yield high performance.

4.3.6 Disable or uninstall add-ins or extensions

An important consideration is using add-ins or extensions in your browser. If you experience problems, always disable or uninstall all add-ins or extensions unless they are necessary

4.3.7 Process Portal Dashboard

This section contains tuning tips for the Process Portal Dashboard.

Apply search filters

Consider to limit the data displayed in the process instances and task dashboards by applying search filters. This can improve the response time of the dashboard. Read the IBM Knowledge Center to determine how to apply search filters:

http://www.ibm.com/support/knowledgecenter/SSFPJS_8.5.0/com.ibm.wbpm.wl
e.widget.doc/topics/cport_search_dashboards.html?cp=SSFPJS_8.5.0

Purge completed instances

Purging completed instances can reduce the rendering time of the Team Performance Dashboard's task list for completed tasks and the Process Portal Index that the dashboards are based on.

More information about purging can be found in 7.3, "Purging Business Process Manager data stores" on page 173.

Gantt View page

The Gantt View page can be used to determine the projected path for an instance. There are several routes, or paths, that can be followed to complete a process. With projected path management enabled on the business process diagram (BPD), the projected path for an instance can be displayed on the process instance dashboard if distinct paths from the start to end nodes are found. If there are many path permutations through the process, due to many decision points or parallel paths, the projected path functionality likely has less business value but requires more processing. Considerable resources will be required to compute all the possible path permutations. For BPDs with many decision points or parallel paths, consider not enabling Projected Path Management in the first place. This can be configured in the BPD overview tab as described in the IBM Knowledge Center:

http://www.ibm.com/support/knowledgecenter/SSFPJS_8.5.0/com.ibm.wbpm.wl
e.widget.doc/topics/cport_search_dashboards.html

4.4 Business Processing Modeling Notation business process tuning

In addition to the common tuning guidance, this section provides tuning guidance that is specific to BPMN business processes. Most of this tuning applies to the Process Server; Tuning the Process Server's database is covered in 5.2, "Tuning and recommendations for BPMN workload" on page 130.

4.4.1 How to Change Configuration Settings for BPMN Processing

Many BPMN configuration settings are stored in xml files in the profiles directories. Values can be changed in `00Static.xml`, but for some installations a preferable approach for upgrade safety is to make such modifications in `100Custom.xml` to override the defaults.

These configuration files are read during server start so changes will require a server restart. The parsed output is written to the `TeamworksConfiguration.running.xml` file during server start. Therefore, to validate that your expected configuration changes actually are being used, confirm the value in this `TeamworksConfiguration.running.xml` file.

> **File is rebuilt at each start:** Making changes in `TeamworksConfiguration.running.xml` has no effect because this file is rebuilt on each server start.

4.4.2 Tune the Event Manager

To optimize throughput and scaling, it is often necessary to tune the Event Manager. These values should be set in the `100Custom.xml` file in the configuration directory for the Process Server. The specific configuration directory is as follows:

`%BPM%/profiles/<profileName>/config/cells/<cellName>/nodes/<nodeName>/servers/server1/process-center or process-server/config/system`

To change the values, directly edit that file. Here are several guidelines for tuning these parameters.

Queue capacity properties
The Event Manager queues are used for scheduling Event Manager tasks. These queues include tasks associated with BPD navigation, and also

synchronous queue and asynchronous queues for undercover agent (UCA) events.

> **Recommendation:** Throttling the Event Manager is done by decreasing the queue capacity.

The following setting specifies the maximum number of Event Manager tasks that are loaded into memory for each synchronous UCA queue.

`<sync-queue-capacity>10</sync-queue-capacity>`

The following setting specifies the number of Event Manager tasks that are loaded into memory for the asynchronous UCA queue.

`<async-queue-capacity>10</async-queue-capacity>`

The following setting specifies the number of Event Manager tasks that are loaded into memory for the business process definition (BPD) queue, which is an asynchronous queue that handles Event Manager tasks that execute BPDs, timers, system tasks, and decision tasks.

`<bpd-queue-capacity>40</bpd-queue-capacity>`

> **Recommendation:** To optimize throughput and scaling, start with a bpd-queue-capacity of 10 per physical processor core (for example, 40 for a 4-processor core configuration), with a maximum value of 80. This is not an absolute limit, but consider the potential impact on Process Server memory utilization if the value is increased beyond 80.

The following setting specifies the number of Event Manager tasks that are loaded into memory for the Event Manager internal system queue, for example, blackout calendar update tasks.

`<system-queue-capacity>10</system-queue-capacity>`

Thread pool properties

The thread pool properties designate the worker threads that are used by the Event Manager.

The following setting specifies whether the WebSphere Application Server Work Manager is designated for the thread pool. It is set to true by default. Changing this value to false is not recommended. This would disable visibility in IBM WebSphere Application Server resource monitoring tools. The Work Manager is configured on the administrative console at the cluster level.

`<use-was-work-manager>true</use-was-work-manager>`

> **Recommendation:** The total available IBM Business Process Manager database connections in the application server connection pool needs to be at least the sum of the maximum number of threads for all nodes in the App Target cluster. This includes at least twice the number of the maximum Event Manager worker threads, plus the number of web container threads.
>
> Use the following guidance to determine the number of Event Manager threads. If <use-was-work-manager> is true, add the number of threads configured in the work manager thread pool configuration. If <use-was-work-manager> is false, the maximum number of Event Manager threads is the sum of the BPD queue capacity plus the asynchronous queue capacity, plus the system queue capacity, plus the number of synchronous queues.

Note that the web container threads also need database connections.

See 5.2, "Tuning and recommendations for BPMN workload" on page 130 for a complete description of tuning the Process Server database.

The following setting specifies the WebSphere Application Server Work Manager that is used.

```
<was-work-manager>wm/BPMEventManagerWorkManager</was-work-manager>
```

> **Recommendation:** Throttling the Event Manager is done by decreasing the queue capacity. If you are using <was-work-manager> thread pool, the maximum thread pool size limits the concurrency of Event Manager threads. It should be large enough to support the queue's capacity. Set this parameter to the size of bpd-queue-capacity plus async-queue-capacity plus system-queue-capacity plus number of sync-queues.

The following setting specifies the minimum number of threads for the Event Manager engine thread pool, if the WebSphere Application Server Work Manager is not used.

```
<min-thread-pool-size>5</min-thread-pool-size>
```

The following setting specifies the maximum number of threads for the Event Manager engine thread pool, if the WebSphere Application Server Work Manager is not used.

```
<max-thread-pool-size>50</max-thread-pool-size>
```

> **Recommendation:** To optimize throughput and scaling, start with a max-thread-pool-size size of 30 plus 10 per physical processor core (for example, 70 for a 4-processor core configuration), with a maximum value of 110.

Task loader properties 11.8

The task loader is a dedicated thread of an Event Manager instance that loads Event Manager tasks into in-memory queues. The default parameter settings of the task loader are optimized for most usage patterns. Typically there is no need to tune any of these parameters. However, they are described below for completeness.

The following setting specifies the number of milliseconds between major loads of the task loader. For every task loader long period, the Event Manager looks at each synchronous and asynchronous queue that it has access to and fills them to capacity. This setting should be less than or equal to the setting of <loader-advance-window>.

```
<loader-long-period>15000</loader-long-period>
```

The following setting specifies the number of milliseconds between minor loads of the task loader. For every task loader short period, the Event Manager looks through each of the queues that the Event Manager filled to capacity during the last short or long period, and fills them to capacity. Do not change this setting.

```
<loader-short-period>2000</loader-short-period>
```

The following setting specifies the number of milliseconds before execution that the task loader can acquire a task. For scheduled tasks, this parameter specifies how far in advance the Event Manager looks for tasks. This setting should be equal to <loader-long-period>.

```
<loader-advance-window>15000</loader-advance-window>
```

The following setting specifies when a new task is scheduled. If set to true (the default), the task loader is kicked into an immediate major load anytime a new task is scheduled, resulting in newly scheduled tasks executing almost immediately. Do not change this setting.

```
<kick-on-schedule>true</kick-on-schedule>
```

Heartbeat properties

A heartbeat is a dedicated thread of an Event Manager instance that periodically signals that the Event Manager instance is still running. Do not change these settings.

The following setting specifies the number of milliseconds between heartbeats.

`<heartbeat-period>30000</heartbeat-period>`

The following setting specifies the number of milliseconds for a heartbeat to expire.

`<heartbeat-period>.<heartbeat-expiration>240000</heartbeat-expiration>`

Reaper property

A reaper is a dedicated thread for each cluster member that checks if there are Event Manager tasks owned by expired Event Manager instances in any cluster member. If it finds those tasks, the reaper resets the state to scheduled and the ownership to unassigned. Therefore, other Event Manager instances will pick up the work.

This setting specifies the number of milliseconds between checks by the reaper for expired event manager instances.

`<reaper-period>120000</reaper-period>`

Other Event Manager properties

The following setting allows the Event Manager to be started in paused state. The Event Manager can be resumed using the Event Manager monitor in the Process Admin Console.

`<start-paused>false</start-paused>`

The following setting specifies the name of the Event Manager instance. The name is used in the Event Manager monitor in the Process Admin.

`<name>EM instance name</name>`

4.4.3 Tune Participant Groups

If you are not using the external email capability (which is enabled by default), turn it off through the following setting in the `100custom.xml` file in the profiles directory. This is especially important for applications using participant groups with many users.

`<send-external-email>false</send-external-email> (default is true)`

This step reduces load on the Process Server and its database by eliminating unnecessary email lookups.

4.4.4 Use fast disk subsystem for recovery logs and Lucene Task Search indexes

In addition to using a fast disk subsystem (for example a RAID adapter with 10 or more physical disks) for databases, using a fast disk subsystem on the Process Server cluster members (App Targets) is also essential. Several disk I/O operations are performed directly on the Process Server system, including compensation logging, transaction logging, profile operations (that is, Administrative activities), and maintaining the Lucene Task Search indexes.

4.4.5 Remove unnecessary snapshots from the Process Server

Over time, unused snapshots can accumulate on the Process Server. This can be the result of many factors, including iterative development resulting in unused snapshots as new versions of a solution are deployed, and obsolete snapshots that were part of a development cycle and were never removed when the solution went into production. These unused snapshots can degrade performance, particularly if they contain Advanced Content. The `BPMDeleteSnapshot` utility should be used to remove unnecessary snapshots from the Process Server. Read more at the following link:

http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r5m0/index.jsp?topic=%2Fco m.ibm.wbpm.ref.doc%2Ftopics%2Frref_deletesnapshot.html

In addition to this information, perform periodic maintenance on the snapshots that are deployed on the Process Server. This maintenance includes deleting unused Process Applications and Toolkits, and archiving unneeded snapshots.

This topic is discussed in more detail in 7.3, "Purging Business Process Manager data stores" on page 173.

4.4.6 Disable notifications and automatic task list refreshes if they are not required

When running at high throughput rates, the overhead of processing notifications and automatic task list refreshes can become a Process Server scaling bottleneck. Specifically, the PortalNotificationTopicSpace can become contended and overrun (notifications can be lost). If notifications or automatic task list refreshes are not a business requirement, they can selectively be turned off on the Process Server by setting the following values in the `100custom.xml` file in the appropriate profile subdirectory; this step eliminates the bottleneck previously described.

To disable both notifications and automatic task list refreshes, set the following parameters, shown in Example 4-1.

Example 4-1   Disable both notifications and automatic task list refreshes

```
<server merge="mergeChildren">
    <web-messaging-push merge="replace" match="elementName" enabled="false">
    <web-messaging type="NOTIFY_TASK_RESOURCE_ASSIGNED" enabled="true"/>
    <web-messaging type="NOTIFY_TASK_COLLABORATION_INVITE" enabled="true"/>
    <web-messaging type="NOTIFY_PROCESS_COMMENT_TAGGED" enabled="true"/>
    <web-messaging type="TASKLIST_TASK_RESOURCE_ASSIGNED" enabled="true"/>
    <web-messaging type="TASKLIST_TASK_FIELD_CHANGED" enabled="true"/>
    </web-messaging-push>
</server>
```

To disable only notifications, set the following parameters, shown in Example 4-2.

Example 4-2   Disable only notifications

```
<server merge="mergeChildren">
    <web-messaging-push merge="replace" match="elementName" enabled="true">
    <web-messaging type="NOTIFY_TASK_RESOURCE_ASSIGNED" enabled="false"/>
    <web-messaging type="NOTIFY_TASK_COLLABORATION_INVITE" enabled="false"/>
    <web-messaging type="NOTIFY_PROCESS_COMMENT_TAGGED" enabled="false"/>
    <web-messaging type="TASKLIST_TASK_RESOURCE_ASSIGNED" enabled="true"/>
    <web-messaging type="TASKLIST_TASK_FIELD_CHANGED" enabled="true"/>
    </web-messaging-push>
</server>
```

To disable only task list auto refresh, set the following parameters, shown in Example 4-3.

Example 4-3   Disable only task list auto refresh

```
<server merge="mergeChildren">
    <web-messaging-push merge="replace" match="elementName" enabled="true">
    <web-messaging type="NOTIFY_TASK_RESOURCE_ASSIGNED" enabled="true"/>
    <web-messaging type="NOTIFY_TASK_COLLABORATION_INVITE" enabled="true"/>
    <web-messaging type="NOTIFY_PROCESS_COMMENT_TAGGED" enabled="true"/>
    <web-messaging type="TASKLIST_TASK_RESOURCE_ASSIGNED" enabled="false"/>
    <web-messaging type="TASKLIST_TASK_FIELD_CHANGED" enabled="false"/>
    </web-messaging-push>
</server>
```

4.4.7 Tune cache parameters

Several cache settings can be changed through configuration parameters in the following XML configuration files in the profiles directories:

- ► `00Static.xml`
- ► `99Local.xml`
- ► `100Custom.xml`

Process Server cache tuning

Several cache settings that might benefit from larger values in a production Process Server, as distinct from the development Process Center.

The following configuration settings control the number of objects in the PO caches. Two configuration flags in the `00Static.xml` file can manage this value:

- ► `<default-unversioned-po-cache-size>500</default-unversioned-po-cache-size>`
- ► `<default-versioned-po-cache-size>500</default-versioned-po-cache-size>`

For low volume environments with relatively few process applications and coaches, the default value may be sufficient. However, for more complex environments with many process applications or coaches, increase this value so that the process applications and coaches are held in the cache after their initial use. This step can improve response time when accessing these process applications and coaches. For example, increase each of these values to 1500.

When the cache is "cold" after a restart, initial coach load times might take longer until the cache is populated.

The effectiveness of these caches can be monitored through the Instrumentation Page in the Process Admin console, which indicates how many cache hits and misses occurred. For more detailed analysis, obtain Instrumentation logs from this page, which can be analyzed by IBM Support through a PMR to determine whether further cache-size increases are warranted.

The UCP column in the Manage Caches page of the Process Admin console can also be used to monitor how effectively the cache is working for a particular runtime server. The Process Admin console has online help that describes the column headings and their meanings.

User group membership information is also stored in a cache so a company with a large LDAP user and group population may require a larger setting for LDAP caches to avoid a performance impact when user/group information is accessed, such as during authorization validation for task assignment, login to Process Portal, or refreshing the Task List in Process Portal.

Cache settings for memory-constrained environments

In a memory-constrained environment, a beneficial approach might be to reduce the size of some caches. An example is the number of snapshots that are cached for a single branch in the Process Center. For very large process applications, reducing this value can reduce JVM heap memory usage. Reduce the following value to accomplish this goal:

```
<snapshot-cache-size-per-branch>64</snapshot-cache-size-per-branch>
```

Similarly, for a Process Server in a memory-constrained environment, reducing the Branch Manager cache size can reduce heap memory for large process apps. The Branch Manager cache contains metadata about the contents of snapshots in memory, and is used to improve performance of certain operations. It is controlled by the configuration flag:

```
<branch-context-max-cache-size>64</branch-context-max-cache-size>
```

The default value is 64. If process applications are particularly large, reducing this value might be necessary, particularly for runtime servers where a new branch is created for each deployed snapshot.

4.5 Process Center tuning

Because all Process Designer users access the Process Center, and for many cases use a significant number of resources on the Process Center (and its database), optimizing its configuration is essential. At a minimum, tune the Process Center in the following ways:

► Ensure that there is a high-bandwidth, low-latency network connection between Process Designer users and the Process Center, and the Process Center and its database.

► Use the generational concurrent garbage collector through the following argument:

```
-Xgcpolicy:gencon JVM
```

► Ensure that the Java heap size is sufficiently large to handle the peak load. For more information about setting the Java heap size, see 4.2.2, "Java memory management tuning parameters" on page 67.

► Set the object request broker (ORB) thread pool size to at least 50.

► Set the maximum Java Database Connectivity (JDBC) connections for the Process Server data source (`jdbc/TeamWorksDB`) to at least double the maximum number of threads in the ORB thread pools for each node in the cell. For example, if two Process Center nodes exist, and the ORB service

thread pool of each node is set to a maximum of 50 threads, set the Process Server data source to accept at least 200 connections.

► Tune the Process Center database. For example, ensue the buffer pool or cache size is at least 2 GB. For more information, see Chapter 5, "Database configuration, tuning, and best practices" on page 111.

► Manage the Process Center database by regularly purging unnamed snapshots. See 7.3.1, "Repository data" on page 174 for further information.

4.6 Advanced tuning

This section describes various advanced tuning tips.

4.6.1 Trace and monitor considerations

The ability to configure tracing and monitoring at different levels for various system components is valuable during periods of system analysis or debugging. The Business Process Manager product set provides rich monitoring capabilities, both in terms of business monitoring and audit logging, and system performance monitoring through the PMI and the Application Response Measurement (ARM) infrastructure. Although these capabilities provide insight into the performance of the running solution, these features can degrade overall system performance and throughput.

Tracing and monitoring effect on performance: Use tracing and monitoring judiciously. When possible, turn off all non-essential tracing and monitoring to ensure optimal performance.

Most tracing and monitoring behaviors are controlled through the administrative console. Validate that the appropriate level of tracing and monitoring is set for the PMI monitoring, logging, and tracing settings through the administrative console.

Use the administrative console to validate that the **Audit logging** check box is cleared in the Business Flow Manager and the Human Task Manager, unless these capabilities are required for business reasons.

For business processes, Integration Designer is also used to control event monitoring. Check the Event Monitor tab for your components and business processes to ensure that event monitoring is applied judiciously.

For BPMN business process definitions (BPDs), auto-tracking is enabled by default. If business requirements do not require tracking events for a BPD, turn

off auto-tracking. If event tracking is required, consider creating a tracking group to define a small set of events for tracking purposes.

4.6.2 Tune for large objects

This section describes tuning for performance when you use large business objects.

Increase the maximum Java heap size

One of the key factors affecting large object processing is the maximum size of the Java heap. For comprehensive heap setting techniques, see 4.12, "Advanced Java heap tuning" on page 103.

Reduce or eliminate other processing while processing a large business object

One way to allow for larger object sizes is to limit concurrent processing within the JVM. Do not expect to process a steady stream of the largest objects possible concurrently with other Business Process Manager server and WebSphere Adapters activities. The operational assumption when considering large objects is that not all objects are large or very large and that large objects do not arrive often, perhaps only once or twice per day. If more than one very large object is being processed concurrently, the likelihood of failure increases dramatically.

The size and number of the normally arriving smaller objects affect the amount of Java heap memory consumption in the system. In general, the heavier the load on a system when a large object is being processed, the more likely that memory problems are encountered.

4.6.3 Tune for maximum concurrency 11.8

For most high-volume deployments on server-class hardware, many operations present themselves to take place simultaneously. Tuning for maximum concurrency ensures that the server accepts enough load to saturate its core. One indication of an inadequately tuned configuration is when additional load does not result in additional core use while the cores are not fully used. To optimize these operations for maximum concurrency, the general guideline is to follow the execution flow and remove bottlenecks one at a time.

Higher concurrent processing means higher resource requirements (memory and number of threads) on the server. High concurrent processing must be balanced with other tuning objectives, such as the handling of large objects, handling large numbers of users, and providing good response time.

Tune edging components for concurrency

The first step in tuning edging components for concurrency is to ensure that business objects are handled concurrently at the edge components of Business Process Manager solutions. If the input business objects come from the adapter, ensure that the adapter is tuned for concurrent delivery of input messages.

If the input business objects come from WebServices export binding or direct invocation from JavaServer Pages (JSPs) or servlets, make sure the WebContainer thread pool is sized right. For example, to allow for 100 in-flight requests to be handled concurrently by a Business Process Manager server, the maximum size of the WebContainer thread pool must be set to 100 or larger. When setting the thread pool size, also factor in other WebContainer usage, such as inbound service calls.

If the input business objects come from messaging, you must tune the ActivationSpec (MDB bindings) and Listener ports (WebSphere MQ or MQJMS bindings). This is discussed in the following section.

Tune message-driven bean ActivationSpec properties

For each JMS export component, there is an MDB and its corresponding ActivationSpec in the Java Naming and Directory Interface (JNDI name is `module name/export component name_AS`). The default value for maxConcurrency of the JMS export MDB is 10, meaning up to 10 business objects from the JMS queue can be delivered to the MDB threads concurrently. Change it to 100 if a concurrency of 100 is wanted.

The IBM Tivoli® Performance Viewer can be used to monitor the maxConcurrency parameter. For each message being processed by an MDB, there is a message on the queue marked as being locked inside a transaction (which is removed after the onMessage completes). These messages are classed as unavailable. The PMI metric UnavailableMessageCount gives you the number of unavailable messages on each queue point. Check this value by selecting the resource name and then clicking **SIB Service → SIB Messaging Engines**. Select the bus name and click **Destinations → Queues.**

If any queue has maxConcurrency or more unavailable messages, this condition implies that the number of messages on the queue is running above the concurrency maximum of the MDB. If this situation occurs, increase the maxConcurrency setting for that MDB.

The maximum batch size in the activation specification also has an impact on performance. The default value is 1 (one). The maximum batch size value determines how many messages are taken from the messaging layer and delivered to the application layer in a single step. This batch size value does not mean that this work is done within a single transaction, and thus this setting does

not influence transactional scope. Increase this value (for example, to 8) for activation specs that are associated with SCA modules and long-running business processes to improve performance and scalability, especially for large multi-core systems.

Configure thread pool sizes 11.8

The sizes of thread pools have a direct impact on the ability of a server to run applications concurrently. For maximum concurrency, you must set the thread pool sizes to optimal values. Increasing the `maxConcurrency` parameter or `Maximum sessions` parameter only enables the concurrent delivery of business objects from the JMS or WebSphere MQ queues. For a Business Process Manager server to process multiple requests concurrently, you must increase the corresponding thread pool sizes to allow higher concurrent execution of these MDB threads.

MDB work is dispatched to threads allocated from the default thread pool. All MDBs in the application server share this thread pool unless a different thread pool is specified. This condition means that the default thread pool size needs to be larger, probably significantly larger, than the maxConcurrency parameter of any individual MDB.

Threads in the WebContainer thread pool are used for handling incoming HTTP and web services requests. This thread pool is shared by all applications deployed on the server, and you must tune it, likely to a higher value than the default.

Object request broker (ORB) thread pool threads are employed for running ORB requests (for example, remote EJB calls). The thread pool size needs to be large enough to handle requests coming through the interface, such as certain human task manager APIs.

Configure dedicated thread pools for message-driven beans

The default thread pool is shared by many WebSphere Application Server tasks. It is sometimes preferable to separate the execution of JMS MDBs to a dedicated thread pool. Complete the following steps to change the thread pool used for JMS MDB threads:

1. Create a thread pool (for example, MDBThreadPool) on the server by clicking **Servers** → **Server Types** → **WebSphere application servers** → **server** → **Thread pools**. Then click **New**.

2. Open the service integration bus (SIB) JMS Resource Adapter administrative console with server scope by clicking **Resources** → **Resource Adapters** → **Resource adapters**. If the adapter is not visible, go to **Preferences**, and select the **Show built-in resources** check box.

3. Change the thread pool alias from **Default** to **MDBThreadPool**.

4. Repeat steps 2 on page 88 and 3 for SIB JMS resource adapters at the node and cell scope.

5. Restart the server so that the changes become effective.

SCA Module MDBs for asynchronous SCA calls use a separate resource adapter, the Platform Messaging Component SPI Resource Adapter. Follow the same steps to change the thread pool to a different one if you want.

Even with a dedicated thread pool, all MDBs that are associated with the resource adapter still share a thread pool. However, they do not have to compete with other WebSphere Application Server tasks that also use the default thread pool.

Configure JMS and JMS service queue connection factories

Multiple concurrently running threads might cause bottlenecks on resources such as JMS and database connection pools if such resources are not tuned properly. The `Maximum Connections pool size` parameter specifies the maximum number of physical connections that can be created in this pool. These physical connections interface with back-end resources (for example, a DB2 database). After the thread pool limit is reached, the requester cannot create new physical connections and must wait until a physical connection that is in use is returned to the pool, or a `ConnectionWaitTimeout` exception is issued.

For example, if you set the Maximum Connections value to 5, and there are five physical connections in use, the pool manager waits for the amount of time that is specified in `Connection Timeout` for a physical connection to become free. The threads waiting for connections to the underlying resource are blocked until the connections are freed and allocated to them by the pool manager. If no connection is freed in the specified interval, a ConnectionWaitTimeout exception is issued.

If you set Maximum Connections property to 0 (zero), the connection pool is allowed to grow infinitely. This setting has the side effect of causing the `Connection Timeout` value to be ignored.

The general guideline for tuning connection factories is that their maximum connection pool size must match the number of concurrent threads multiplied by the number of simultaneous connections per thread.

For each JMS, WebSphere MQ, or MQJMS Import, a connection factory exists that was created during application deployment. Make the Maximum Connections property of the connection pool, associated with the JMS connection factory, large enough to provide connections for all threads concurrently running in the import component. For example, if 100 threads are

expected to run in a given module, set the `Maximum Connections` property to 100. The default is 10.

From the connection factory configuration panel, click **Additional Properties** → **Connection pool properties**. Set the Maximum Connections property to the maximum size of the connection pool.

4.6.4 Clustered topology tuning

One reason for deploying a clustered topology is to add more resources to system components that are bottlenecked as a result of increasing load. Ideally, you can scale up a topology incrementally to match the required load. The Business Process Manager Network Deployment (ND) infrastructure provides this capability. However, effective scaling still requires standard performance monitoring and bottleneck analysis techniques.

The following list details several considerations and tuning guidelines for configuring a clustered topology. In the description, the assumption is that additional cluster members also imply additional server hardware.

► If deploying more than one cluster member (JVM) on a single physical system, monitoring the resource use (processor cores, disk, network, and other components) of the system as a whole is important. Also monitor the resources that are used by each cluster member. With monitoring, you can detect a system bottleneck because of a particular cluster member.

► If all App Target members of a cluster are bottlenecked, you can scale by adding one or more App Target members to the cluster, backed by appropriate physical hardware.

► If a single server or cluster member is the bottleneck, consider additional factors:

  – A messaging engine in a cluster with a "One of N" policy (to preserve event ordering) might become the bottleneck. The following scaling options are available to correct this issue:

    • Host the active cluster member on a more powerful hardware server or remove extraneous load from the existing server.

    • If the messaging engine cluster services multiple buses, and messaging traffic is spread across these buses, consider employing a separate messaging engine cluster per bus.

    • If a particular bus is a bottleneck, consider whether destinations on that bus can tolerate out-of-order events. In this case, the cluster policy can be changed to allow workload balancing with partitioned destinations.

Partitioning a bus also includes considerations for balancing work across the messaging engine cluster members.

– A database server might become the bottleneck. Consider the following approaches to correct this issue:

- If the database server is hosting multiple databases that are active (for example, BPMDB, CMNDB, PDWDB, BPEDB), consider hosting each database on a separate server.

- If a single database is driving load, consider a more powerful database server.

- See Chapter 5, "Database configuration, tuning, and best practices" on page 111 for a detailed discussion about database monitoring and tuning.

4.6.5 Web services tuning

If the target of the web services import binding is hosted locally in the same application server, you can further improve the performance by taking advantage of the optimized communication path that is provided by the web container. Normally, requests from the web services clients are sent through the network connection between the client and the service provider. For local web services calls, however, WebSphere Application Server offers a direct communication channel, bypassing the network layer completely.

Complete the following steps to enable this optimization. Use the administrative console to change these values:

1. Click **Application servers** and the then the name of the server that you want. Click **Container Settings** → **Web Container Settings** → **Web container** → **Additional Properties** → **Custom Properties**. Set the web container custom property enableInProcessConnections to `true`.

   Do not use wildcard characters (*) for the host name of the web container port. Replace it with the host name or IP address. The property can be accessed by clicking the following path:

   Application servers → **messaging engine** → **Container Settings** → **Web Container Settings** → **Web container** → **Additional Properties** → **Web container transport chains** → **WCInboundDefault** → **TCP inbound channel (TCP_2)** → **Related Items** → **Ports** → **WC_defaulthost** → **Host**.

2. Use localhost instead of host name in the web services client binding. Using the actual host name (even if it is aliased to localhost), disables this optimization. To access the host name, use the following steps:

   a. Click **Enterprise Applications** and select the application name.

b. Click **Manage Modules** and select the application EJB JAR file.

c. Click **Web services client bindings** → **Preferred port mappings** and select the binding name.

Use the localhost (for example, `localhost:9080`) in the URL.

3. Make sure that there is not an entry for the server host name and IP address hosts file of your server for name resolution. An entry in the hosts file inhibits this optimization by adding name resolution processor usage.

4.6.6 Tune Human Workflow for Business Space

The following optimizations are applicable when you use the Advanced Human Workflow Template with Business Space clients.

Optimize performance when not using federation mode

If you are familiar with the performance characteristics of Human Workflow widgets with WebSphere Process Server V7.0, you might experience an initial performance degradation when using Human Workflow widgets such as Tasks, Processes, and Task Definitions in Business Process Manager V8.5. This degradation is apparent only if you use Business Space on a new (not a migrated) installation of Business Process Manager V8.5. These performance considerations apply if you use the Human Task Management widgets with only Business Process Definition (BPD) processes and human services on a single application server, or on a cluster.

This performance degradation is caused by the addition of a federation service layer and the higher number of items to be retrieved. Addition of the federated service layer is a usability enhancement initially delivered in Business Process Manager V7.5 that enables the display of all tasks in a single unified task list.

To check the Business Space target services configuration in the administration console, complete the following steps:

1. Open the IBM Business Process Manager Integrated Solutions Console.

2. Click the **Servers** tab in the navigation bar.

3. Click **WebSphere**.

4. Click **Application Servers** server type and choose the application server on which you want to run Business Space.

5. In the Business Integration section on the Configuration tab, click the **Business Space Configuration** entry.

6. On the Business Space Configuration page, in the Additional Properties section, click the **REST service endpoint registration** entry.

7. On the REST service endpoint registration page, check the Service Endpoint Target values for the Process services and Task services REST endpoint types. The default initial setting is `Federated REST services.`

To improve performance by using the Business Process Choreographer REST End Point instead of the Federated REST Endpoint, complete the following steps:

1. Change the Business Space target services configuration in the administration console.

2. Change the Process services to the Business Process Choreographer REST services endpoint of the application server that is running your processes and human tasks.

3. Change the Tasks services to the Business Process Choreographer REST services endpoint of the application server that is running your processes and human tasks.

4. Click **OK** to save the configuration changes.

5. Log out of Business Space and then log in again.

4.6.7 Set AIX threading parameters

The IBM JVM threading and synchronization components are based on threads implementation that is compliant with AIX POSIX. The following environment variable settings improve Java performance in many situations. The variables control the mapping of Java threads to AIX native threads, turn off mapping information, and allow for spinning on mutually exclusive (mutex) locks:

► `export AIXTHREAD_COND_DEBUG=OFF`
► `export AIXTHREAD_MUTEX_DEBUG=OFF`
► `export AIXTHREAD_RWLOCK_DEBUG=OFF`
► `export AIXTHREAD_SCOPE=S`
► `export SPINLOOPTIME=2000`

4.7 Tune for Business Process Execution Language business processes

This section provides advanced tuning tips for Business Process Execution Language (BPEL) business processes.

4.7.1 Tune Work Manager-based navigation for business processes

Work Manager-based navigation is the default navigation mode for BPEL business processes (rather JMS-based navigation). Work Manager-based navigation provides two methods to optimize performance, keeping the quality of service of (JMS-based) process navigation consistent with persistent messaging:

► Work Manager-based navigation

WorkManager is a thread pool of Java Platform Enterprise Edition threads. WorkManager process navigation takes advantage of an underlying capability of WebSphere Application Server to start the processing of ready-to-navigate business flow activities without using messaging as provided by JMS providers.

► InterTransactionCache

This cache is a part of the Work Manager-based navigation mode that holds process instance state information in memory, reducing the need to retrieve information from the BPE database.

Several parameters control usage of these two optimizations. In the administrative console, to find the first set of these parameters, use the following steps:

1. Click **Application Servers** and select the server name that you want.

2. Click **Business Integration** → **Business Process Choreographer** → **Business Flow Manager** → **Business Process Navigation Performance**.

Enabling this capability at the cluster level overrides the settings for a specific server. So, in a clustered environment, enabling this capability at the cluster lever is the easiest approach.

Key parameters are as follows:

► Enable advanced performance optimization

Select this property to enable both the Work Manager-based navigation and InterTransactionCache optimizations.

► Work-Manager-Based Navigation Message Pool Size

This property specifies the size of the cache used for navigation messages that cannot be processed immediately, provided Work Manager-based navigation is enabled. The cache defaults to the message size (computed by 10 times the thread pool size of the BPENavigationWorkManager). If this cache reaches its limit, JMS-based navigation is used for new messages. For optimal performance, ensure that this Message Pool size is set to a sufficiently high value.

► InterTransaction Cache Size

This property specifies the size of the cache used to store process state information that has also been written to the BPE database. Set this value to twice the number of parallel running process instances. The default value for this property is the thread pool size of the BPENavigationWorkManager.

In addition, you can customize the number of threads for the Work Manager that use these settings by clicking **Resources** → **Asynchronous Beans** → **Work Managers** → **BPENavigationWorkManager**.

Increase the minimum number of threads from its default value of 5, and increase the maximum number of threads from its default value of 12, using the methodology outlined in 4.6.3, "Tune for maximum concurrency" on page 86. If the thread pool size is modified, also modify the work request queue size and set it to be twice the maximum number of threads.

4.7.2 Tune the business process container for Java Message Service navigation

If JMS-based navigation is configured, you must optimize the following resources for efficient navigation of business processes:

► Activation specification BPEInternalActivationSpec

The Maximum Concurrent Endpoints parameter specifies the parallelism that is used for process navigation across all process instances. Increase the value of this parameter to increase the number of business processes run concurrently. Find this resource in the administrative console by clicking **Resources** → **Activation Specifications** → **BPEInternalActivationSpec**.

► JMS connection factory BPECFC

Set the connection pool size to the number of threads in the BPEInternalActivationSpec plus 10%. Find this resource in the administrative console by clicking **Resources** → **JMS** → **Connection factories** → **BPECFC** → **Connection pool properties**. This connection factory is also

used when Work Manager-based navigation is in use, but only for error situations or if the server is highly overloaded.

4.7.3 Tune task list and process list queries

The programmer creates task list and process list queries in Business Process Manager applications by using the standard query APIs, that is, query() and queryAll(), and related REST and web services interfaces. Task list and process list queries are also created by the query table APIs queryEntities() and queryRows(). All task list and process list queries result in SQL queries against the Business Process Choreographer database. These SQL queries might need special tuning to provide optimal response times, as follows:

▶ Up-to-date database statistics are key for good SQL query response times.

▶ Databases offer tools to tune SQL queries. In most cases, additional indexes improve query performance with some potential impact on process navigation performance. For DB2, the DB2 design advisor can be used to guide in choosing indexes. For more information, refer to Chapter 5, "Database configuration, tuning, and best practices" on page 111.

4.7.4 Tune Business Process Choreographer API calls

Business Process Choreographer API calls are triggered by requests external to the Business Process Manager server run time. Examples include remote EJB requests, web service requests, web requests over HTTP, requests that come through the SCA layer, or JMS requests. The connection pools associated with each of these communication mechanisms might need tuning.

Consider the following hints when tuning the connection pools:

▶ API calls for task list and process list queries might take more time to respond, depending on the tuning of the database and the amount of data in the database.

▶ Ensure that concurrency (parallelism) is sufficiently high to handle the load and to use the processor. However, increasing the parallelism of API call execution beyond what is necessary can negatively influence response times. Also, increased parallelism can put excessive load on the Business Process Choreographer database. When tuning the parallelism of API calls, measure response times before and after tuning, and adjust the parallelism if necessary.

4.7.5 Tune intermediate components for concurrency

If the input business object is handled by a single thread from end to end, the tuning for the edge components is normally adequate. In many situations, however, multiple thread switches exist during the end-to-end execution path. Tuning the system to ensure adequate concurrency for each asynchronous segment of the execution path is important.

Asynchronous invocations of an SCA component use an MDB to listen for incoming events that arrive in the associated input queue. Each SCA module defines an MDB and its corresponding activation specification (JNDI name is `sca/`*module name*`/ActivationSpec`). The SCA module MDB is shared by all asynchronous SCA components within the module, including SCA export components. Take this shared state into account when you configure the maxConcurrency property value of ActivationSpec. SCA module MDBs use the same default thread pool as those for JMS exports.

The asynchronicity in a long-running business process occurs at transaction boundaries (see 3.3.9, "Transactional considerations" on page 51 for more details about settings that affect transaction boundaries). BPE defines an internal MDB and its ActivationSpec as BPEInternalActivationSpec. The maxConcurrency parameter must be tuned by following the same guideline as for SCA module and JMS export MDBs (as described in 4.6.3, "Tune for maximum concurrency" on page 86).

The only issue is that only one BPEInternalActivationSpec exists for a single Business Process Manager server.

4.8 Mediation flow component tuning

Additional configuration options are relevant to tuning Mediation Flow Components. These are described in this section.

See 8.2, "Mediation Flow Component settings" on page 185 for a suggested set of initial values to use.

4.8.1 Tune the database if using persistent messaging

If you use persistent messaging, the configuration of your database is important. Use a remote DB2 instance with a fast disk array as the database server. You might benefit from tuning the connection pooling and statement cache of the data source.

For more information about tuning DB2, see Chapter 5, "Database configuration, tuning, and best practices" on page 111.

4.8.2 Configure Service Registry and Repository cache timeout

WebSphere Service Registry and Repository (WSRR) is used by the Mediation Flow Component for endpoint lookup. When accessing the WSRR (for example, using the endpoint lookup mediation primitive), results from the registry are cached in the Mediation Flow Component. You can configure the lifetime of the cached entries from the administrative console by clicking **Service Integration** → **WSRR Definitions** and entering the WSRR definition name. Then, click **Timeout of Cache** and set a value for the cached registry results.

Validate that the timeout is a sufficiently large value. The default timeout is 300 seconds, which is reasonable from a performance perspective. A value that is too low results in frequent lookups to the WSRR, which can be expensive (especially if retrieving a list of results) and includes the associated network latency if the registry is on a remote machine.

4.9 Business Monitor tuning

This section provides advanced tuning suggestions for Business Monitor.

4.9.1 Configure Java heap sizes

The default maximum heap size in most implementations of Java is too small for many of the servers in this configuration. The Business Monitor Launchpad installs the Business Monitor and its prerequisite servers with larger heap sizes, but you might check that these sizes are appropriate for your hardware and workload.

For IBM Business Monitor golden topologies, configuring Java heap sizes is an important task. In general, the Support cluster must be adjusted to use a larger heap since this cluster performs more of the processing than other clusters.

4.9.2 Configuring Common Event Infrastructure

By default, when an event arrives at Common Event Infrastructure (CEI), it is delivered to the registered consumer (in this case, a particular monitor model) and into an additional default queue.

Performance improves by avoiding this double-store by removing the All Events event group:

1. In the administrative console, select **Service Integration** → **Common Event Infrastructure** → **Event Service** → **Event Services** → **Default Common Event Infrastructure event server** → **Event Groups**.

2. Remove the event group.

Beyond its persistent delivery of events to registered consumers, CEI offers the ability to explicitly store events in a database. Database event storage requires significant processor usage, so avoid storing the events in the database if this additional functionality is not needed. You can also configure the CEI data store:

1. In the administrative console, select **Service Integration** → **Common Event Infrastructure** → **Event Service** → **Event Services** → **Default Common Event Infrastructure event server**.

2. Clear the **Enable Data Store** check box.

4.9.3 Configure message consumption batch size

Processing events in large batches is much more efficient than doing it one at a time. Up to some limit, the larger the batch size, the higher the throughput rate. But there is a trade-off: processing and persisting events to the Monitor database is done as a transaction. Although a larger batch size yields better throughput, it costs more if you must roll back. If you experience frequent rollbacks, consider reducing the batch size. You can reduce cache size in the administrative console under the server scope:

1. Click **Applications** → **Monitor Models** and select the version of the batch.

2. Click **Runtime Configuration** → **Tuning** → **Message Consumption Batch size** and set the batch size that you want. The default value is 100.

4.9.4 Key performance indicator caching

When we look at key performance indicators we want to point out the following two specifics:

► Model-specific caching
► KPI-specific caching

Model-specific caching

The cost of calculating aggregate key performance indicator (KPI) values increases as completed process instances accumulate in the database. A KPI cache is available to reduce the resource usage of these calculations at the cost

of some staleness in the results. The refresh interval is configurable in the administrative console:

1. Select **Applications** → **Monitor Models** and select the version.

2. Select **Runtime Configuration** → **KPI** → **KPI Cache Refresh Interval**.

A value of zero (the default) disables the cache.

KPI-specific caching

It is also possible to configure each KPI to have a KPI-specific cache value, which provides greater flexibility to refine how often the cache is refreshed. Determining the refresh rate is often a business decision. To set a specific refresh rate for a given KPI, perform the following steps:

1. Sign in to the Business Space and navigate to the page where the KPI Manager is located, then pick a KPI.

2. Go to the Other tab.

3. Select the **Override the default KPI cache interval (in minutes)** check box and provide a reasonable value.

4. Select **Ok** to save your changes.

4.9.5 Enable the Data Movement Service

By default, the same tables are used for event processing and for dashboard reporting. You can enable an optional scheduled service, called the *Data Movement Service (DMS)* to switch dashboard reporting against a separate set of tables. DMS also periodically copies the data from the event processing tables for the server to the dashboard reporting tables.

This processing and reporting mode optimizes the following components:

► Indexes for each table
► Event processing tables for the server for quick insert/update/delete
► Dashboard reporting tables for common dashboard-related queries

We suggest that you enable DMS in any production scenario.

4.9.6 Dynamic KPI performance

Dynamic KPIs are created as the name suggests, dynamically on the Business Space dashboard. You need to have your DBA create an index for this KPI in order to have reasonable response times when calculating the KPI value.

4.9.7 Cognos tuning

It is important to tune the IBM Cognos Business Intelligence Server that is used by IBM Business Monitor. For information about how to tune Cognos, see IBM Cognos BI Tuning in the IBM Knowledge Center at the following link:

http://www.ibm.com/support/knowledgecenter/api/content/SSEP7J_10.2.1/com.ibm.swg.ba.cognos.crn_arch.10.2.1.doc/c_arch_cognos8tuning.html#arch_Cognos8Tuning?locale=en

4.9.8 DB2 tuning

For IBM Business Monitor, it is recommended that DB2 customers set the following two DB2 tuning parameters:

```
db2set DB2_SKIPINSERTED=ON
db2set DB2_SKIPDELETED=on
```

For other tips regarding DB2 tuning, see the appropriate section in Chapter 5, "Database configuration, tuning, and best practices" on page 111.

4.10 Enable browser caching

Business Process Manager clients rely heavily on Ajax and JavaScript code; their performance depends on the browser's efficiency in processing this code. To improve Business Process Manager client-response times, you might need to adjust both HTTP server and client browser settings. This section covers ensuring that browser caching is enabled. HTTP server caching is addressed in 4.11, "Tune the HTTP server" on page 103.

4.10.1 Ensure browser cache is enabled in Internet Explorer

By default, Business Process Manager allows the browser to cache static information such as style sheets and JavaScript files for 24 hours. However, if the browser is configured in a way that does not cache data, the same resources are loaded repeatedly, resulting in long page-load times.

To enable caching, use the following steps:

1. Click **Tools** → **Internet Options** → **General** → **Browsing History**.
2. Set the cache size, indicated by **Disk space to use**, to at least 50 MB.

Ensure browser cache is not cleared on logout

Modern browsers have options to clear the cache on exit; make sure that this setting is not enabled.

Use the following procedure to disable cache clearing:

1. Click **Tools** → **Internet Options**.
2. Under the **General** tab, make sure the **Delete browser history on exit** check box is cleared.
3. Click the **Advanced** tab and make sure the **Empty Temporary Internet Files folder when browser is closed** check box is cleared. Click **OK** to save the settings.
4. Restart your browser to make sure that the changes have taken effect.

4.10.2 Ensure browser cache is enabled in Firefox

By default, Business Process Manager allows the browser to cache static information such as style sheets and JavaScript files for 24 hours. However, if the browser is configured in a way that does not cache data, the same resources are loaded repeatedly, resulting in long page-load times.

To enable caching, use the following steps:

1. Click **Tools** → **Options** → **Advanced** → **Network** → **Offline Storage**.
2. Set the offline storage to at least 50 MB.

Use the following procedure to enable cache clearing:

1. Click **Tools** → **Options**.
2. Click the **Privacy** tab and make sure that it reads `Firefox will: Remember History`. Click **OK**. This setting enables caching.

   If you click **Never remember history**, caching is disabled and you must change the setting.

   If you click **Use custom settings for history**, you have additional options that still allow caching:

   – If you select **Automatically start Firefox in a private browsing session**, caching is enabled. However, after the browser is closed, everything is erased (not only cache, but also browsing history) as though you were never using the browser.

   – If private browsing is not selected, make sure the **Clear history when Firefox closes** check box is cleared.

3. Restart the browser to apply the changes.

4.11 Tune the HTTP server

The Process Server and Process Center in production scenarios are often deployed by using a topology that includes an HTTP server or an HTTP server plug-in. The Process Portal and Process Designer send multiple requests to their respective server, which are then handled by this HTTP server. To minimize the number of requests and the size of the response data to obtain good user response times, tune your HTTP server to efficiently handle the expected load.

In particular, ensure that caching is effectively enabled in the HTTP server. Much of the content that is requested by the Process Portal and Process Designer is static (for example images and JavaScript files), and can be cached in the browser. Of course, ensure that browser caching is enabled first, but also tune the HTTP server to support caching at that level. Specifically, in the `httpd.conf` file, use the following caching and compression settings:

► nExpiresActive on
► ExpiresByType image/gif A86400
► ExpiresByType image/jpeg A86400
► ExpiresByType image/bmp A86400
► ExpiresByType image/png A86400
► ExpiresByType application/x-javascript A86400
► ExpiresByType text/css A86400

Tip: 86400 seconds = 1 day.

For more information about tuning the IBM HTTP Server, which is included as part of Business Process Manager V8.5, see the IBM HTTP Server Performance Tuning page:

http://publib.boulder.ibm.com/httpserv/ihsdiag/ihs_performance.html

Further, if the Process Portal or Process Designer clients are not located physically close to the Process Server, locating HTTP Servers near these clients can improve response time by reducing the network latency for retrieving the static content described above.

4.12 Advanced Java heap tuning

Because the Business Process Manager product set is written in Java, the performance of the JVM has a significant impact on the performance delivered by these products. JVMs externalize multiple tuning parameters that might be used to improve both authoring and runtime performance. The most important of these

parameters are related to garbage collection and setting the Java heap size. This section explains these topics in detail.

The products covered in this book use IBM JVMs on most platforms (for example, AIX, Linux, and Windows), and the HotSpot JVMs on selected other systems, such as Solaris. Business Process Manager 8.5 uses Java 6.

The IBM Java 6 diagnostics guide is at the following website:

http://publib.boulder.ibm.com/infocenter/javasdk/v6r0/index.jsp

The guide describes many more tuning parameters than those described in this book, but most are for specific situations and are not of general use. For a more detailed description of IBM Java 6 garbage collection algorithms, see the section on memory management.

4.12.1 Monitoring garbage collection

To set the heap correctly, determine how the heap is being used by collecting a verbosegc trace. A verbose garbage collection (verbosegc) trace prints garbage collection actions and statistics to stderr in IBM JVMs and stdout in Oracle HotSpot JVMs. The verbosegc trace is activated by using the Java runtime option `-verbose:gc`. Output from verbosegc differs for the HotSpot and IBM JVMs, as shown by Example 4-4 and Example 4-5 on page 105.

Example 4-4   IBM JVM verbosegc trace output

```
<af type="tenured" id="12" timestamp="Fri Jan 18 15:46:15 2008" intervalms="86.539">
  <minimum requested_bytes="3498704" />
  <time exclusiveaccessms="0.103" />
  <tenured freebytes="80200400" totalbytes="268435456" percent="29" >
    <soa freebytes="76787560" totalbytes="255013888" percent="30" />
    <loa freebytes="3412840" totalbytes="13421568" percent="25" />
  </tenured>
  <gc type="global" id="12" totalid="12" intervalms="87.124">
    <refs_cleared soft="2" threshold="32" weak="0" phantom="0" />
    <finalization objectsqueued="0" />
    <timesms mark="242.029" sweep="14.348" compact="0.000" total="256.598" />
    <tenured freebytes="95436688" totalbytes="268435456" percent="35" >
      <soa freebytes="87135192" totalbytes="252329472" percent="34" />
      <loa freebytes="8301496" totalbytes="16105984" percent="51" />
    </tenured>
  </gc>
  <tenured freebytes="91937984" totalbytes="268435456" percent="34" >
    <soa freebytes="87135192" totalbytes="252329472" percent="34" />
    <loa freebytes="4802792" totalbytes="16105984" percent="29" />
```

```
    </tenured>
    <time totalms="263.195" />
</af>
```

Example 4-5   Solaris HotSpot JVM verbosegc trace output (young and old)

```
[GC 325816K → 83372K(776768K), 0.2454258 secs]
[Full GC 267628K → 83769K <- live data (776768K), 1.8479984 secs]
```

Oracle HotSpot JVM verbosegc output can be more detailed by setting additional options:

▶   -XX:+PrintGCDetails
▶   -XX:+PrintGCTimeStamps

Parsing the verbosegc output by using a text editor can be tedious. Visualization tools that can be used for more effective Java heap analysis are available on the web. The IBM Pattern Modeling and Analysis Tool for Java Garbage Collector (PMAT) is one such tool. It is available for download at IBM alphaWorks® at the following website:

http://www.alphaworks.ibm.com/tech/pmat

PMAT supports the verbosegc output formats of JVMs offered by major JVM vendors such as IBM, Oracle, and HP.

4.12.2 Set the heap size for most configurations

This section contains guidelines for determining the appropriate Java heap size for most configurations. If your configuration requires that more than one JVM run concurrently on the same system, see 4.12.3, "Set the heap size when running multiple JVMs on one system" on page 107. For example, you might need more than one JVM if you run both a Business Process Manager server and Integration Designer on the same system. If your objective is designed to support large business objects, read 4.6.2, "Tune for large objects" on page 86.

When the heap size is too low, OutOfMemory errors occur. For most production applications, the IBM JVM Java heap size defaults are too small, so increase them. When the heap size is too low, OutOfMemory errors occur. In general, the HotSpot JVM default heap and repeated nursery size are also too small so increase them also.

To set optimal heap sizes, start with reasonable initial values, and then conduct performance and load tests to determine if further tuning is required. Also, use IT Monitoring tools while in production to ensure that the heap size is optimal for performance.

For many installations, starting with values of 768 MB for the minimum heap size and 3072 MB for the maximum heap size is reasonable. Note, however, that the maximum heap size must be set such that the heap is not paged to disk. The heap *must always stay* in physical memory, or response times will be significantly affected.

The JVM will work to keep the GC time within reasonable limits by growing and shrinking the heap as needed. The output from verbosegc is used to monitor garbage collection (GC) activity.

If Generational Concurrent GC is used (`-Xgcpolicy:gencon`), you can also set the new area size to specific values. By default, the new size is a quarter of the total heap size or 64 MB, whichever is smaller. For better performance, set the nursery size to a range of 1/4 to 1/2 of the heap size. You can set new area sizes by using the following JVM options:

► `-Xmn<size>`
► `-Xmns<initialSize>`
► `-Xmnx<maxSize>`

You can use a similar process to set the size of HotSpot heaps. In addition to setting the minimum and maximum heap size, also increase the nursery size to a range of 1/4 to 1/2 of the heap size. Never increase the nursery to more than half the full heap.

You can set the nursery size by using the MaxNewSize and NewSize parameters:

► `-XX:MaxNewSize=128m`
► `-XX:NewSize=128m`

After the heap sizes are set, use verbosegc traces to monitor GC activity. After analyzing the output, modify the heap settings accordingly. For example, if the percentage of time in GC is high and the heap has grown to its maximum size, you might improve throughput by increasing the maximum heap size. As a general guideline, greater than 10% of the total time spent in GC is considered high.

Increasing the maximum size of the Java heap might not always solve this type of problem because the problem might be memory overuse. Conversely, if response times are too long because of GC pause times, decrease the heap size. If both problems are observed, an analysis of the application heap usage is required.

4.12.3 Set the heap size when running multiple JVMs on one system

Each running Java program has a heap associated with it. If you have a configuration where more than one Java program is running on a single physical system, setting the heap sizes appropriately is of particular importance.

An example of one such configuration is when the Integration Designer is on the same physical system as a Business Process Manager server. Each of these applications is a separate Java program that has its own Java heap. If the sum of all of the virtual memory usage (including both Java heaps and all other virtual memory allocations) exceeds the size of addressable physical memory, the Java heaps are subject to paging. Such paging causes total system performance to degrade significantly. To minimize the possibility of total system degradation, use the following guidelines:

► First, collect a verbosegc trace for each running JVM.

► Based on the verbosegc trace output, set the initial heap size to a relatively low value. For example, assume that the verbosegc trace output shows that the heap size grows quickly to 256 MB, and then grows more slowly to 400 MB and stabilizes at that point. Based on this change, set the initial heap size to 256 MB (`-Xms256m`).

► Also based on the verbosegc trace output, set the maximum heap size appropriately. Be careful not to set this value too low, or out-of-memory errors occur. The maximum heap size must be large enough to allow for peak throughput. Using the same example, a maximum heap size of 768 MB might be appropriate (`-Xmx768m`). Correct sizing of maximum heap gives the Java heap room to expand beyond its current size of 400 MB, if required. The Java heap grows only if required (for example, if a period of peak activity drives a higher throughput rate), so setting the maximum heap size higher than current requirements is generally a good policy.

► Be careful not to set the heap sizes too low, or garbage collections will occur frequently, which might reduce throughput. Again, a verbosegc trace assists in determining garbage collection frequency. You must ensure that the heap sizes are large enough that garbage collections do not occur too often. At the same time, you must still ensure that the heap sizes are not cumulatively so large as to cause the heap to page to the file system. This balancing act depends on configuration.

4.12.4 Reduce or increase heap size if OutOfMemory errors occur

The `java.lang.OutOfMemory` exception is used by the JVM in various circumstances, so that finding the source of the exception is difficult. There is no conclusive mechanism for telling the difference between these potential error

sources, but a good start is to collect a trace using verbosegc. If the problem is a lack of memory in the heap, you can easily see this condition in the output. For more information about verbosegc output, see 4.12.1, "Monitoring garbage collection" on page 104. Repeated garbage collections that produce little free heap space generally occur preceding this exception. If this lack of free heap space is the problem, increase the size of the heap.

If there is enough free memory when the `java.lang.OutOfMemory` exception occurs, the next item to check is the finalizer count from the verbosegc (only the IBM JVM provides this information). If this count appears high, a subtle effect might be occurring whereby resources outside the heap are held by objects within the heap and being cleaned by finalizers. Reducing the size of the heap can alleviate this situation by increasing the frequency with which finalizers are run. In addition, examine your application to determine whether the finalizers can be avoided or minimized.

Out-of-memory errors can also occur for issues unrelated to JVM heap usage, such as running out of certain system resources. Examples of this problem include insufficient file handles or thread stack sizes that are too small.

In some cases, you can tune the configuration to avoid running out of native heap. Try reducing the stack size for threads (the **-Xss** parameter). Deeply nested methods might force a thread stack overflow if there is insufficient stack size.

Use the `-Xgc:preferredHeapBase` parameter to avoid native out-of-memory issues because of exhaustion of memory addresses below 4 GB (for example, classes and threads are allocated in this region).

You can use the `-Xgc:preferredHeapBase` option to move the Java heap outside of the lower 4 GB address space. See the JVM IBM Knowledge Center for a more detailed description of the solution:

http://ibm.co/1OCGjeT

If you are using an in-process version of the JDBC driver, it is possible to find an out-of-process driver that can have a significant effect on the native memory requirements. For example, you can use type 4 JDBC drivers (DB2 Net drivers or Oracle's Thin drivers), or you can switch IBM MQSeries® from Bindings mode to Client mode. See documentation for DB2, Oracle, and IBM MQSeries for more details.

4.13 Tuning for WebSphere InterChange Server migrated workloads

The following tuning suggestions are unique to workloads that are migrated by using the WebSphere InterChange Server migration wizard in the Integration Designer. In addition to these suggestions, see the other Business Process Manager server tuning suggestions detailed in this document:

► For JMS-based messaging used to communicate with older WebSphere Business Integration Adapters or custom adapters, use non-persistent queues when possible.

► For JMS-based messaging used to communicate with older WebSphere Business Integration Adapters or custom adapters, use WebSphere MQ-based queues, if available. By default, the adapters use the WebSphere MQ APIs to connect to the service integration bus destinations through MQ Link. MQ Link is a protocol translation layer that converts messages to and from MQ-based clients. By switching to WebSphere MQ-based queues, MQLink translation costs are eliminated and performance is improved.

► Turn off server logs for verbose workloads. Some workloads emit log entries for every transaction, causing constant disk writes and reducing overall throughput. Turning off server logs might reduce the throughput degradation for such workloads.

5

Database configuration, tuning, and best practices

IBM Business Process Manager uses a relational database not only for persisting the state of business processes and tasks, it also leverages the multi-user concurrency and transactional features of the database to guarantee integrity and consistency in a multi-user clustered BPM environment.

The nature of the process models has an immediate effect on database performance. Poorly designed process models can slow down the whole BPM system. Refer to Chapter 3, "Development best practices" on page 29. A well tuned database and database infrastructure is key to performance. In this chapter we cover the following aspects:

► Database tuning considerations
► Tuning and recommendations for BPMN workload
► Tuning and recommendations for BPEL workload
► Suggestions for Business Monitor

Each of the sections first provides a general discussion, and then discusses more details for a particular database product.

111

5.1 Database tuning considerations

This chapter describes common tuning tasks and recommendations for the BPM database. The general section describes the tuning tasks from a conceptual point of view. It describes important aspects of the BPM database that typically require tuning. More database-specific topics will be added in the subsequent chapters, as shown in this overview:

► General database tuning
► DB2 specific database tuning
► Oracle specific database tuning
► SQL Server specific database tuning

5.1.1 General database tuning

This section describes common BPM database tuning tasks and configuration parameters, common for all database products or specific workloads.

Follow common database best practices

For every database system, there is set of best practices and recommendations available as books, product documentation, or on the web. Examples for best practices are monitoring the database periodically, keeping the optimizer statistics up to date, keeping track of changes, and so on. These best practices are also valid in the context of BPM. In the remainder of this chapter, the authors emphasize particular aspects that are important for a healthy and well performing BPM database.

Table spaces

When you set up your production database, use table spaces for the Business Process Choreographer component. The tables of the ProcessServer and Performance Data Warehouse components do not specify table spaces. You can add table space information to the DDL scripts manually, or you can use database tools to reassign the tables to different table spaces. Using table spaces allows for flexibly balancing the I/O load between different disk drives or other attached storage and to tune caching for large objects (LOBs).

File system I/O, caching, and special considerations for large objects

Because a database attempts to keep frequently accessed data in memory, in many cases it is desirable to bypass the file system cache when accessing

database files to avoid holding more than one copy of the data in memory. This can be accomplished by using direct I/O or concurrent I/O.

Using concurrent I/O also bypasses per file locking by shifting responsibility for concurrency control to the database, and is a preferable choice for performance on those operating systems that support it.

An important exception to this guideline is for LOBs that are not typically cached in database memory (types such as binary large object (BLOB), character large object (CLOB), and so on). Business Process Manager uses LOB columns for a number of tables to store data. If these LOBs are not cached in database memory or the file system cache, they will be accessed directly from disk, which can result in a severe performance penalty.

Database systems offer the possibility to use separate table spaces for table data and LOBs. Use table spaces with file system caching for LOBs and table spaces without file system caching for table data.

Care must be taken to arrange the database storage with these considerations in mind. The specific database product sections describe this in further detail.

Transaction log considerations

In this section we examine the following details:

► Place database log files on a fast disk subsystem
► Place database log files on a separate device from the table space containers

Place database log files on a fast disk subsystem

Databases are designed for high availability, transactional processing, and recoverability. For performance reasons, changes made to table data might not be written immediately to disk. These changes can be recovered if they are written to the database log. Updates are made to database log files when the log buffer fills, at transaction-commit time, and for some implementations after a maximum interval of time.

As a result, database log files might be heavily used. More important, the log-writes hold commit operations pending, meaning that the application is synchronously waiting for the write to complete. Therefore, the performance of write access to the database log files is critical to overall system performance. For this reason, we suggest that database log files be placed on a fast disk subsystem with write-back cache.

Place database log files on a separate device from the table space containers

A basic strategy for all database storage configurations is to place the database logs on dedicated physical disks, ideally on a dedicated disk adapter. This placement reduces disk access contention between I/O to the table space containers and I/O to the database logs and preserves the mostly sequential access pattern of the log stream. Such separation also improves recoverability when log archival is employed.

Memory

Accessing data in memory is much faster than reading it from disk. Because 64-bit hardware is readily available and memory prices continue to fall, the sensible approach is to provision enough memory to avoid most disk reads in steady state for many performance-critical workloads.

Be careful to avoid virtual memory paging in the database machine. The database manages its memory with the assumption that it is never paged and does not cooperate well if the operating system swaps some of its pages to disk.

Care needs to be taken when multiple DBs are located on the same server. To avoid paging, the combined maximum memory settings should not exceed the physical memory in the machine. Memory usage by OS and file system caching should also be considered when calculating what memory settings to use.

Statistics

Databases use a wide variety of available choices when determining the best approach to access data. Statistics, which describe the shape of the data, are used to guide the selection of a low-cost data access strategy. Statistics are maintained in tables and indexes. Examples of statistics include the number of rows in a table and the number of distinct values in a certain column.

Gathering statistics can be expensive, but fortunately, for many workloads, a set of representative statistics allows for good performance over a large span of time. Refreshing statistics periodically if the data population shifts dramatically might be necessary:

► If database utilization is running at high throughput rates, consider disabling some or all database auto-maintenance tasks to avoid impacting peak throughput. However, if you disable these capabilities, be sure to regularly update database statistics.

► For IBM DB2, exclude the table SIBOWNER from automatic runstats execution as described in the following technote:

http://www.ibm.com/support/docview.wss?uid=swg21452323

Tuning queries

In this section we examine the following details:

► Monitor top SQL statements
► Add indexes as required
► Optimization profiles

Monitor top SQL statements

Use the database vendor's tools to discover expensive SQL statements, for example the `SYSIBMADMIN.TOP_DYN_SQL` view of DB2, or the automated workload repository (AWR) report of an Oracle database. Even in a perfectly tuned database, you can find a most expensive SQL query, even if this query needs no tuning at all.

Add indexes as required

Business Process Manager products provide a reasonable set of indexes for the database tables they use. In general, creating indexes involves a tradeoff between the cost of queries and the cost of statements that insert, update, or delete data. For query-intensive workloads, providing a rich variety of indexes as required to allow rapid access to data makes sense. For update-intensive workloads, a helpful approach is to minimize the number of indexes defined, because each row modification might require changes to multiple indexes. Indexes are kept current even when they are infrequently used.

Index design therefore involves compromises. The default set of indexes might not be optimal for the database queries generated by a Business Process Manager product in a specific situation. If database processor or disk use is high or there are concerns with database response time, it might be helpful to consider changes to the indexes.

DB2 and Oracle databases provide assistance in this area by analyzing indexes in the context of a given workload. These databases offer suggestions to add, modify, or remove indexes. One caveat is that if the workload does not capture all relevant database activity, a necessary index might appear unused, leading to a suggestion that it be dropped. If the index is not present, future database activity can suffer as a result.

Optimization profiles

In rare cases, no good query execution plans can be found for particular queries, although you have followed all recommendations, and even if the optimizer statistics are up to date and appropriate indexes are in place. In these cases, you can use database vendors' tools to tune the particular query. You need to generate a good query execution plan, and create an optimization profile, that can be applied to the database, so that the good query execution plan can be picked up later, when the BPM runtime uses the query.

Tuning with optimization profiles requires much effort and can best be done by an experienced database administrator. With a single optimization profile, you can improve only one or a few similar queries, so tuning many queries will be a time-consuming task. You need to also take into account, that over time the data of optimization profiles can get out-dated, so that the query performance degrades again. In this case, you need to remove the optimization profile and run the tool again to create a new optimization profile.

On DB2, you can apply optimization profiles. Refer to the following IBM developerWorks article:

http://www.ibm.com/developerworks/data/library/techarticle/dm-1202store dprocedure

See also the DB2 product documentation in the IBM Knowledge Center:

http://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw .admin.perf.doc/doc/c0024522.html

On Oracle, you can run the SQL Tuning Advisor to create an SQL profile. Refer to the Oracle Database SQL Tuning Guide.

Delete completed process instances

Over time, completed process instances accumulate in the database of the servers. This accumulation can alter the performance characteristics of the solution being measured. Delete or archive completed process instances to ensure that the database size is controlled. If you do not archive completed process instances and Performance Data Warehouse events, they grow unbounded over time, which impacts overall system performance. Refer to 7.3, "Purging Business Process Manager data stores" on page 173.

Data source configuration

In this section, we describe the following details:

► Maximum Connections
► Prepared statement cache size

Maximum Connections

Make the Maximum Connections property of data sources large enough to allow concurrent access to the databases from *all threads*. There are a number of data sources configured in Business Process Manager servers. Set the Maximum Connections property of each data source to match the maximum concurrency of other system resources as described in 4.6.3, "Tune for maximum concurrency" on page 86 and in 4.4.2, "Tune the Event Manager" on page 76. Also ensure on the database side, that the database can handle the number of incoming connections.

Prepared statement cache size

IBM Business Process Manager uses prepared statements extensively. In high-throughput scenarios, increase the statement cache size to profit from shorter statement preparation times and faster response times resulting from caching of prepared statements. However, for Oracle databases, each prepared statement can take a significant amount of Java heap space (140 KB per statement for some cases that we have seen in Business Process Manager solutions). As such, do the following tasks:

► Judiciously set prepared statement cache sizes to be large enough to improve performance (particularly for the Business Process Choreographer data source for Business Process Execution Language (BPEL) processes, which realize significant benefit from the prepared statement cache).

► Monitor Java heap utilization to ensure that the prepared statements are not using too much heap space or causing OutOfMemory exceptions.

5.1.2 DB2 specific database tuning

Providing a comprehensive DB2 tuning guide is beyond the scope of this book. DB2 provides a number of automatic features to assist in managing and tuning a database system and it is recommended to utilize these when possible. In addition, the authors of this book recommend several guidelines to assist in improving the performance of Business Process Manager products when used in DB2 environments. This chapter describes these guidelines and provides pointers to more detailed information.

A complete set of DB2 information (including database tuning guidelines) is available in the DB2 IBM Knowledge Center:

http://www.ibm.com/support/knowledgecenter/#!/SSEPGG_9.7.0/com.ibm.db2.luw.kc.doc/welcome.html

http://www.ibm.com/support/knowledgecenter/#!/SSEPGG_10.1.0/com.ibm.db2.luw.kc.doc/welcome.html

For general DB2 monitoring and tuning best practices, refer to the article *Tuning and Monitoring Database System Performance* on IBM developerWorks:

https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/Wc9a068d7f6a6_4434_aece_0d297ea80ab1/page/Tuning%20and%20Monitoring%20Database%20System%20Performance

Another useful reference is the "Best practices for DB2 for Linux, UNIX, and Windows and IBM InfoSphere® Warehouse" community, which can be found at:

http://www.ibm.com/developerworks/data/bestpractices/db2luw

Table spaces

By default, the tables of the Process Server database (Business Processing Modeling Notation (BPMN) workload) and the Performance Data Warehouse database are created in the user's default table space. It can be beneficial to place the tables into dedicated table spaces so you can distribute the I/O activity across multiple files, or even across multiple storage devices.

At least, move the LOB columns to a table space that supports file system caching.

With BPM 8.5.5, the table creation scripts include a storage clause for every CREATE TABLE statement that allows you to change the default table space when you create the tables. The storage clause is commented out by default. If you want to use the storage clause, you need to perform the following steps before running the script on your database:

1. In a text editor, open the schema creation script.

2. Replace all occurrences of

   `IN @TAB_TS@ INDEX IN @IDX_TS@ LONG IN @LOB_TS@`

   with

   `IN MY_TAB_TS INDEX IN MY_IDX_TS LONG IN MY_LOB_TS`

   where `MY_TAB_TS`, `MY_IDX_TS` and `MY_LOB_TS` are valid 32K table spaces.

   `MY_TAB_TS` will be used for storing table data, `MY_IDX_TS` for index data, and `MY_LOB_TS` for lob data.

3. Save the schema creation script.

If you already have created the database tables, you can use a DB2 administrative procedure to change the table spaces of a particular table, as shown in Example 5-1.

Example 5-1   Changing table space definition using a DB2 administrative procedure

```
CALL SYSPROC.ADMIN_MOVE_TABLE('MYSCHEMA', 'MYTABLE', 'MY_TAB_TS',
'MY_IDX_TS', 'MY_LOB_TS', NULL, NULL, NULL, NULL, NULL, 'MOVE'), where
MY_TAB_TS, MY_IDX_TS and MY_LOB_TS are valid 32K table spaces.
MY_TAB_TS will be used for storing table data, MY_IDX_TS for index
data, and MY_LOB_TS for lob data.
```

File system I/O, caching, and special considerations for large objects

To achieve optimal performance with DB2 databases, LOBs can be managed separately from other data using the following approach:

- ► Use the NO FILE SYSTEM CACHING setting on the table spaces that do not have LOBS to enable the use of concurrent I/O.

- ► Move the LOBs to table spaces with the FILE SYSTEM CACHING setting to enable the use of the file system cache.

A number of LOB columns will also have a SET INLINE LENGTH value already specified for a default install as this can significantly improve performance by allowing the database manager to store LOB data inline with regular columns.

Transaction log considerations

When using circular logging, it is important that the available log space permits dirty pages in the buffer pool to be cleaned at a reasonably low rate of speed. Changes to the database are immediately written to the log, but a well-tuned database coalesces multiple changes to a page before eventually writing that modified page back to disk. Naturally, changes recorded only in the log cannot be overwritten by circular logging. DB2 detects this condition and forces the immediate cleaning of dirty pages required to allow switching to a new log file. Although this mechanism protects the changes recorded in the log, it suspends all application logging until the necessary pages are cleaned.

DB2 works to avoid pauses when switching log files by proactively triggering page cleaning under control of the database level softmax parameter. The default value of 100 for softmax begins background cleaning when the size gap between the current head of the log and the oldest log entry recording a change to a dirty page exceeds 100% of one log file. In extreme cases, this asynchronous page cleaning cannot keep up with log activity, leading to log switch pauses that degrade performance.

Increasing the available log space gives asynchronous page cleaning more time to write dirty buffer pool pages and avoid log-switch pauses. A longer interval between cleanings offers the possibility for multiple changes to be coalesced on a page before it is written, which reduces the required write-throughput by making page cleaning more efficient.

Available log space is governed by the product of log file size and the number of primary log files, which are configured at the database level. The logfilsiz setting is the number of 4 KB pages in each log file. The logprimary setting controls the number of primary log files.

As a starting point, try using 10 primary log files that are large enough so they do not wrap for at least a minute in normal operation.

Increasing the primary log file size has implications for database recovery. Assuming a constant value for softmax, larger log files mean that recovery might take more time. Although you can lower the softmax parameter to counter this effect, consider that more aggressive page cleaning might also be less efficient. Increasing the softmax parameter offers additional opportunities for write coalescing at the cost of longer recovery time.

The default value of softmax is 100, meaning that the database manager attempts to clean pages such that a single log file needs to be processed during recovery. For best performance, we suggest increasing this value to 300 as follows, meaning that three log files might need processing during recovery:

```
db2 update db config for <yourDatabaseName> using softmax 300
```

Memory

DB2 provides a self-tuning memory feature that simplifies the task of memory configuration by automatically setting values for memory configuration parameters and sizing buffer pools. This feature is on by default and controlled with the self_tuning_mem database parameter. It is recommended to use this feature and make specific adjustments if necessary, such as the following guidelines.

Setting buffer pool sizes correctly

A *buffer pool* is an area of memory into which database pages are read, modified, and held during processing. If a required page of data is already in the buffer pool, that page is accessed faster than if the page must be read directly from disk. As a result, the size of the DB2 buffer pools is important to performance.

You can enable self-tuning for individual buffer pools by using SIZE AUTOMATIC with the **CREATE** or **ALTER** commands.

To choose appropriate buffer pool size settings manually, monitor database container I/O activity by using system tools or by using DB2 buffer pool snapshots. The amount of memory used by a buffer pool depends on two factors:

► Size of buffer pool pages
► Number of pages allocated

All buffer pools are in database global memory, which is allocated on the database machine. The buffer pools must coexist with other data structures and applications, all without exhausting available memory. In general, having larger

buffer pools improves performance up to a point by reducing I/O activity. Beyond that point, allocating additional memory no longer improves performance.

Ensuring sufficient locking resources are available

Locks are allocated from a common pool controlled by the locklist database level parameter, which is the number of pages set aside for this use. A second database level parameter, maxlocks, bounds the percentage of the lock pool held by a single application. When an application attempts to allocate a lock that exceeds the fraction allowed by maxlocks, or when the free lock pool is exhausted, DB2 performs lock escalation to replenish the supply of available locks. Lock escalation involves replacing many row locks with a single table-level lock.

Although lock escalation addresses the immediate problem of lock-pool overuse, it can lead to database deadlocks. Because a BPM system frequently updates records in the database, the probability for deadlocks resulting from lock escalations is very high. In some cases, the process application behavior can be altered to reduce pressure on the lock pool by breaking up large transactions that lock many rows into smaller transactions. It is simpler to tune the locking resources of the database first.

DB2 adjusts the locklist and maxlocks parameters automatically, by default. To tune these parameters manually, observe whether lock escalations are occurring either by examining the `db2diag.log` file or by using the system monitor to gather snapshots at the database level. If the initial symptom is database deadlocks, consider whether these deadlocks are initiated by lock escalations, as follows:

1. Check the lock escalations count in the output.:

   ```
   db2 get snapshot for database <yourDatabaseName>
   ```

2. Obtain current values for locklist and maxlocks by examining the output.:

   ```
   db2 get db config for <yourDatabaseName>
   ```

3. If necessary, alter these values. For example, if you use BPEL workload to 5000, and if you use only BPMN workload to 200 for locklist. Use 20 for maxlocks, as shown in Example 5-2.

Example 5-2   Setting BPEL values for locklist and maxlocks

```
db2 update db config for <yourDatabaseName> using locklist 5000
maxlocks 20
```

When increasing the locklist size, consider the impact of the additional memory allocation that is required. Often, the locklist is relatively small compared with memory dedicated to buffer pools, but the total memory required must not lead to virtual memory paging.

When increasing the maxlocks fraction, consider whether a larger value allows a few applications to drain the free lock pool. This drain might lead to a new cause of escalations as other applications needing relatively few locks encounter a depleted free lock pool. Often, a better way is to start by increasing locklist size alone.

Statistics

DB2 provides an automatic statistics feature that is enabled by default and is controlled with the `auto_runstats` parameter.

If necessary, statistics can be updated manually using various commands such as the following `runstats` example:

```
runstats on table <schema>.<table> with distribution and detailed
indexes
```

Table reorganization

As a result of many insert, delete, and update operations, tables and indexes can become disorganized so that the performance of queries degrades significantly. DB2 provides utilities to detect and to restructure disorganized tables and indexes.

Periodically, run the `reorgchk` utility to find disorganized tables and indexes. If the tool suggests a reorganization, run the `reorg` utility. After the reorganization, execute the `runstats` utility to reflect the reorganized tables and indexes in the optimizer statistics. Without this runstats step, the optimizer statistics of the reorganized tables and indexes will be out of date.

Automatic database maintenance tasks

The messaging engines required by BPM store data in tables that are located in the BPM database. To assign a specific messaging engine to these tables, the messaging engine uses exclusive row locks (in prior releases, exclusive table locks) on the SIBOWNER table. These locks are held during the lifetime of the messaging engine and can have a negative impact on automatic maintenance tasks, such as the DB2 health monitor, or automatic statistic updates. To avoid locking issues with this table, exclude the SIBOWNER table from all automatic maintenance tasks.

5.1.3 Oracle specific database tuning

Providing a comprehensive Oracle database tuning guide is beyond the scope of this book. However, several guidelines can assist in helping you improve the performance of Business Process Manager products when used in Oracle

environments. For a detailed description of the Oracle tools and procedures mentioned in this section, refer to the Oracle database documentation:

► Oracle Database 11g Release 2 documentation (includes a performance tuning guide):

http://docs.oracle.com/cd/E11882_01/index.htm

In addition, the following reference is useful:

► *Oracle Architecture and Tuning on AIX v2.20* white paper:

http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100883

Table spaces

When creating table spaces, consider minimizing the number of table space expansions (extends) that occur by setting large initial and autoextend sizes. This step can help produce fewer spikes in database utilization under peak load. For high load production systems, consider an extent size of 256 MB or more.

Alternatively, manually extend the table spaces during periods of relatively low activity to mitigate this issue.

File system I/O, caching, and special considerations for large objects

To achieve optimal performance with Oracle databases, LOBs can be managed separately from other data using the following approach:

► Disable concurrent I/O at the database level. For example, for Oracle on AIX set filesystemio_options to asynch.

► Mount the file systems containing data files that do *not* have LOBs with options to enable the use of concurrent I/O (taking special care that only Oracle data files will be affected).

► Move the LOBs to table spaces that reside on file systems mounted with options to enable the use of the file system cache (which are typically the default mount options).

► Partition the LOB table spaces to reduce file locking contention.

SECUREFILE versus BASICFILE

SECUREFILE is an alternative LOB storage offered by Oracle, which performs better than the traditional BASICFILE. In high throughput scenarios, consider to change the LOB columns of the hot spot tables to use SECUREFILE with default options. Example 5-3 on page 124 shows how to change a LOB column from the default storage option to SECUREFILE in a dedicated table space.

Example 5-3   Change a LOB column to SECUREFILE

```
ALTER TABLE SCOPED_VARIABLE_INSTANCE_B_T MOVE LOB("DATA") STORE AS
SECUREFILE (TABLESPACE LOBTS)
```

If using AIX JFS2 file systems, place redo logs and control files on a separate file system with the `agblksize` parameter set to 512, also mounted for concurrent I/O.

Transaction log considerations

Oracle performs an expensive checkpoint operation when switching logs. The checkpoint involves writing all dirty pages in the buffer cache to disk. Therefore, an important step is to make the log files large enough that switching occurs infrequently. Preferably, log file switches should occur rather in the range of minutes than seconds. Applications that generate a high volume of log traffic need larger log files to achieve this goal.

Memory

Oracle provides automatic memory management for buffer caches. For more information about configuring automatic memory management and for guidance on manually setting buffer cache sizes, see the following references:

► For Oracle 11g R1

http://download.oracle.com/docs/cd/B28359_01/server.111/b28274/memory.htm#i29118

► For Oracle 11g R2

http://docs.oracle.com/cd/E11882_01/server.112/e41573/memory.htm#PFGRF94264

Statistics

Oracle provides an automatic statistics-gathering facility, which is enabled by default. One approach for updating statistics manually on all tables in a schema is to use the procedures in the `dbms_stats` package. Using the default sampling size is a good choice for most of the tables. However, for a few tables that are used by search queries, it is beneficial to gather more fine grained statistics. See the workload-specific sections on tuning task lists.

For more information about the `dbms_stats` package, refer to the Oracle product documentation.

Tuning queries

In this section we examine the following details:

► Using the Oracle SQL Tuning Advisor for long running SQL statements
► Maintaining correct table indexing

Using the Oracle SQL Tuning Advisor for long running SQL statements

If analysis shows that a particular SQL statement is taking a long time to execute, it might be because the Oracle database is executing the SQL in a non-optimal manner. The Oracle SQL Tuning Advisor can be used to optimize the performance of long running SQL statements. Use the following methodology to identify, and improve, the performance of these SQL statements:

► Identify long-running SQL statements by using an automatic workload repository (AWR) report, or through the Oracle Enterprise Manager.

► Assess the statement whether or not it needs specific tuning.

► Run the SQL Tuning Advisor against the long-running SQL statements.

► Evaluate and accept (as is appropriate) the recommendations from the SQL Tuning Advisor.

Maintaining correct table indexing

The SQL Access Advisor, available from the Enterprise Manager, provides suggestions for schema changes, including changes to indexes. You can find the SQL Access Advisor by starting at the database home page, then following the Advisor Central link in the Related Links section at the bottom of the page.

Additional data source configurations

For Oracle databases, each prepared statement can take a significant amount of Java heap space (in some cases, 140 KB per statement have been observed in Business Process Manager solutions). As such, do the following tasks:

► Judiciously set prepared statement cache sizes to be large enough to improve performance (particularly for the Business Process Choreographer data source for BPEL processes, which realize significant benefit from the prepared statement cache).

► Monitor Java heap utilization to ensure that the prepared statements are not using too much heap space or causing OutOfMemory exceptions.

Specific tuning parameters

For our internal performance evaluation of the Business Process Manager 8.0 Process Server, we changed the following Oracle database settings from their default. This approach is a useful starting point for tuning the Oracle database,

but follow the suggested methodology for your applications and configuration. The first four settings in the following list generally vary by workload and volume, so consult with your database administrator to determine appropriate values. Also, the last two settings work well for Business Process Manager solutions where the Oracle undo operation is typically not used, but might not work well for non-Business Process Manager databases:

► processes: 500
► open_cursors: 1000
► undo_retention: 200
► _undo_autotune: FALSE

References to the IBM Business Process Manager online IBM Knowledge Center

The following website explains how to create an Oracle database for Business Process Choreographer. It provides details about Business Process Choreographer database (BPEDB) creation, including pointers to useful creation scripts for a production environment:

http://www.ibm.com/support/knowledgecenter/#!/SSQH9M_7.0.0/com.ibm.websphere.bpc.doc/doc/bpc/t2codbdb.html

The following website explains how to specify initial Oracle database tuning settings:

http://www.ibm.com/support/knowledgecenter/#!/SSQH9M_7.0.0/com.ibm.websphere.bpc.doc/doc/bpc/t5tuneint_spec_init_db_oracle.html

5.1.4 SQL Server specific database tuning

A complete tuning guide for SQL Server is beyond the scope of this book. This chapter gives some advice about how to tune a SQL Server database for improved performance with Business Process Manager Environments. For more detailed information about SQL Server tuning, refer to the following references and additional resources:

► Microsoft Performance Monitoring and Tuning How-to Topics

   http://technet.microsoft.com/en-us/library/ms187830(v=sql.105).aspx

► Microsoft How to: Tune a Database

   http://technet.microsoft.com/en-us/library/ms190392(v=sql.105).aspx

Monitoring system performance

Monitoring the performance of a SQL Server database is the starting point for the identification and analysis of potential bottlenecks, which might lead to reduced system performance.

To get an overview about system health and runtime behavior, use the SQL Server Management Studio tools. They enable the user to administer and configure databases and to monitor the system.

Activity Monitor

The Activity Monitor gathers and displays information about the database server. It gives an overview about processor utilization, worker thread states, database I/O and other relevant factors. It also contains a list of recent expensive queries.

Reports

The SQL Server Management Studio reports give an overview about performance relevant issues like the most expensive queries, system memory consumption, transaction and locking behavior, and others. SQL Server comes with a couple of predefined reports and offers the possibility to define custom reports for specific requirements. The Performance Dashboard Reports for SQL Server is a predefined custom report for performance optimization. The Performance Dashboard Reports can be downloaded at the following location:

http://www.microsoft.com/en-us/download/details.aspx?id=29063

Use the Database Engine Tuning Advisor

The Database Engine Tuning Advisor is a stand-alone software tool for optimizing the SQL Server database. It suggests indexes for potentially expensive queries, partitioning, and indexed views. The Database Engine Tuning Advisor suggests which columns should be part of an index and also suggests the optimal order within the index. It covers DDL-statements like UPDATE, INSERT, and DELETE.

The Database Engine Tuning Advisor takes different types of input for the tuning analysis.

Usage of the database plan cache

This variant does not need any additional user action as the Tuning Advisor analyzes the currently cached execution plans in the plan cache and makes suggestions how the physical database structure may be optimized by the use of additional indexes, partitioning, or other tuning techniques.

It is important that the database is in a representative state comparable to a realistic production environment. Clear the plan cache before running a

representative workload on the system so that the outputs of the Tuning Advisor fit the needs of the production environment.

Usage of a prerecorded workload

The workload capture can be done using the SQL Server Profiler, which is able to record a large variety of events inside the SQL Server. It contains predefined templates for the most common analysis tasks on a database system like deadlock-analysis or a detailed analysis of TSQL durations. The Tuning Advisor SQL Server Profiler contains a Tuning Template that can be used to capture workloads, which can later be analyzed for tuning purposes. It is important that the workload represents a realistic workload scenario for the production environment. Unrealistic workloads can lead to suggestions that might even lead to a performance degradation on the target system.

For additional information about the Database Engine Tuning Advisor, refer to the official Microsoft SQL Server resources:

http://msdn.microsoft.com/en-us/library/ms166575.aspx

Optimize database statistics

Database statistics are inputs for the SQL Server query optimizer. These statistics contain information about individual columns or sets of columns.

The up-to-dateness of statistics data is important for the Query Optimizer. As statistics depend on various factors like the number of rows in a table or the distribution of records in a column database, statistics need to be updated on a regular basis to guarantee the best possible plan generation.

> **Note:** Automatic statistics gathering is on by default since SQL Server 2000 and is triggered on demand by the optimizer.

If the optimizer needs statistics about an object and there is no information or the statistics data is out of date, the system gathers new statistics for this object. For additional information about how this statistics gathering works, which parameters can be changed by the user, and how statistics updates can be disabled or manually invoked, refer to the documentation available at:

http://technet.microsoft.com/en-us/library/dd535534(v=sql.100).aspx

Identify additional indexes

During the computation and evaluation of potential query plans, the database optimizer stores information that can be used for new additional indexes to optimize execution plans.

The missing indexes package *sys. sys.dm_db_missing_index-package* uses this information and suggests indexes based on the generated information from the query optimizer.

The query optimizer suggestions can be accessed in several ways:

► By invoking the *sys. sys.dm_db_missing_index-package* procedures
► By using the report mechanism of the SQL Server Management Studio
► By reviewing the execution plans with the showplan-functionality

Optimize worker threads

The number of worker threads defines the number of clients that are able to connect to the database server at a time. SQL Server uses thread pools to efficiently manage the available threads and reuse instead of destroying and re-creating them on demand.

The max worker threads parameter controls the maximum pool size.

Additional information: By default, max worker threads are set to 0. This value causes SQL Server to automatically compute the pool size on server start. The most relevant factors for this computation are the number of available processors inside the system and the architecture of the operating system.

Optimizing the number of worker threads

Depending on the workload, changing the maximum pool size manually can improve performance.

Hints for too small pool sizes are frequently appearing THREADPOOL wait events in the server reports. These wait events are thrown in situations when there are no worker threads left in the system to handle incoming client requests. In such setups, manually increasing the max worker threads parameter can improve system performance. For additional information about this topic, refer to the SQL Server documentation, available at:

https://technet.microsoft.com/en-us/library/ms187024%28v=sql.105%29.asp
x

Tune the buffer cache

The size of the buffer cache available for the database is important for the system performance because it has an impact on read and write operations.

Modifying the Buffer Cache

For more information about manually modifying the size of the virtual address space of the buffer cache, refer to SQL Server documentation at:

http://technet.microsoft.com/en-us/library/aa337525(v=sql.105).aspx

5.2 Tuning and recommendations for BPMN workload

In this section, we describe tuning and recommendations for BPMN workload that have a direct influence on the database workload:

► Hot spot tables
► Task list queries (saved searches)
► Index considerations for Oracle
► Index recommendation for the Performance Data Warehouse

5.2.1 Hot spot tables

The following database tables have a significant impact on database performance in terms of cardinality, update frequency, and space on disk. It is important that the *optimizer statistics* are current for these tables and that query execution plans are optimized for SQL queries reading from these tables:

► LSW_TASK

► LSW_TASK_EXECUTION_CONTEXT

► LSW_BPD_INSTANCE

► LSW_BPD_INSTANCE_DATA

► LSW_BPD_INSTANCE_VARIABLES

► LSW_EM_TASK

► LSW_USR_XREF

► LSW_USR_GRP_XREF

► LSW_PO_VERSIONS

5.2.2 Task list queries (saved searches)

If slow response times or high database usage are observed when performing business data searches, perform the following tasks:

► Ensure that you do not use more than 10 business data variables in a single query. If there are more than 10 of these variables, consider to review your business logic. Our experience shows that 10 or less is generally sufficient.

► Enable Process Search optimizations through the Saved Search Accelerator Tools for the Process Server. This technique is often much faster, and uses fewer database resources, than the default mechanism. The Process Search optimizations can be enabled by using command-line tools, as described at the following location:

http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r0m1/index.jsp?topic=%2
Fcom.ibm.wbpm.main.doc%2Ftopics%2Fctuneprocportal.html

► If the SQL statement for a search query appears to be in the top SQL list of your database monitoring tool, check the access plan of that statement. Examine if the query uses a table scan on one of the hot spot tables. If the optimizer statistics are out of date, gather current optimizer statistics. If the database continues to use a table scan, consider to add a more suitable index. Use database tools such as the DB2 design advisor to find a more suitable index.

5.2.3 Index considerations for Oracle

In our internal evaluation, we found that the following indexes are often helpful for BPMN business processes and task lists (saved searches) if Business Process Manager runs on Oracle. Consider these indexes as a starting point for defining indexes in the *process server database*:

► CREATE INDEX TASK_BIS ON LSW_TASK("BPD_INSTANCE_ID","STATUS");

► CREATE INDEX TASK_UGI on LSW_TASK("USER_ID","GROUP_ID","TASK_ID");

► CREATE INDEX TASK_UI on LSW_TASK("USER_ID","TASK_ID");

► CREATE INDEX TSC_UPPER ON LSW_TASK_STATUS_CODES(UPPER("NAME"), "STATUS_VALUE");

► CREATE INDEX ISC_UPPER ON LSW_BPD_STATUS_CODES(UPPER("NAME"), "STATUS_ID");

The index list has been evaluated with BPM 8.5. Some of the indexes or similar indexes have been included in later releases of BPM.

5.2.4 Index recommendation for the Performance Data Warehouse

Add the following indexes to the *performance data warehouse* database:

```
CREATE INDEX PDW.TASK_IDS ON
PDW.LSW_TASK("SYSTEM_ID","SYSTEM_TASK_ID");
```

5.3 Tuning and recommendations for BPEL workload

In this section, we describe tuning and recommendations for BPEL workload:

► Hot spot tables
► Task and process list queries (query tables)
► Drop unutilized indexes
► Use administrative task optimization
► Follow the operating guidelines for the BPM database
► Special tuning considerations for DB2 for Linux, UNIX, and Windows

5.3.1 Hot spot tables

For BPEL workload, the following tables are frequently updated and the cardinality can grow quickly. It is essential to monitor the cardinalities of these tables already in your pre-production environment. Shortcomings in the process model can lead to a scalability bottleneck, easily, and if you recognize a fast growth rate in one of these tables, this is most likely an indication for a suboptimal process model.

In a production system, maintain the optimizer statistics and monitor the cardinalities of the following tables. On DB2, schedule a periodic reorgchk for these tables. Refer to "Statistics" on page 122.

► `PROCESS_CONTEXT_T`

► `PROCESS_INSTANCE_B_T`

► `SCOPED_VARIABLE_INSTANCE_B_T`

► `SCOPE_INSTANCE_B_T`

► `ACTIVITY_INSTANCE_B_T`

► `TASK_INSTANCE_T`

► `WORK_ITEM_T`

► `SWI_T`

► `EVENT_INSTANCE_B_T`

5.3.2 Task and process list queries (query tables)

Do not use the old query API for process and task lists. Use the *query table API* instead. The query tables API has better query response times compared to the query API.

Use inline custom properties for most efficient filtering

For most efficient filtering, use inline custom properties for entities instead of custom properties or queryable variable instances. Inline custom properties are stored in the same table as the entity, for example the task table, so a join is not necessary. See **Query tables and query performance** in the online IBM Knowledge Center:

http://www.ibm.com/support/knowledgecenter/#!/SSFTDH_8.5.0/com.ibm.wbpm
.bpc.doc/topics/c6bpel_querytables_perform.html

Do not use too many queryable variables or custom properties

Although you can achieve good response times with query tables, you should limit the number of queryable variables or custom properties in your query table. For typical task lists, not more than 10 queryable variables or custom properties is a reasonable number. If you use a higher number, especially in combination with filtering, you might observe a performance degradation.

Use the optimizeForFiltering option when applicable

To optimize queries that include or filter on multiple properties of an attached supplemental query table, use the **optimizeForFiltering** option in the query table builder tool. This option results in a single inner join between the supplemental and the primary query tables instead of multiple correlated sub queries.

Apply filters directly in the query table definition

Instead of passing filters at run time, use static filters in the query table definition. You can use *parameters* to pass an actual filter value at run time. See **Filters and selection criteria of query tables** in the online IBM Knowledge Center:

http://www.ibm.com/support/knowledgecenter/#!/SSFTDH_8.5.0/com.ibm.wbpm
.bpc.doc/topics/c6bpel_querytables_lang.html

Create indexes, if necessary

If a query table query shows up in your top SQL list, check the query execution plan of the particular query. If the query uses expensive scan operations or sorts, consider additional indexes. Due to the flexibility of the query table API, it is a necessary tuning step to define the optimal set of indexes for your queries. The out-of-the-box indexes will only support a set of well known or commonly used

queries, but not your individual query instances. Before you start to add indexes, ensure that the optimizer statistics for the tables used by the query are up to date.

Apply database optimization profiles, if necessary

If no other tuning options have lead to an acceptable result and the query access plan still includes expensive scan operations, consider to apply optimization profiles. With optimization profiles, you can for a particular query direct the optimizer to prefer for example index access over a full table scan. See "Optimization profiles" on page 115.

5.3.3 Drop unutilized indexes

If you do not use the old query API, some indexes on the `WORK_ITEM_T` are not required for queries and can be dropped, so that they do not need to be maintained by the database. To drop these indexes, use the `dropClassicWIIndexes.sql` script. If you do not use the query API or the query table API at all, you can use the `OptionalUpgradeIndexes.sql` script to drop indexes that support these APIs.

5.3.4 Use administrative task optimization

If you do not explicitly use administrative tasks, for example if you do not use a custom client application that uses administrative tasks, you can dramatically reduce the number of tasks and work items in your database tables, in particular `TASK_INSTANCE_T`, `WORK_ITEM_T`, `SWI_T`. Using the option useProcessAdminAuthorizationOnly is the most efficient way to reduce the database load without changing application logic. Administrative task optimization can also be switched on later. See *Optimizing BPEL process administration*:

http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r5m0/topic/com.ibm.wbpm.bpc.doc/topics/t4adprocessovr.html

5.3.5 Follow the operating guidelines for the BPM database

To maintain a healthy BPM system, it is essential to maintain a healthy BPM database. Monitoring the hot spot tables, the buffer pool hit ratio, the top SQL queries, maintaining optimizer statistics, and de-fragmenting tables are important tasks that need to be done periodically. Refer to the following IBM developerWorks article:

http://www.ibm.com/developerworks/websphere/library/techarticles/0912_grundler/0912_grundler.html

5.3.6 Special tuning considerations for DB2 for Linux, UNIX, and Windows

In this section we describe:

► How to tune inline LOB columns
► How to change the volatile setting
► Where you can find references for initial database settings

Tune inline length of large objects if the block size of your table spaces is bigger than 8 K

The database tables required for BPEL workload, are laid out for a page size of 8 K. LOB columns are stored in a separate segment outside of the rows. If the tables are assigned to a table space with a larger page size than 8 K, there is some space available so that LOB columns can be stored inside the row, which can provide a big performance benefit.

Table 5-1 shows the most frequently used LOBs for BPEL workloads and the suggested inline LOB length for a 32 K page size.

Table 5-1   Inline LOB lengths

Table and LOB column	32 K inline length
ACTIVITY_INSTANCE_B_T.UNHANDLED_EXCEPTION	16.384
CORRELATION_SET_PROPERTIES_B_T.DATA	1.024
EVENT_INSTANCE_B_T.MESSAGE	8.192
GRAPHICAL_PROCESS_MODEL_T.ID_MAPPING	16.384
NAVIGATION_EXCEPTION_T.PROCESS_EXCEPTION	8.192
PROCESS_CONTEXT_T.SERVICE_CONTEXT	16.384
PROCESS_INSTANCE_B_T.UNHANDLED_EXCEPTION	8.192
REQUEST_INSTANCE_B_T.REPLY_CONTEXT	16.384
RESTART_EVENT_B_T.INPUT_MESSAGE	8.192
SCHED_TASK.TASKINFO	16.384
SCOPE_INSTANCE_B_T.BPEL_EXCEPTION	8.192
TASK_CONTEXT_T.SERVICE_CONTEXT	16.384
VARIABLE_SNAPSHOT_B_T.DATA	16.384

Table and LOB column	32 K inline length
AWAITED_INVOCATION_T.CORRELATION_INFO	8.192
SAVED_ENGINE_MESSAGE_B_T.ENGINE_MESSAGE_L	16.384
TASK_MESSAGE_INSTANCE_T.DATA	16.384
SCOPED_VARIABLE_INSTANCE_B_T.DATA	16.384

You can change the inline length parameter of the LOB so that the LOB will be stored in the row, if there is enough free space available. Use the following command to specify the inline length of a LOB column:

```
ALTER TABLE SCOPED_VARIABLE_INSTANCE_B_T ALTER COLUMN "DATA" SET INLINE
LENGTH 16384
```

Volatile

Many of the tables required for BPEL workload are specified as *volatile*. Most of these tables are not volatile, but with the volatile flag enabled, locking issues can be avoided if the tables are empty as they are in freshly installed systems. If the volatile flag is enabled, the DB2 optimizer favors index access over table scans without using the optimizer statistics. If the cardinality grows, it can be beneficial to remove the volatile flag of the core BPEL tables, so the DB2 optimizer can use the optimizer statistics and the full set of available indexes on a table to compute the optimal query execution plan.

Use the following command to change the volatile setting:

```
ALTER TABLE ACTIVITY_INSTANCE_B_T NOT VOLATILE
```

For more references, visit the following resources:

► DB2 IBM Knowledge Center (SQL Reference):

   http://www.ibm.com/support/knowledgecenter/#!/SSEPGG_9.7.0/com.ibm.d
   b2.luw.sql.ref.doc/doc/r0000888.html?cp=SSEPGG_9.7.0%2F2-10-6-26

► BPM Support technote (Explanation and Scripts):

   http://www.ibm.com/support/docview.wss?uid=swg21425402

Initial database settings

The following website explains the specification of initial DB2 database settings and provides examples of creating table spaces for the BPEDB. It also contains useful links for planning and fine-tuning the BPEDB:

http://www.ibm.com/support/knowledgecenter/#!/SSQH9M_7.0.0/com.ibm.webs
phere.bpc.doc/doc/bpc/t5tuneint_spec_init_db_settings.html

The following website explains how to create DB2 databases for Linux, UNIX, and Windows for BPEL business processes. The website gives details about BPEDB creation, including pointers for creating scripts for a production environment:

http://www.ibm.com/support/knowledgecenter/#!/SSQH9M_7.0.0/com.ibm.websphere.bpc.doc/doc/bpc/t2codbdb.html

5.4 Suggestions for Business Monitor

This section describes how to tune and maximize the performance of Business Monitor V8.5.5 in the following sections:

► Improving concurrency by setting registry variables
► Setting lock timeout properly
► Limiting event XML to 32 KB where possible
► Using materialized query tables

5.4.1 Improving concurrency by setting registry variables

DB2 allows you to defer row locks for cursor stability (CS) or read stability (RS) isolation scans in certain cases. This deferral can last until a record is known to satisfy the predicates of a query when row locking is performed during a table or index scan.

To improve concurrency, deferring row locking until after determining that a row qualifies for a query might be possible. Usually, concurrency is improved by setting the registry variables DB2_SKIPDELETED to permit scans to unconditionally skip uncommitted deletes, and DB2_SKIPINSERTED to permit scans to unconditionally skip uncommitted inserts. Example 5-4 shows how to enable these two registry variables for the MONITOR database.

Example 5-4   Enabling registry values DB2_SKIPDELETED and DB2_SKIPINSERTED

```
db2 connect to MONITOR
db2set DB2_SKIPDELETED=ON
db2set DB2_SKIPINSERTED=ON
```

5.4.2 Setting lock timeout properly

The `LOCKTIMEOUT` parameter specifies the number of seconds that an application waits to obtain a lock. This parameter helps avoid global deadlocks for applications. If you set this parameter to 0 (zero), the application does not wait for

locks. In this situation, if the lock is unavailable at the time of the request, the application immediately receives a -911 return code.

The default value of this parameter is -1, which means that lock timeout detection is turned off.

A value of 30 seconds is a good starting value; tune as necessary after setting this value. The following example shows how to set this parameter for the MONITOR database.

Example 5-5   Starting value for LOCKTIMEOUT

```
db2 -v update db cfg for MONITOR using LOCKTIMEOUT 30
```

5.4.3 Limiting event XML to 32 KB where possible

Events that are small enough are persisted to a regular VARCHAR column in the incoming events table; large events are persisted to a BLOB column instead. In DB2, the largest VARCHAR column is 32768 bytes. Performance improves considerably when a VARCHAR column is used instead of a BLOB.

5.4.4 Using materialized query tables

When you have a deep history of data (for example, more than 10 million Monitoring Context instances), the response time for dashboard navigation in dimensional reports can degrade significantly. IBM Business Monitor V8.5.5 uses Cognos Business Intelligence V10.1.1 for its dimensional reports. Business Monitor provides a tool that can be used with DB2 to generate cube summary tables that pre-compute the values of measures for known dimensional member values. Such values include the value of the "average loan amount" measure based on values for the "customer loyalty level" dimension, say, for bronze, silver, and gold customers. DB2 calls these tables *materialized query tables*, and Business Monitor provides a scheduled service to refresh them on a periodic basis.

For details about using this capability, see the following IBM Knowledge Center topic:

http://www.ibm.com/support/knowledgecenter/SSFPJS_7.5.0/com.ibm.wbpm.mon.admin.doc/data/enable_cubesumtable_refresh.html

6

Migration considerations

In this chapter, we talk about the IBM BPM migration procedure from a performance perspective. The migration topics discussed here include both in-flight data migration between process application snapshots on the same production, and version-to-version migration between different IBM BPM versions. The intention of the chapter is to provide tuning guidance and recommendations for typical migration scenarios. We cover the following topics:

► Migration overview
► Instance migration considerations
► Version-to-version migration considerations

6.1 Migration overview

Migration in this context refers to moving applications, application data, and configuration information from one place to another. The following list explains typical scenarios that BPM users usually encounter.

► When a new version (snapshot) of a process application is deployed, users want to migrate the running process instances from the old version of the snapshot to the new one. This kind of migration is referenced as *Instance Migration*.

► When a new version of the Business Process Manager product is available, users expect to use the new version of the product to replace the old one. At the same time, they want to preserve the existing application artifacts and instance data they already have in the old version. In this case, users need to perform a series of required migration steps to ensure that existing artifacts work well on the new version of product. This kind of migration is called a *version-to-version migration*.

► For IBM BPM Advanced, an instance migration can be performed between different versions of Business Process Execution Language (BPEL) processes. Migrating a running BPEL process instance to a new version of a BPEL process only has limited impact and involves no blackout period. Therefore, we do not specifically address this kind of instance migration in this chapter. For details about how to migrate running BPEL process instances, refer to the following information:

http://www.ibm.com/support/knowledgecenter/SSFTDH_8.5.0/com.ibm.wbpm.bpc.doc/topics/tprocversioning_migration.html

The difference between instance migration and version-to-version migration lies in the fact that no production installation changes are required for instance migration, where all the changes occur at the process application level in the same production runtime environment. The challenge in this migration scenario is to properly move the active process instances that are running on the old version of the process application snapshots to the new snapshot.

In contrast, the version-to-version migration involves installing a new version of IBM BPM. Therefore, all process applications and their instance data, as well as configuration information need to be migrated to the new version of the product. The migration scope of a version-to-version migration is much larger than with the instance migration.

A common point for both instance migration and version-to-version migration is that there is a *blackout period*, or system downtime, during migration, which can lead to some business functions being unavailable. For instance migration, the event manager needs to be stopped to ensure that no events are processed

during the migration. For version-to-version migration, a full production shutdown is required. Therefore, the time spent to perform the migration process is critical for BPM users. Faster migration can mean a shorter time period of unavailable business functions.

In this chapter, we focus on performance considerations for migration, so that faster migration can be achieved. We do not repeat the steps required to perform an instance migration or version-to-version migration because you can find them in the IBM Business Process Manager IBM Knowledge Center or in "Chapter 9. Overview of migration" of the IBM Redbooks publication *Business Process Management Deployment Guide Using IBM Business Process Manager V8.5*, SG24-8175.

6.2 Instance migration considerations

During an instance migration, you migrate existing active process instances from one snapshot of a process application to another. Stopping the event manager is required to stop scheduling events while the migration is in process. Therefore, a well planned and fast instance migration is important for IBM BPM users.

6.2.1 How instance migration works

In order to understand instance migration well, look at what happens behind the scene. After instance migration is initiated, the following steps are executed:

1. The snapshot with running process instances that will be migrated is deactivated.

2. The replacement snapshot is installed on the server, if it is not installed yet.

3. The migration program migrates global data to the replacement (or updated) snapshot. It uses the most recent time stamps for each exposed process variable to identify the global data to use in the replacement snapshot. The global data includes:

   – Participant groups

   – Environment variables

   – Exposed process variables

4. The migration program migrates process instances that are not complete from the old snapshot, including active, failed, and suspended instances. For each instance, it includes the following sub steps:

   – Update the snapshot ID

- – Update each business process diagram (BPD) instance execution context and initialize each new variable to default values

- – For each task in the instance

  - • Update the snapshot ID

  - • Update the task execution context with the new snapshot ID

- – Apply the orphaned token policy if it is defined

5. If the old snapshot is defined as the default snapshot, the migration program moves the default designation to the replacement (or updated) snapshot.

6. BPD instances from the old snapshot are deactivated.

7. The replacement (or updated) snapshot is activated.

There are two steps in the preceding list that can be time consuming. The first is the update step for each BPD instance execution context, and the other is for each task in the instance, to update the task execution context with the new snapshot ID.

Because the execution context for a BPD instance or task is saved as a binary large object (BLOB) data type, the migration program has to load and deserialize the BLOB data, analyze, and update it, then serialize and save it back to the database. These are expensive database operations. If the volume of BPD instances and tasks that need to be migrated is large, the overall instance migration time can be significant. IBM BPM 8.5.5 contains significant improvements in the total time to migrate BPD instances and tasks.

6.2.2 Instance migration time estimation

An instance migration requires time where the business application will be unavailable. It is hard to provide a formula to estimate this time accurately because the time spent on migration depends on many factors that can vary greatly across users and scenarios. Generally, you need to consider the following factors when you estimate the migration time:

► The performance of the system where the migration program runs. This performance typically depends on the combinations of processors, memory, disk I/O, network, and so on.

► The performance of the database server, especially the associated storage performance.

► The data volume that needs to be migrated and the size of each data item, especially the execution context of process instances and tasks. A bigger execution context usually means more resources and longer time to load, analyze, and store.

► The amount for JavaScript execution that is needed during migration. For example, some instance migration processes need JavaScript execution to initiate variables.

IBM BPM production installations can have millions of business process instances that need to be migrated to a new snapshot. It is important for BPM administrators to properly estimate the time spent on the migration so that the blackout period can be reasonably scheduled. In fact, as shown in 6.2.1, "How instance migration works" on page 141, it is not that difficult to find an estimated migration time because it is directly proportional to the *total number of process instances that need to be migrated + total number of tasks that belong to those instances*.

In order to have a rough time estimation, we need to know the average time spent on the migration of one process instance including its associated task instances. We can use a test environment to prepare an estimation. The following are general steps on how to do the time estimation using a test environment:

► Generate a number of process instances, for example 1000, that have similar process content with the production environment. The current active tasks associated with these process instances should also be proportional to the ones in production.

   You can use process admin or related wsadmin commands to get the number of process instances for a specified snapshot.

► Migrate these instances to the new snapshot on the test environment and record the time taken.

► Compute the average time per process instance. To do this divide the migration time by the sum of the process instances.

   The average time per process instance increases when the real process instance number is much higher than the process instance number used in test environment. Therefore, we can multiple the average time per process instance value by a factor as the estimated average time per process instance. For example, if the real process instance number is 10 times as high as the one in the test environment, the factor can be set to 1.1.

Use *avgTimePerInstance* to refer to the average time per process instance. For a single thread, the migration time on a production environment can be estimated as following:

```
Estimated migration time = total process instance number *
avgTimePerInstance + padding time
```

The *padding time* refers to the time spent on the migration for non-process instances or tasks processing, such as exposed environment variables,

participant groups, environment variables, and so on. The padding time can be roughly computed as following:

```
Padding time = (total number of exposed environment variables +
environment variables + participant groups + tokens) *
avgTimePerInstance
```

6.2.3 Tuning for instance migration

The instance migration program runs on one member of the application cluster. Instance migration on IBM BPM versions earlier than V8.5.5 is single threaded using single database operations. Generally, we do not need to perform any special tuning for instance migration on IBM BPM versions earlier than V8.5.5, other than ensuring the database is well tuned and provisioned.

IBM BPM V8.5.5 now supports multiple threads and several options for instance migration, aimed to improve performance. Users can tune the following items for instance migration in IBM BPM V8.5.5:

► Thread pool size for instance migration. The default size is 5. This is the number of threads that concurrently perform migration activities.

► Migrate completed tasks or not. The default value is set to skip completed tasks during migration.

► Defer to updating task execution context during migration or not.

  Deferring to updating task execution context means that the migration program will not load, analyze, and write back the task execution context to the database. This process can save much time for migration because task execution context is saved as a BLOB, which requires expensive operations to load from and save to a database. The task execution context will be updated when the task is loaded in the future.

You can set the thread size, the mode for completed task migration, and the decision to defer updating for the task execution context using the 100Custom.xml configuration file. As shown in Example 6-1, copy the following xml section to 100Custom.xml.

Example 6-1   Instance migration configuration for BPM 8.5.5

```
<server>
        <instance-migration merge="mergeChildren">
            <thread-pool-size merge="replace">5</thread-pool-size>
            <migrate-tasks merge="replace">skip-closed</migrate-tasks>
            <defer-ec>false</defer-ec>
        </instance-migration>
```

```
</server>
```

Set the value for the `thread-pool-size` to the number of threads you need. For the value of `migrate-tasks`, there are three options, which are listed and explained in Table 6-1. The `true` or `false` value for `defer-ec` determines if deferring updating task execution context is enabled or not.

Table 6-1   Options for migrate-tasks setting

Option	Explanation
all	Migrate all tasks
none	Do not migrate any tasks
skip-closed	Migrate all tasks except the closed tasks.

The following are recommended settings for these options:

► Set the thread pool size value to be a little bigger than the number of available processor cores of the application server. For example, 1.2 to 1.5 times the number of processor cores.

► If there is no requirement to preserve the completed task information, the `skip-closed` option is recommended for the task migration mode selection. This can significantly shorten the migration time.

► The `defer-ec` option is set to `false` by default. Set it to `true` only if the estimated time for instance migration is longer than the blackout period that can be tolerated.

6.2.4 Recommendations for instance migration

In this section, we provide recommendations for instance migration to minimize downtime.

Plan for instance migration

When a new version of a process application needs to be deployed, you need to make a choice for the existing process instances on the current snapshot:

► Leave them on the current snapshot.

   The instances continue to run based on the current business process definitions. No migration needed.

► Migrate them to new a snapshot.

   The instances run based on the new version of business process definition. Instance migration is needed.

At the beginning of a business process definition design, you need to consider the following:

► Do current active BPD instances need to be migrated when a new version of a process application is deployed?

► Do you need to preserve the task history information after tasks are completed?

By default, the IBM BPM server preserves completed task information as a BLOB in the database for audit purposes. The instance migration program migrates all the preserved information of the completed tasks for the process instances that need to be migrated. Large volumes of completed tasks can greatly impact the performance of instance migration. If you do not have specific requirements for auditing completed tasks, you can do the following for your business process definition in Process Designer:

► For system tasks, select *Delete task on completion* if you want to run an automated service that does not require routing. When you select this check box, audit data for the task is not retained by Process Server. By default, this option is enabled in IBM BPM 8.5.5.

► For system tasks and human tasks, select *Clean State* if you want to clear the runtime execution state of an activity after it is completed. By default, this option is disabled.

Purge application data before instance migration

Purging the application data, especially the completed instances that you do not need to preserve, can dramatically decrease the migration load. The following actions are important for instance migration performance:

► Purge completed process instances.

This action can remove all runtime data in the Process Server database that is associated with the completed business process definition instances. You can use `BPMProcessInstancesCleanup` wsadmin command to perform the purging.

► Purge completed tasks.

This action deletes all completed tasks from the Process Server database. You can use the Task Cleanup utility delivered in the Process Administrative console to perform the purging. The Task Cleanup utility is available in the Server Admin area of the Process Administrative console. For details about how to use this utility, see the IBM Knowledge Center for more details:

http://www.ibm.com/support/knowledgecenter/SSFPJS_8.5.0/com.ibm.wbpm.admin.doc/adminguide/topic/managing_procsvr_caches_dbs_B.html?cp=SSFPJS_8.5.0&lang=en

Perform iterative testing for instance migration

Successful instance migration within a reasonable time frame is important for IBM BPM users. It is valuable to perform iterative testing in your test environment using different tuning options to find the best solution before taking the instance migration to your production environment.

6.3 Version-to-version migration considerations

Version-to-version migration refers to moving applications, application data, and configuration information from an earlier version of the BPM product to a newer version. In this section, we talk about migrating from earlier BPM versions, including:

► IBM Teamworks® 6.x
► WebSphere Lombardi Edition (WLE)(7.1.x or 7.2.x)
► WebSphere Process Server (6.2.x or 7.0.0.x)
► IBM BPM 7.5.x or 8.0.x to IBM BPM 8.5.0.1 or 8.5.5.

As with the Instance Migration discussion, we focus on performance-related topics.

6.3.1 Overview of version-version migration

IBM BPM 8.5.0.1 or 8.5.5 migration utilities support two major migration paths. One path is to migrate from IBM BPM (8.x or 7.5.x), WebSphere Process Server (6.2.x or 7.0.0.x), or WebSphere Lombardi Edition (WLE) (7.1.x or 7.2.x). For simplicity, we refer to this path as migrating from a previous BPM, WebSphere Process Server, or WLE release.

The other path is to migrate from Teamworks 6.x. We refer to this path as migrating from Teamworks 6.x.

The two migration paths have completely different migration strategies and implementations. Therefore, we talk about them separately in the following sections.

Overview of migrating from previous BPM, WebSphere Process Server, or WLE releases

Table 6-2 on page 148 lists all supported paths of migrating from previous BPM, WebSphere Process Server, and WLE releases.

Table 6-2   Supported migration paths of migrating from previous BPM, WebSphere Process Server, or WLE

Source	Target (IBM BPM 8.5.0.1 or 8.5.5)
IBM BPM Standard 7.5.x or IBM BPM Standard 8.0.x	IBM BPM Standard or IBM BPM Advanced
IBM BPM Advanced 7.5.x or IBM BPM Advanced 8.0.x	IBM BPM Advanced
WebSphere Lombardi Edition 7.1.x or WebSphere Lombardi Edition 7.2.x	IBM BPM Standard or IBM BPM Advanced
WebSphere Process Server 7.0.0.x or 6,2,x	IBM BPM Advanced

The detailed steps for these migration paths can be found in the Interactive Migration Guide in the IBM Knowledge Center of IBM BPM. We do not repeat the content here but only list the high-level steps in Table 6-3.

Table 6-3   High-level steps for migrating from previous BPM, WebSphere Process Server, or WLE

Step	Migration Task	Source /Target
1	Verify that the hardware and operating system is supported.	Target
2	Install IBM BPM V8.5.0 and upgrade to V8.5.0.1 or 8.5.5.	Target
3	Package remote migration utilities and copy them to source (only in the case of a remote target).	Target
4	Update the `migration.properties` file with the configuration information for the source environment.	Source
5	Extract all database-related information from the source environment.	Source
6	Configure the target IBM BPM 8.5 environment by using the `BPMConfig` command and properties file for topology.	Target
7	Check the readiness of source environment for migration by using the `BPMExtractSourceInformation -precheck` command (Advanced version or WebSphere Process Server only).	Source
8	Run the `BPMManageApplications` command to disable automatic starting of all your applications and schedulers. This action confirms that no new events or processes in the source system and no scheduler tasks are used during the migration process.	Source
9	Take a snapshot of the source environment by using the `BPMExtractSourceInformation` command.	Source

Step	Migration Task	Source /Target
10	Stop all running processes, including deployment environment, node agent, and deployment manager.	Source
11	Copy the snapshot and migration properties files to the target computer. This is required only if your target is on a different system.	Source
12	Back up all your old databases, including customer's system of records.	Source
13	Run the `BPMGenerateUpgradeSchemaScripts` command to generate the SQL scripts to initialize the newly configured database and the upgrade SQL script that is used by the database upgrade command.	Target
14	Run the database upgrade script to upgrade source databases to IBM BPM 8.5.0.1 or 8.5.5.	Target
15	Disable the automatic starting of all applications and schedulers, start the target environment and import the migration snapshot by running the `BPMMigrate` command.	Target
16	Apply custom configurations to target environment. Use the `PerformanceTuningParameters.properties` file that is generated by the `BPMExtractSourceInformation` command to understand the custom configurations from the source environment.	Target
17	Complete configuration for IBM Business Process Manager, Business Process Choreographer (Advanced version or WebSphere Process Server only) and Business Space (WebSphere Process Server only).	Target
18	Restart the environment and verify migration.	Target

The production server needs to be down from step 8 to step 18. Step 14 *Run the database upgrade script to upgrade source databases to IBM BPM 8.5.0.1* is the most time-consuming step for migrating large volumes of application data. Step 15 involves running the `BPMMigrate` command. For IBM BPM Advanced, if you need to migrate process applications for a BPEL process, the command may take a long time to execute depending on the number and complexity of the process applications for the BPEL process. This is because the command needs to deploy the process applications for a BPEL process to the installation of the new BPM version.

Overview of migrating from Teamworks 6.x

Migrating from Teamworks 6.x to IBM BPM 8.5.0.1 or 8.5.5 is a separate migration path. The major characteristic for this path is that the migration is required to be executed side by side, meaning the source and target

environments are completely separate. The high-level steps are summarized in Table 6-4.

Table 6-4   High-level steps for migrating from Teamworks 6.x

Step	Migration Task	Source /Target
1	Install IBM BPM 8.5.0 and upgrade to 8.5.0.1 or 8.5.5 (both Process Center and Process Server for test, stage, and production).	Target
2	Prepare artifacts for migration: Install and run the Upgrade Readiness Check tool to analyze your existing process assets and resolve issues before you migrate.	Source
3	Migrate the Teamworks 6.x development environment to IBM Business Process Manager V8.5.0.1 and export artifacts from Teamworks 6 and Import them to Process Center of IBM BPM 8.5.0.1 or 8.5.5.	Source and Target
4	Edit the `upgrade.properties` file by providing the correct values for the source (Teamworks 6.x) and target (IBM BPM 8.5.0.1) environments.	Target
5	Map the Teamworks 6 artifacts to IBM Business Process Manager V8.5 artifacts and stop source (Teamworks 6).	Target
6	Run the **upgrade** command to complete the migration.	Target
7	Restart the target environments.	Target

The production server is required to be down from step 5 to step 7. Step 6 is expected to be the most time consuming step for migrating large volumes of application data.

6.3.2 Time estimation for version-to-version migration

The IBM BPM production server needs to be down during a version-to-version migration. A good estimation of system downtime can be important for IBM BPM users. In this section, we introduce methods to estimate the system downtime.

Time estimation for migrating from previous BPM, WebSphere Process Server, or WLE

From the high-level migration steps explained in Table 6-3 on page 148, step 14 *Run the database upgrade script to upgrade source databases to IBM BPM 8.5.0.1* is the most time consuming step. The time spent on this step is directly proportional to the volume of process instances, task instances, and durable event messages. The time spent on other steps is not directly related to application data. Those steps require constant time, which is related to the performance of the individual systems that are used for the migration. Therefore,

estimating the time required for step 14 is important for the overall time estimation.

There are two ways to estimate the time for database migration:

► Generate test data in a test environment and use that to project the migration time for production based on the time consumed in your test environment.

► Use a backup database of your production deployment to perform migration testing. Your test environment should resemble the production environment as closely as possible. This way the migration time that you measure in your test environment is similar to the production migration.

The first choice provides an easy and fast way for a good estimation. The real production migration time can be projected by using the migration time from the test environment multiplied by the ratio of your real application data volume (number of process instances plus task number plus durable number of event messages) to the application data volume of the test environment. The main drawback for this method is that the estimate is not very accurate for the real production environment. The accuracy depends on the following factors:

► If the test environment is similar to the production environment.

► If the tuning parameters on the test environment are the same as for the production environment.

► If the test application data has similar content compared to the production environment.

► If the database server and storage implementation is similar to the production environment.

Using the backup database of the production environment to estimate migration time has the following advantages:

► The estimated migration time can be very accurate if the test environment is similar to the production environment because it uses the actual application data.

► Because the test can be repeated as many times as you like, it is possible to apply different tuning options to compare the time consumed on the migration and choose the best solution for the final production migration.

► This test can find any potential problems of the migration, and you can address those in advance, which can ensure that the final migration is safe and successful.

A side effect of this kind of time estimation is that you can spend much time to come up with a proper estimate, especially for large volumes of application data. It can potentially take several hours for one migration test if you have to account

for millions of application data records. However, this choice is our recommended approach to properly estimate migration time.

Time estimation for migrating from Teamworks 6.x

Compared with the migration from previous BPM, WebSphere Process Server, and WLE, the main difference for the migration from Teamworks 6.x to BPM 8.5.0.1 or 8.5.5 is that it is a side-by-side migration process. The target environment running BPM 8.5.0.1 or 8.5.5 is installed in parallel with the running source Teamworks 6.x. The upgrade scripts from the migration utilities read data from the source database, perform a transformation, and then write the data to the target database system. The time estimation methods used for migrating from previous BPM, WebSphere Process Server, and WLE also apply to this migration path.

We recommend using a backup database of the production environment to perform time estimation because it is the most accurate way to estimate the time. At the same time that you test the migration, you can apply different tuning options to find the best solution. In addition, it is possible to find any migration problems during the test and address them in advance.

6.3.3 Tuning considerations for version-to-version migration

Migrating from previous versions to BPM 8.5.0.1 or 8.5.5 is typically a time-consuming process for large volumes of application data, although significant improvements were made in BPM 8.5.5. Appropriate tuning before the fact can accelerate the overall migration process.

Tuning for migration from previous BPM, WebSphere Process Server, or WLE

Running the database upgrade script to upgrade the source databases to IBM BPM 8.5.0.1 or 8.5.5 is the most expensive step in the migration. The time spent on this step is proportional to the volume of the application data in the source database. The main tuning is focused on this step, and the tuning options are:

► The *thread size* for the DB upgrade utility.

   The thread size defines how many threads can work concurrently to perform database upgrade processing. The parameter for thread size can be found in the `BPM_home/util/dbUpgrade/upgrade.properties` file.

   The `worker.thread.size` property is used for thread size and the default value is 1. The recommended value equals the number of processor cores of the machine that runs the migration utility.

► The *batch size* for DB upgrade utility.

The batch size defines how many records in the database can be processed at one time. This parameter can be found in the `BPM_home/util/dbUpgrade/upgrade.properties` file.

The `database.batch.size` property default value is 1000. This value should be set based on the average complexity of persistent data and heap size settings of the JVM migration utility. Batch size can be increased with smaller business variables and larger heap size.

► *Heap size* settings for DB upgrade Java process.

By default, the heap size settings are not specified for the DB upgrade Java process. That means it only uses the default settings of the JVM. The default minimum and maximum heap sizes are platform-dependant. Refer to related JVM documents for details.

We recommend specifying the heap size before migration. The heap size can be set in the `BPM_home/bin/DBUpgrade.sh` file.

The following example shows how to change the heap size of the JVM for the **DBUpgrade** command:

```
JVM_HEAPSIZE_OPTIONS="-Xms512m -Xmx2048m
$JAVA_EXE $JVM_HEAPSIZE_OPTIONS -cp $CLASSPATH
```

Maximum heap size should be set to more than

```
average size of business variables * batch size * thread size
```

► *Log levels* for migration.

Logging during migration can consume processor and I/O resources. Fine granularity logging can impact migration performance drastically. We suggest not to set the log level to *fine* or higher during formal migration. Use the `BPM_home/util/dbUpgrade/logging.properties` file to set the log level for migration.

Tuning for migration from Teamworks 6.x

The database upgrade utility in the `BPM_home/BPM/Lombardi/tools/upgrade/upgrade_6x/` folder is responsible for migrating from Teamworks 6.x to IBM BPM. If the target version is IBM BPM 8.5.0.1 or 8.5.5, the DB upgrade utility implementation is single threaded and uses single database operations. The following items can be tuned before migration.

► *Heap size* settings for the DB upgrade Java process.

By default, the heap size settings are not specified for the DB upgrade Java process. That means it only uses the default settings of the JVM. The default

minimum and maximum heap sizes are platform-dependant. Refer to related JVM documents for details.

We recommend specifying the heap size before migration in the `BPM_home/BPM/Lombardi/tools/upgrade/upgrade_6x/upgrade_6x.sh` file.

The following example shows how to change the heap size of the JVM for the **DBUpgrade** command:

```
JVM_HEAPSIZE_OPTIONS="-Xms512m -Xmx2048m
$JAVA_EXE $JVM_HEAPSIZE_OPTIONS -cp $CLASSPATH
```

► *Log levels* for migration.

Logging during migration can consume processor and I/O resources. Fine granularity logging can impact migration performance drastically. We suggest not to set log level to *fine* or higher during formal migration. Use the `BPM_home/BPM/Lombardi/tools/upgrade/upgrade_6x/logging.properties` file to set the log level for migration.

For IBM BPM 8.5.5, the database upgrade utility supports multiple threads and batch database operations. Users can tune the following additional items:

► The *thread size* for DB upgrade utility.

The thread size defines how many threads can work concurrently to perform database upgrade processing. The parameter for thread size can be found in the `BPM_home/BPM/Lombardi/tools/upgrade/upgrade_6x/upgrade.xml` file.

The recommended value equals the number of processor cores of the machine that runs the migration utility.

► The *batch size* for DB upgrade utility.

The batch size defines how many records of data in the database can be processed at one time. The parameter can be found in the `BPM_home/BPM/Lombardi/tools/upgrade/upgrade_6x/upgrade.xml` file.

The `database.batch.size` property default value is 1000. This value should be set based on the average complexity of persistent data and heap size setting of the JVM migration utility. Batch size can be increased for smaller business variables and larger heap size.

6.3.4 Recommendations for version-to-version migration

In this section, we explain some common recommendations for both migration paths:

► The migration utilities support both topology changes and hardware changes. Therefore, this is a good opportunity to review topologies and hardware configurations and make appropriate changes based on the requirements for

the new version setup. Refer to Chapter 2, "Architecture best practices" on page 5 to choose the best topology and hardware for your solution.

- ► Have a well thought-out plan before migrating.
  - – Perform a proper migration test before migrating your production environment to get an estimated migration time and find potential problems in advance.
  - – Do iterative migration testing to optimize tuning parameters to meet your target migration time.
  - – Have a good migration plan including scheduled production downtime, as well as a production backup and restore plan in case anything goes wrong during migration.
- ► Perform purging before the actual migration to decrease the load of migration. The purging includes the following:
  - – Undeploy process applications no longer needed for BPEL processes.
  - – Delete the process application snapshots that are no longer required.
  - – Purge completed process instances by using the appropriate wsadmin commands or stored procedures.
  - – Purge completed tasks for running instances by using task cleanup utilities delivered in the Process Administration console.
  - – Purge durable subscription messages by using the appropriate wsadmin commands or stored procedures.
  - – Purge the performance database.
- ► Set appropriate log level for migration. Try to avoid setting the log level to fine or higher because it can have a negative impact on migration performance.
- ► Determine if indexes have been created for important tables in the source Process Server database. Indexes are important for migration performance. The indexes that you need to pay close attention to are shown in Table 6-5.

Table 6-5   Check the following source indexes before migration

Tables	Columns
LSW_BPD_INSTANCE_DATA	BPD_INSTANCE_ID
LSW_BPD_NOTIFICATION	BPD_INSTANCE_ID
LSW_TASK_EXECUTION_CONTEXT	TASK_ID
LSW_TASK_EXTACT_DATA	TASK_ID
LSW_DUR_MESSAGE_RECEIVED	MSG_ID

▶ Apply tuning properties to the target IBM BPM environment.

For migration from previous BPM, WebSphere Process Server, or WLE, you can apply the same tuning properties as used in the source system or tune the system using new values. The performance tuning parameters that you used in the previous version can be found in the `snapshot_folder/cell_name/Configurations/PerformanceTuningParameters.properties` file.

This file is created when you run the `BPMExtractSourceInformation` utility. The file contains the values of the following performance tuning parameters from the source environment:

– Java virtual machine (JVM)

– J2C connection factory

– J2C activation specification

– Data source

– Listener port

– Thread pool

– Service integration bus (SIB) messaging engine

– Web container

– Work manager information

This concludes our recommendations for proper tuning of performance considerations in BPM migration scenarios.

7

IT monitoring and tuning for performance testing and managing a production environment

In this chapter, we discuss how to use IT monitoring to obtain the relevant metrics to tune a production environment, and to conduct a performance or load test. Our objective in this chapter is to present a generalized methodology encompassing performance measurement, IT monitoring, and tuning that can be applied across a wide range of IBM BPM solutions.

Each of these topics is important to properly set up and maintain a well performing production environment.

► Performance and load testing
► IT monitoring and tuning
► Purging Business Process Manager data stores

7.1 Performance and load testing

In this section, we discuss key elements of planning for, executing, and interpreting the results of a performance or load test by addressing the following points:

► Create a performance evaluation checklist
► Establish clear performance objectives
► Plan the performance evaluation early in the development cycle
► Ensure performance results are representative

7.1.1 Create a performance evaluation checklist

Performance evaluations take many forms. The following two forms are the most common:

► A performance test, which entails pushing the system to the limit to determine the maximum load that can be achieved.

► A load test, which is typically done to ensure a given volume of work (for example, a predefined throughput rate or number of concurrent users) can be sustained on a specified topology while meeting a service level agreement (SLA).

Executing performance evaluations requires great care in planning the tests, and obtaining relevant results. Following is a checklist of key attributes to consider, each of which is detailed further in the remainder of this chapter:

► Establish clear performance objectives that are representative of business requirements.

► Ensure that the scenario measured accurately represents the one described in the objective.

► Perform initial tuning of the environment.

► Monitor systems on an on-going basis (for example, processor, memory, disk usage) and adjust tuning accordingly.

► When obtaining measurements, follow the following guidelines:

– Ensure the environment is warmed up before obtaining measurements.

– Preload system to the wanted state (particularly queues and databases).

– Ensure realistic think times are used between requests.

– Maintain a consistent flow of work into the system (for example, do not place 20 K input requests on a JMS queue and then start; implement flow control to pace the input requests).

- Maintain steady state in the databases: consistent number of active and completed tasks and instances.
- Restore the system to a consistent state (matching the preload state) before each measurement.
- Check for errors, exceptions, and time outs in logs and reports.

► For problem determination, see 7.2.2, "Information required to diagnose and resolve performance issues" on page 165.

7.1.2 Establish clear performance objectives

Performance objectives should be defined early in a project. These should be developed by interlocking with the business stakeholders to define the key scenarios, and then establishing SLAs for the required performance for each scenario. These SLAs should be clearly stated and measurable. An example of a well-defined response time focused SLA can be:

Refreshing the Task List in Process Portal should take < 3 seconds using Internet Explorer v10 with 100 milliseconds of network latency between the Process Portal and Process Server.

Another example of a well-defined throughput focused SLA can be:

The system will be able to sustain steady state throughput of 20 process instances completed per second while maintaining response times less than 2 seconds, measured at the 90th percentile.

> **Tip:** Use a *divide and conquer strategy* when evaluating performance objectives. For instance, single user response time metrics can be evaluated separately from overall system throughput metrics, using measurement tools customized for each. This allows simplification by eliminating traffic unnecessary for the task at hand. For example, in many cases, the push notifications driven by cometd can be removed from http recordings when building a workload driver to measure human workflow implemented in Business Processing Modeling Notation (BPMN).

7.1.3 Plan the performance evaluation early in the development cycle

After the performance objectives are defined, a performance plan should be written that describes the methodology that will be used to assess the objectives. One of the key elements of this plan will be to describe how to obtain

performance data that is representative of the expected production usage. Among the topics to include, consider:

► Define a workload that properly models the business scenarios of interest.

► Provision a topology that is as close to the production topology as possible.

► Define a database population (for example, user and group membership, number of process instances and tasks, both active and complete) that represent the expected steady state of the production environment.

► Choose a performance testing tool that can script the scenarios and apply the necessary load to mimic a production environment. IBM Rational® Performance Tester is one example.

► Design workloads that are reliable, produce repeatable results, are easy to run, and will complete in a reasonable amount of time.

7.1.4 Ensure performance results are representative

When running the performance test, ensure that the results obtained are representative of the expected production usage. To do this, follow these guidelines:

► Perform initial tuning of the environment, as is described in 4.1, "Tuning checklist" on page 64.

► Deploy IT monitoring tools in key components of the infrastructure, most notably the Process Server and its database. This is detailed in 7.2, "IT monitoring and tuning" on page 161.

► Plan to run the performance test multiple times, using the IT monitoring tools' output to guide further tuning. This is described further in 7.2.1, "Performance tuning methodology" on page 162.

– Since this is an iterative process, it is important that the performance test runs in a reasonable amount of time, and cleanup is done after each run.

► Use the following measurement methodology to obtain reliable, repeatable, representative performance data:

– Ensure that the environment is warmed up before obtaining measurements. This generally requires running several hundred iterations of the test to bring the system to steady state by ensuring that Just In Time compiling Java code, populating caches, and DB buffer pools, are all completed.

– Ensure realistic think times are used between requests. The think times should match the expected behavior of the users. One way to determine realistic think time is to determine the expected peak throughput rate, and how many concurrent users will be working during that time. With that

data, and the number of tasks that will be completed during peak periods, you can calculate the think time between tasks during the peak period as follows:

```
Think Time between tasks = instances completed * tasks per
instance / concurrent users
```

A simple example would be if the peak throughput rate is 600 completed instances per hour, with each instance consisting of 10 tasks, and 100 concurrent users working. The approximate think time between tasks for this example is 60 seconds (600 * 10 / 100).

– Maintain a consistent flow of work into the system. For example, do not place 20,000 input messages on a JMS queue and then start the test: this can place unrealistic pressure on system memory and threads, for example. Instead, implement flow control to pace the input messages to maintain a consistent amount of work available on the input queue during the measurement period.

– Maintain steady state in the Process Server database, such as a consistent number of active and completed tasks and instances.

– Clean up at the end of the run, particularly queues and the databases. To ensure that repeatable results are obtained, each measurement run should start with the system in a state consistent with the start of the previous measurement run and matching the preload state defined in your performance plan.

► Check for errors, exceptions, and time outs in logs and reports. These are typically indicative of functional problems or configuration issues, and can also skew performance results. Any sound performance measurement methodology insists upon fixing functional problems before evaluating performance metrics.

7.2 IT monitoring and tuning

In this section, we discuss how to use IT monitoring tools to guide performance tuning of the topology by addressing the following points.

► Performance tuning methodology
► Information required to diagnose and resolve performance issues
► IT monitoring tools
► Sample use cases

7.2.1 Performance tuning methodology

We suggest a system-wide approach to performance tuning the Business Process Manager environment. System performance tuning, which requires training and experience, is not exhaustively described here. Rather, we highlight key aspects of tuning that are important.

Tuning encompasses every element of the deployment topology:

► Physical hardware topology choices
► Operating system parameters
► Business Process Manager server, WebSphere Application Server, database, and messaging engine settings

The methodology for tuning can be stated as an iterative loop:

1. Pick a set of reasonable initial parameter settings and run the system.

2. Monitor the system to obtain metrics that indicate where performance is being limited.

3. Use monitoring data to guide further tuning changes.

4. Repeat until done.

These steps are described next.

Set reasonable initial parameter settings

Use the tuning checklist in 4.1, "Tuning checklist" on page 64 for a systematic way to set parameters.

For specific initial values, see Chapter 8, "Initial configuration settings" on page 179 for settings of the workloads that are used in IBM internal performance evaluation. You might consider these values for initial values.

Monitor the IT topology

Monitor the system components to determine system health and the need for further tuning as follows:

► For each physical machine in the topology, including front-end and back-end servers such as web and database servers, monitor the following processes by using the relevant OS tools (such as `vmstat`, `iostat`, `netstat`, or their operating system-specific equivalents):

 – Processor core use
 – Memory use
 – Disk use
 – Network use

- Complete the following actions for each Java virtual machine (hereafter called JVM) process started on a physical machine (for example, process server, messaging engine server, and other components):
  - Use tools such as `ps` or their equivalents to get core and memory usage per process.
  - Collect verbose garbage collection (`verbosegc`) statistics to obtain information about Java memory usage.

- For each Business Process Manager server or messaging engine JVM, use a tool such as IBM Tivoli Performance Viewer, or the WebSphere Performance Tuning Toolkit, to monitor these types of data:
  - Data connection pool use for each data source
  - Thread pool use for each thread pool (web container, default, and Work Managers)
  - JMS and IBM MQ queue depths, both internal to BPM (SCA, Event Manager, Failed Events, and so on), and external queues such as input queues via MQ or JMS bindings.

  You can download the WebSphere Performance Tuning Toolkit from this location:

  http://www.ibm.com/developerworks/websphere/downloads/peformtuning.html

- Monitor the database systems, which include the databases for the Process Center, Process Server, Performance Data Warehouse, Business Monitor, and any other applications that are part of the Business Process Manager solution. Business Process Manager solutions are often database-intensive, so ensuring excellent performance for the databases is critical. Use the database vendor's monitoring tools, along with operating system tools such as `iostat`, `vmstat`, or the equivalent.

IT monitoring tools

IT monitoring is an extensive topic and there are many tools available; a complete discussion of this topic is beyond the scope of this IBM Redbooks publication. Instead, the intent is to discuss the tools most commonly used for IT monitoring of Business Process Manager topologies.

The first step in establishing IT monitoring procedures is identifying the components of the system that are most critical. For essentially all Business Process Manager solutions, the Process Server, Process Center, and their databases are key components that need to be monitored, and tuned accordingly. Also, for many solutions the Messaging Engine, Support Cluster, HTTP Server, and external service providers are also key. Finally, client systems and their web browsers are also important for many cases, especially to resolve user response time issues.

For each physical system in the topology, processor, memory, and network utilization should typically be monitored. System tools are generally available for this, such as vmstat, iostat, netstat, ps, or their operating system-specific equivalents.

In addition, for database systems monitoring tools are typically provided by the database vendor. See the relevant documentation, or consult with your database administrator.

For WebSphere Application Server instances, the WebSphere Application Server administrative console (that is, Tivoli Performance Viewer) provides IT monitoring capabilities, including presenting data in graphs or tables, and providing summaries and tuning recommendations.

There are also a number of tools that use PMI instrumentation to monitor WebSphere Application Server instances, including the WebSphere Application Server Performance Tuning Toolkit, and third-party tools.

Finally, for client systems, in addition to the system tools noted above use the browser's tracing and debug tools to analyze both the browser's performance, and the network traffic between the client and the Process Server. Two examples are Firebug for Firefox browsers and Developer Tools for Chrome browsers.

You can find more information about these topics in the following references:

► WebSphere Application Server Performance Tuning Toolkit:

   http://www.ibm.com/developerworks/websphere/downloads/peformtuning.html

► WebSphere Application Server Performance Tuning Toolkit available for IBM Support Assistant 4.1:

   http://www.ibm.com/support/docview.wss?uid=swg21577660

► WebSphere Application Server PMI Documentation:

   http://www-01.ibm.com/support/knowledgecenter/SS7K4U_8.0.0/com.ibm.websphere.javadoc.doc/web/apidocs/com/ibm/websphere/pmi/stat/package-summary.html

Use monitoring data to guide further tuning changes

Correctly using monitoring data to determine how to tune the Business Process Manager requires skill and experience. In general, this phase of tuning requires the analyst to study the collected monitoring data, detect performance bottlenecks, and do further tuning. The key characteristic of this phase of tuning is that it is driven by the monitoring data that is collected in the previous phase.

Performance bottlenecks include, but are not limited to, these situations:

► Excessive use of physical resources, such as processor cores, disk, and memory. These issues can be resolved either by adding more physical resources or by rebalancing the load across the available resources.

► Excessive use of virtual resources. Examples of these resources include heap memory, connection pools, thread pools, and other resources. In this case, use tuning parameters to remove the bottlenecks.

When adjusting settings to improve performance, focus on the issues that matter the most and change settings that produce the highest payback. Let the monitoring data be your guide:

► Find the top few improvements that would give the most measurable improvement.

► Follow good experimental design: change one parameter at a time, measure its impact, and repeat.

See the next section for a discussion of approaches to resolving common performance issues.

7.2.2 Information required to diagnose and resolve performance issues

Several sources of information are highly valuable, even necessary, when diagnosing and resolving performance problems. This information is often referred to as *must-gather information*. It includes the following items:

► Client (Process Designer or Portal) processor utilization (for example, vmstat)

► Process Center or Process Server processor utilization

► Database server processor, disk subsystem (for example, iostat), and network utilization; also, for Oracle, the AWR Report

► The verbosegc logs (even if this problem is not a memory problem)

► SystemOut and SystemErr logs

► The contents of the configuration directory for the active profile of the server being studied, for example, the following location:

`profile_dir/config/cells/cellName/nodes/nodeName/servers/serverName`

This information is useful for obtaining a system view of performance, and often pinpoints the area that needs a more detailed examination. Additional diagnostic information for common performance bottlenecks is as follows:

► For a hang or unusually slow response time:

– A series of javacore files (3 - 5 dumps, separated by 30 seconds or a few minutes).

– Characterization of the network latency between key components of the system.

From client to server:

• Between Process Designer and Process Center
• Between Process Designer and Process Server
• From Portal to Process Server

From server to database:

• Between Process Center and the Process Center database
• Between Process Server and the Process Server database

► For memory failures (OutOfMemory):

– heapdump data (phd) from the time of the memory failure

– javacore from the time of the memory failure

► For a suspected memory leak:

– A series of heapdump files (PHD files) taken at different times before the out-of-memory exception.

– If the issue also presents itself as unusually slow response times, it might be more effective to analyze it that way (that is, gather the data listed under this category).

► For database performance issues:

– SQL traces

• Statement Monitor for DB2

  http://www.ibm.com/developerworks/data/library/techarticle/0303kolluru/0303kolluru.html

• SQL Trace for Oracle

  http://www.orafaq.com/wiki/SQL_Trace

7.2.3 IT monitoring tools

In IBM BPM, some of the key performance metrics, such as processor usage, memory usage, thread pool size, connection pool size, and disk I/O, can be

collected and analyzed using different tools. In this section, we list some of the tools that can be used to diagnose performance issues, and we explain how the tools can be used to collect the data and diagnose the issues in IBM BPM:

► nmon
► WebSphere Application Server Performance Tuning Toolkit
► IBM Monitoring and Diagnostic Tools for Java - Health Center
► IBM Monitoring and Diagnostic Tools for Java - Memory Analyzer
► Pattern Modeling and Analysis Tool for IBM Garbage Collector
► IBM Thread and Monitor Dump Analyzer for Java
► Database monitoring tools

nmon

The nmon[1] tool can be used to monitor system metrics on IBM AIX and Linux systems, which includes:

► Processor utilization
► Memory use
► Kernel statistics and run queue information
► Disk I/O rates, transfers, and read/write ratios
► Free space on file systems
► Disk adapters
► Network I/O rates, transfers, and read/write ratios
► Paging space and paging rates
► Processor and AIX specification
► Top processors
► IBM HTTP Web cache
► User-defined disk groups
► Machine details and resources
► Asynchronous I/O -- AIX only
► Workload Manager -- AIX only
► Network file system
► Dynamic LPAR changes -- only pSeries p5 and IBM OpenPower™ for either AIX or Linux

For more information about nmon, see the following IBM developerWorks article:

http://www.ibm.com/developerworks/aix/library/au-analyze_aix

WebSphere Application Server Performance Tuning Toolkit

WebSphere Application Server Performance Tuning Toolkit (PTT) is an Eclipse-based tool used to collect real-time performance metrics from the WebSphere Application Server used by IBM BPM.

---

[1] We suggest that you consider using utilities such as Task Manager and Perfmon for monitoring the Windows servers. If you are hosting IBM BPM on virtual servers, contact your operational team for recommendations on monitoring tools that can be used to monitor those servers.

Using Java Management Extensions (JMX) application programming interface, PTT captures data from basic Performance Monitoring Interface (PMI) of the WebSphere Application Server, and presents it in the form of charts and graphs. PTT requires the PMI to be enabled on WebSphere Application Server used by IBM BPM. Care must be taken when enabling PMI tracking points to only activate those that are needed because there is runtime overhead associated with each tracking point. For information about enabling PMI in IBM BPM, see the IBM Business Process Manager 8.5.5 IBM Knowledge Center at:

http://www.ibm.com/support/knowledgecenter/SSFPJS_8.5.5/com.ibm.wbpm.admin.doc/topics/tmon_prfstartadmin.html

Once configured, the PTT tool captures the following key performance metrics:

► IBM JVM processor utilization.

► JVM memory (heap size and used memory) utilization.

► EJB and servlet throughput and response times (the name of the EJB or servlet can normally be used to locate the SCA modules in the development tool).

► Discarded JDBC statement cache.

► Connection pool usage of JDBC data sources.

► Connection pool usage of JMS connection factory.

► Rolled back transactions.

► Timed out transactions.

► Used and available thread pool counts for WebSphere Application Server's thread pools such as web container, default, and ORB thread pools.

In addition to capturing the key performance metrics, PTT also shows the topology retrieved after connecting to the IBM BPM cell and a list of servers configured in the IBM BPM cell. For more information about PTT, see the following IBM developerWorks article:

http://www.ibm.com/developerworks/websphere/downloads/performtuning.html

IBM Monitoring and Diagnostic Tools for Java - Health Center

The IBM Monitoring and Diagnostic Tools for Java - Health Center (IBM Health Center) provides a no-cost low overhead diagnostic tool and API for monitoring an application running on a JVM. IBM Health Center gives the ability to assess the status of a running Java application.

IBM Health Center's continuous monitoring provides the following information to identify and help resolve problems with your application:

- Identify if JVM native memory or JVM heap memory is leaking. Once identified, memory leaks can be investigated further using IBM Monitoring and Diagnostic Tools for Java - Memory Analyzer in order to identify the source of the memory leak.

- IBM Health Center can be used with PTT and nmon to identify methods that lead to high CPU usage.

- Pin down I/O bottlenecks.

- Visualize garbage collection statistics. Concerns that are highlighted in garbage collection statistics can be investigated further using verbose garbage collection (verbose GC) traces in WebSphere Application Server. We suggest using Pattern Modeling and Analysis Tool (PMAT) for IBM Garbage Collector for analysis of the verbose GC trace files.

- View any lock contentions.

- Analyze unusual WebSphere real time events.

- Monitor application thread activity.

- Detect deadlock conditions in an application.

- Gather class histogram data.

In an IBM BPM production environment, we suggest configuring the IBM Health Center agent in *headless mode*. In headless mode, the IBM Health Center agent writes data to the local files on the JVM's system and the resulting `.hcd` files can be loaded to the Health Center client on a different system. This eliminates the need for changes in the firewall rules to establish connectivity between IBM Health Center client and IBM Health Center agent, assuming that the client and agent run on different systems. For more information about IBM Health Center, see the IBM Knowledge Center at:

http://www.ibm.com/support/knowledgecenter/SS3KLZ/com.ibm.java.diagnostics.healthcenter.doc/homepage/plugin-homepage-hc.html?lang=en

IBM Monitoring and Diagnostic Tools for Java - Memory Analyzer

IBM Monitoring and Diagnostic Tools for Java - Memory Analyzer (IBM Memory Analyzer) is a feature-rich Java heap analyzer that can help find memory leaks and reduce memory consumption. IBM Memory Analyzer provides both high-level understanding and analysis summaries using a number of standard reports. IBM Memory Analyzer allows users to carry out in-depth analysis through browsing and querying the Java objects present on the Java heap. Features that are offered by IBM Memory Analyzer include:

► Standard report that uses built-in capabilities to look for probable leak suspects.

► In-depth leak identification capabilities that look for collections with large numbers of entries, single large objects or groups of objects of the same class.

► Overview report that provides information about the Java heap usage, system property settings, threads present, and a class histogram of the memory usage.

► Top memory consumers report that gives a breakdown of the Java heap usage by largest objects and also which class loaders and classes are responsible for utilizing the most memory.

► Reports outlining the top memory consumers and providing information about potential memory inefficiencies in any selected component.

► Ability to browse the Java heap using a reference tree-based view. This makes it possible to understand the relationships between Java objects and to develop a greater understanding of the Java object interactions and the memory requirements of the application code.

For more information about IBM Memory Analyzer, see the IBM Support Assistant 5.0 IBM Knowledge Center at:

http://www.ibm.com/support/knowledgecenter/SS3KLZ/com.ibm.java.diagnostics.memory.analyzer.doc/homepage/plugin-homepage-ma.html?cp=SSLLVC_5.0.0%2F7&lang=en

Pattern Modeling and Analysis Tool for IBM Garbage Collector
Pattern Modeling and Analysis Tool for IBM Garbage Collector (PMAT) analyzes verbose GC traces by parsing the traces and building pattern models. PMAT recommends key configurations by executing a diagnostic engine and pattern modeling algorithm. If there are any errors that are related with Java heap exhaustion or fragmentation in the verbose GC trace, PMAT can diagnose the root cause of failures. PMAT provides rich chart features that graphically display Java heap usage. Features offered by PMAT include:

► Garbage collection analysis.
► Garbage collection table view.
► Allocation failure summary.
► Garbage collection usage summary.
► Garbage collection duration summary.
► Garbage collection graph view.
► Garbage collection pattern analysis.
► Zoom in or zoom out or selection or center of chart view.

For more information about PMAT, see the IBM Support Assistant 5.0 IBM Knowledge Center at:

http://www.ibm.com/support/knowledgecenter/SSLLVC_5.0.0/com.ibm.trove.t
ool.pmat.doc/docs/readme.html?cp=SSLLVC_5.0.0%2F8&lang=en

IBM Thread and Monitor Dump Analyzer for Java

IBM Thread and Monitor Dump Analyzer for Java (IBM Thread and Monitor Dump Analyzer) allows identification of hangs, deadlocks, resource contention, and bottlenecks in Java threads. Features that are offered by IBM Thread and Monitor Dump Analyzer include:

► Java heap information such as maximum heap size, initial heap size, garbage collection counter, free heap size, and allocated heap size.

► Ability to report on number of runnable threads, number of blocked and waiting threads, and total number of threads in the JVM.

► Ability to report on number of locked monitors.

► Ability to identify deadlocks between threads.

► Ability to compare multiple thread dumps (also called as javadumps or javacore files).

For more information about the IBM Thread and Monitor Dump Analyzer, see the IBM Support Assistant 5.0 IBM Knowledge Center at:

http://www.ibm.com/support/knowledgecenter/SSLLVC_5.0.0/com.ibm.esuppor
t.tool.tmda.doc/docs/readme.htm?cp=SSLLVC_5.0.0%2F9&lang=en

Database monitoring tools

We suggest using database monitoring tools such as IBM Data Studio web console to view the database health information, which can indicate the overall health of the databases used by IBM BPM, and provide details about specific database-related health indicators. For more information about IBM Data Studio web console, see the Data Studio IBM Knowledge Center at:

http://www.ibm.com/support/knowledgecenter/SS62YD_4.1.1/com.ibm.datatoo
ls.db.web.health.doc/topics/health_overview.html

7.2.4 Sample use cases

In this section, we discuss sample use cases, highlighting the usage of performance diagnostic tools in addressing some of the commonly seen performance issues in IBM BPM.

For the sake of this discussion, we assume that IBM BPM is installed on an AIX operating system:

► High processor consumption.

- – Use the nmon tool to identify the process that consumes high processor.
- – If the high processor consumption is attributed to an IBM BPM JVM process, configure the IBM Health Center tool on that JVM to identify the methods that lead to high processor usage in the JVM.
► High memory consumption.
- – Use the nmon tool to identify the process that consumes high memory.
- – If the high memory consumption is attributed to an IBM BPM JVM process, configure the IBM Health Center tool on that JVM to identify if the issue is related to high native memory consumption or high heap memory consumption. If the IBM Health Center tool points to high native memory usage, we suggest raising a problem management record with IBM Support.
- – If the issue is related to high heap memory usage:
  - • Enable verbose GC on this JVM and use the PMAT tool to identify any potential issues in the JVM configuration. JVM tuning is typically required to address JVM configuration issues. We suggest that you enable verbose GC in the production environment. There is no measurable overhead with letting verbose GC run in production. However, consider a file management strategy for output files generated by verbose GC because these files will typically grow over time, if left unmanaged.
  - • Use the IBM Memory Analyzer tool to spot any memory leaks and analyze application footprint for anomalies in heap memory usage.
► Exhaustion of WebSphere Application Server container resources such as thread pools, database connection pool, discard statement cache, and so on.

As an example, one of the common areas to monitor is the web container thread pool. Use the PTT tool to monitor the web container thread pool count. If the used thread pool count continues to rise over time, we suggest taking multiple thread dumps and analyzing and comparing the javacores in the IBM Thread and Monitor Dump Analyzer tool for hung threads, resource contentions, or deadlocks. Increase in web container's used thread pool count over time can also indicate increased workload during peak hours. Multiple thread dumps taken over time can also be used to see if active threads were reaching the maximum threads limit and whether the thread pool settings needed addressing to enable increased concurrency.

The PTT tool can be used to monitor other container resources, such as database connection pool, discard statement cache, and so on.

► Delayed response time.

Delayed response times in IBM BPM can be a symptom of several factors in the underlying system or application.

- Use the nmon tool to spot high CPU or memory consumption issues, I/O issues, and network issues.

- Use the PTT tool to monitor the IBM BPM system for alerts associated with thread pool count, used connection pool size, cache statement discard count, servlet and EJB response times, and transaction rollback count.

- Use the IBM Health Center tool to identify any resource locking issues.

- Use database monitoring tool to identify any issues associated with the health of databases used by IBM BPM.

- In IBM BPM Advanced, use the WebSphere Application Server Performance Monitoring Infrastructure tool to monitor the Service Integration Bus (hereafter called SIBus) queue depth. Because SIBus is used by IBM BPM Advanced, it is important to monitor the SIBus exception queues for error messages. Large accumulation of error messages on the exception queue can lead to performance degradation in IBM BPM Advanced. For instructions on enabling automatic monitoring of the WebSphere Application Server SIBus queue depth, see:

  http://www.ibm.com/support/docview.wss?uid=swg21618858

When IBM BPM is interfacing with external systems, monitor the network traffic between these systems for potential response time delays. We suggest that you consider using packet tracing tools (approved by your company) for monitoring the network traffic. We also suggest that you consult with the team monitoring the external systems themselves to see if there are any issues with those systems.

7.3 Purging Business Process Manager data stores

To improve overall system performance, it can be important to purge data from the BPM databases. For the following data stores, a purging strategy should be considered from a performance point of view:

► Repository data
► Process Instance Data (and the tasks associated with the instances)
► Durable subscription messages
► Document attachments
► Dynamic groups
► Shared business objects
► Case properties
► Performance Warehouse data
► Business Monitor data

7.3.1 Repository data

This section describes how to remove repository data on a Process Center and on a Process Server.

Process Center

Process Center holds snapshots of Process Applications (PAs) and Toolkits as they are developed. In addition to every named snapshot you create, every time a save operation is done in Process Designer, an unnamed snapshot is created. As a result, a significant number of unnamed snapshots can accumulate over time, which can impact the performance of many Process Center operations. To remove named and unnamed snapshots from the system use the `BPMSnapshotCleanup` command. This command is documented in the IBM Knowledge Center:

http://www.ibm.com/support/knowledgecenter/#!/SSFTDH_8.5.0/com.ibm.wbpm
.ref.doc/topics/rref_bpmsnapshotcleanup.html

Since IBM BPM version 8.5.0.1, unnamed snapshots can be deleted automatically using a configuration setting. Read more about this setting in the IBM Knowledge Center at:

http://www.ibm.com/support/knowledgecenter/#!/SSFTDH_8.5.0/com.ibm.wbpm
.admin.doc/managinglib/topic/managing_snapshots_j.html

When a PA has content from Integration Designer, directly or indirectly via a toolkit, a WebSphere Business Level Application (BLA) is created, and for each SCA module in the PA, an enterprise archive (EAR) is generated and put into the BLA. A BLA will be created:

► For the Tip
► For every named snapshot of both toolkits and PAs containing advanced content

These can accumulate. It is recommended to have a regular practice of deleting these BLAs and their EARs, for those not needed. Typically, you can delete all except those of the Tip.

> **Important:** These BLAs will be re-created on demand as needed, for example during a playback.

The snapshot Undeploy and Archive commands in the Process Center console remove these BLAs and EARs.

Also, after deleting a named and archived snapshot, IBM Business Monitor's monitor models will not be deleted when the named and archived snaphost is deleted. You need to manually delete these monitor models using the administrative console.

Process Server

Process Server holds named snapshots of process applications that have been deployed to it. To remove named snapshots from the Process Server use the `BPMDeleteSnapshot` command, which is documented in the IBM Knowledge Center:

http://www.ibm.com/support/knowledgecenter/#!/SSFTDH_8.5.0/com.ibm.wbpm
.ref.doc/topics/rref_deletesnapshot.html

7.3.2 Process Instance Data

There are two types of processes to consider:

► BPM Standard holds instances of BPD processes and tasks
► BPM Advanced also holds instances of Business Process Execution Language (BPEL) processes and tasks

> **Important:** The cleanup procedures touch many records in multiple database tables. Run these cleanup procedures in off shift hours to minimize negative impact on overall system performance and to avoid locking issues.

BPMN: Delete completed process and task instances

If completed process and task instances are not removed, they accumulate over time, which can impact overall system performance, especially for task list queries like saved searches.

There are several options for deleting process artifacts after completion:

► Delete *system tasks*.

► Delete task context of *system tasks* and *user tasks*.

► Delete *process instances* including all related tasks.

If you select the **Delete task on completion** check box in the **Implementation** tab of a system task, the task will be deleted after completion. This is the default choice in BPM 8.5.5.

For both user tasks and system tasks, selecting the **Clean state** check box in the **Implementation** tab of a task will delete the task context as the task completes.

The task context includes for example, the variables of task. Task contexts can grow quite large and consume considerable space on disk.

After a process instance is completed and no longer needed, use the wsadmin command `BPMProcessInstanceCleanup` to delete business process instances, refer to the following technote for more details:

http://www.ibm.com/support/docview.wss?uid=swg21439859

BPEL: delete process instance and task instance data

There are several objects that need to be cleaned up:

► Process and task instances
► Audit log entries
► Shared work items
► People queries

Refer to "Cleanup procedures for Business Process Choreographer" in the IBM Business Process Manager online IBM Knowledge Center:

http://www.ibm.com/support/knowledgecenter/SSFPJS_8.5.0/com.ibm.wbpm.bpc.doc/topics/c2_cleanup.html

You can also model business processes and human tasks with automatic deletion. You can choose the options *delete on completion*, *delete on successful completion*, or *none*. If you have chosen to not delete business processes or human tasks automatically, use one of the preceding cleanup procedures.

7.3.3 Durable subscription messages

Durable subscription messages exist and persist in the database if durable subscriptions are enabled for intermediate message events or intermediate content events. To delete durable subscription messages, you can use the `BPMDeleteDurableMessages` wsadmin command. For more information, refer to:

http://www.ibm.com/support/knowledgecenter/SSFPJS_8.5.0/com.ibm.wbpm.ref.doc/topics/rref_bpmdeletedurablemessages.html

7.3.4 Document attachments

Purging of process instances removes its associated document attachments. The JavaScript method `deleteAllVersions()` can be also used to delete attachments.

7.3.5 Dynamic groups

Purging of process instances removes associated dynamic groups.

7.3.6 Shared business objects

Shared business objects have their own lifecycle and hence are not deleted when the process instance ends.

7.3.7 Case properties

Unused case properties can accumulate in the database. Unused case properties are case properties that are currently specified in the case and document classes but are no longer used because the associated case application snapshot has been deleted.

To delete unused case properties, use the `cleanupDocumentStoreProperties` command as described in the IBM Knowledge Center:

http://www.ibm.com/support/knowledgecenter/SSFPJS_8.5.5/com.ibm.wbpm.ref.doc/topics/rref_cleanupdocumentstoreproperties.html

7.3.8 Performance Warehouse data

Use the `prune` command to delete records from the *performance data warehouse* database. For more information about how to apply this command, refer to the IBM Knowledge Center at:

http://www.ibm.com/support/knowledgecenter/SSFPJS_8.5.0/com.ibm.wbpm.admin.doc/adminguide/topic/using_perfsvr_commands_prune.html

7.3.9 Business Monitor data

To increase dashboard efficiency and performance, purge instance data from the IBM Business Monitor database and archive it. IBM Business Monitor has a scheduled service to purge and archive instances older than specified value. To purge and archive instances, follow the IBM Knowledge Center at:

http://www.ibm.com/support/knowledgecenter/#!/SSFPJS_8.0.1/com.ibm.wbpm.mon.admin.doc/admin/purge_archive.html

8

Initial configuration settings

This chapter suggests initial settings for several relevant parameters. These values might not be optimal in all cases, but the values work well in internal IBM performance evaluations. They are, at a minimum, useful starting points for many proof-of-concepts and customer deployments. As described in 4.1, "Tuning checklist" on page 64, tuning is an iterative process. Follow that procedure and adjust these values as appropriate for your environment.

8.1 Business Process Manager server settings

This section provides settings based on IBM internal performance evaluations of the Business Process Manager V8.5.0 and V8.5.5 (referred to as 8.5.x below) servers. These settings were derived by using the tuning methodology and guidelines described in Chapter 4, "Performance tuning and configuration" on page 63. Consider these settings useful starting points for your use of this product. For settings that are not listed, use the default settings that are supplied by the product installer as a starting point, and then follow the tuning methodology specified in 4.1, "Tuning checklist" on page 64.

Three settings are described in this section:

► A three-tiered configuration for Business Process Execution Language (BPEL) business processes, with the production database on a separate server.

► A three-tiered configuration for Business Processing Modeling Notation (BPMN) business processes, with the production database on a separate server.

► A two-tiered (client/server) configuration for BPEL business processes, with the production database collocated on the server.

8.1.1 Three-tiered: BPEL business processes with web services and remote DB2

Through the WebSphere Application Server, we used a three-tiered configuration in our internal performance work to evaluate the performance of a BPEL business process that models automobile insurance claims processing. This configuration is an example of many production environments where DB2 is on a separate system than the Business Process Manager server. The web services binding was used for communications. The business process has two modes of operation:

► A BPEL microflow (straight-through process) that processes claims where no human intervention is required

► A BPEL microflow plus macroflow (long-running process) pattern, where the macroflow is started when a review or approval is required (for example, if the claim amount is above a certain limit)

Three systems were used in this configuration:

► Request driver
► Business Process Manager V8.5.x server
► DB2 database server

Extensive tuning was applied to the Business Process Manager server and the DB2 database server in order to ensure that the system achieved maximum throughput. Some tuning varied because of the operating system (such as AIX and Windows) and the number of processor cores. These variations are presented in tabular format after the description of common tuning.

For all topologies in this section, we suggest you complete the following actions to tune Business Process Manager and DB2 and maximize throughput:

► Use the production template.

► Define the Common database as local DB2 type 4.

► Establish BPEL Business Process support with `bpeconfig.jacl`. Click **Data sources** → **BPEDataSourceDb2** → **WebSphere Application Server data source properties statement cache** to 300.

► Disable PMI.

► Set `HTTP maxPersistentRequests` to -1.

Set GC policy to `-Xgcpolicy:gencon` (see Table 8-1 and Table 8-2 on page 182 for nursery setting `-Xmn`).

► Use remote DB2 databases (connection type 4) for the SIB System and SIB Business Process Choreographer database (BPEDB).

Table 8-1 lists Business Process Manager server-related settings to modify from their default value when the Business Process Manager server is deployed on AIX.

Table 8-1   Three-tiered application cluster settings for AIX

Setting	Value
Java heap Megabytes	3072
Java nursery Megabytes -Xmn	1024
Default thread pool max	100
BPEDB Data source → connection pool max	300
BPEDB Data source → WebSphere Application Server data source properties → Statement cache size	300
BPC messaging engine data source → connection pool max	50
SCA SYSTEM messaging engine data source → connection pool max	50
BPM Common Data source → connection pool max	500
J2C activation specifications → "Your_AS" → Custom properties → maxConcurrency, maxBatchSize	50

Setting	Value
Resources → Asynchronous Beans → Work Managers → BPENavigationWorkManager → Work request queue size, max threads, growable	400, 50, no
Application Cluster → Business Flow Manager → Message pool size, Intertransaction cache size	5000, 400
WebContainer thread pool min, max	100, 100
com.ibm.websphere.webservices.http.maxConnection	50

Table 8-2 lists Business Process Manager server-related settings to modify from their default value when Business Process Manager server is deployed on Windows and Linux on Intel systems.

Table 8-2   Three-tiered web service and remote DB2 tuning variations

Tuning - 4 core Process Server	Microflow	Macroflow
Java heap Megabytes	3072	3072
Java nursery Megabytes -Xmn	1024	1024
Web container thread pool max	150	300
Default thread pool max	200	200
BPE database connection pool max	250	350
BPC messaging engine database connection pool max	30	150
SYSTEM messaging engine database connection pool max	40	100
Common database connection pool max	80	100
J2C activation specifications → "Your AS" → Custom properties → maxConcurrency	40	160
BPEInternalActivationSpec batch size		10
"Your AS" batch size		32
Java custom property com.ibm.websphere.webservices.http.maxConnection	200	200
Application servers → server1 → Business Flow Manager → allowPerformanceOptimizations		Yes

Tuning - 4 core Process Server	Microflow	Macroflow
Application servers → server1 → Business Flow Manager → interTransactionCache.size		400
Application servers → server1 → Business Flow Manager → workManagerNavigation.messagePoolSize		4000
Resources → Asynchronous Beans → Work Managers → BPENavigationWorkManager → min threads, max threads, request queue size		30, 30, 30

The DB2 database server has several databases that are defined for use by the Business Process Manager server. Spread the database logs and table spaces across a Redundant Array of Independent Disks (RAID) to distribute disk use. Tune the `SCA.SYSTEM.<cellname>.BUS` database and the BPEDB as follows:

▶ `db2 update db cfg for sysdb using logbufsz 512 logfilsiz 8000 logprimary 20 logsecond 20 auto_runstats off`

▶ `db2 alter bufferpool ibmdefaultbp size 30000`

Create and tune the BPE database by using the following DB2 commands and generated scripts:

▶ `db2 CREATE DATABASE bpedb ON /raid USING CODESET UTF-8 TERRITORY en-us`

▶ `db2 update db cfg for bpedb using logbufsz 512 logfilsiz 10000 logprimary 20 logsecond 10 auto_runstats off`

▶ `db2 -tf createTablespace.sql` (Business Process Manager V8.5.x server generated script)

▶ `db2 -tf createSchema.sql` (Business Process Manager V8.5.x server generated script)

▶ `db2 alter bufferpool ibmdefaultbp size 132000`

▶ `db2 alter bufferpool bpebp8k size 132000`

Ensure that database statistics are current.

Use your database vendor's tool to obtain recommendations for indexes to create.

8.1.2 Three-tiered: Human Services with BPMN business processes

We used a three-tiered configuration in our internal performance work to evaluate the performance of a BPMN business process that models automobile insurance claims processing. Human Services were used to process the claims with a Call Center scenario of Query Tasks, Claim Task, Complete Task, and Commit Task.

This configuration is an example of many production environments where DB2 is on a separate system than the Business Process Manager server. The web services open SCA binding was used for communications. Three systems were used in this configuration:

► Request driver
► Business Process Manager V8.5.x Process Server
► DB2 database server

Business Process Manager Process Server in three-tiered configuration

Use the following settings for the Business Process Manager V8.5.x Process Server in a three-tiered configuration:

► Use a 64-bit Java virtual machine (JVM) and alter the Java memory management settings by adding Java command-line parameters:

   `-Xgencon, -Xms1800M, -Xmx1800M, -Xmn800M`

► Increase size of the WebContainer ThreadPool to a minimum of 200, maximum of 400.

► Increase the maximum size of the Default Thread Pool to 40.

► Disable logging for selected Business Process Manager Process Server components (for example, web services).

► Set bpd-queue-capacity to 10 times the number of physical processor cores, capped at 80.

► Set max-thread-pool-size to 30 plus 10 times the number of physical processor cores, capped at 110.

DB2 database server in three-tiered configuration

To use the DB2 database server in a three-tiered configuration, the following settings are required:

► Separate log files and containers on to separate (RAID) disks

► Increase log file size for the Process Server database to 16,384 pages

► Enable file system caching for the Process Server database:

    db2 alter tablespace userspace1 file system caching

► Exclude the table SIBOWNER from automatic runstats execution, as described in the following technote:

    http://www.ibm.com/support/docview.wss?uid=swg21452323

► Ensure that database statistics are current.

► Use your database vendor's tool to obtain recommendations for indexes to create.

8.2 Mediation Flow Component settings

This section describes settings used for selected internal performance evaluations of Mediation Flow Components. These settings were derived by using the tuning methodology and guidelines described in Chapter 4, "Performance tuning and configuration" on page 63. Consider these settings starting points for using Mediation Flow Components. For settings that are not listed, use the default settings that are supplied by the product installer as a starting point. See 4.1, "Tuning checklist" on page 64 for a description of tuning methodology.

8.2.1 Mediation Flow Component common settings

Mediation Flow Component settings are good starting points for tuning a WebSphere ESB solution, regardless of binding choices:

► Tracing is disabled.

► If security is required, application security instead of Java2 security is used to reduce processor usage.

8.2.2 Mediation Flow Component settings for web services

The Mediation Flow Component settings for web services are as follows:

► PMI monitoring is disabled

- ▶ Value of WebContainer thread pool sizes is set to max 50 and min 10
- ▶ Value of WebContainer thread pool inactivity timeouts for thread pools is set to 3500

8.2.3 Mediation Flow Component settings for Java Message Service

The Mediation Flow Component settings for WebSphere MQ and JMS are as follows:

- ▶ Activation specification

  Set the maximum concurrent endpoints to 50.

- ▶ Queue Connection factory

  Set the maximum connection pool size to 51.

- ▶ DiscardableDataBufferSize

  Set the size to 10 MB and set CachedDataBufferSize to 40 MB.

8.2.4 DB2 settings for Java Message Service persistent

Set the following values. They are relevant only for Java Message Service (JMS) persistent configurations because they use the database to persist messages:

- ▶ Place database table spaces and logs on a fast disk subsystem.
- ▶ Place logs on separate device from table spaces.
- ▶ Set buffer pool size correctly.
- ▶ Set the connection min and max to 30.
- ▶ Set the statement cache size to 40.
- ▶ Set up a raw partition for DB2 logs.

Otherwise, unless noted in the workload description, use the default settings as supplied by the product installer.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering this publication, see "How to get Redbooks" on page 188. The document referenced here might be available in softcopy only.

► *IBM Business Process Manager V8.0. Production Topologies*, SG24-8135

► *IBM Operational Decision Management (WODM) V8.0 Performance Tuning Guide*, REDP-4899

► *Leveraging the IBM BPM Coach Framework in your organization*, SG24-8210

► *Business Process Management Deployment Guide using Business Process Manager 8.5*, SG24-8175

Online resources

These websites are also relevant as further information sources:

► WebSphere Application Server Performance

   http://www.ibm.com/software/webservers/appserv/was/performance.html

► WebSphere Application Server Knowledge Center (including Tuning Guide)

   http://www-306.ibm.com/software/webservers/appserv/was/library/?S_CMP=rnav

► DB2 Version 10.5 Knowledge Center

   http://www.ibm.com/support/knowledgecenter/SSEPGG_10.5.0/com.ibm.db2.luw.kc.doc/welcome.html

► Diagnostics Guide for IBM SDK and Runtime Environment Java Technology Edition, Version 6

   http://publib.boulder.ibm.com/infocenter/javasdk/v6r0/index.jsp

► IBM Pattern Modeling and Analysis Tool for Java Garbage Collector

   http://www.alphaworks.ibm.com/tech/pmat

- Oracle 11g Documentation Library

  http://docs.oracle.com/cd/B28359_01/index.htm

- IBM Business Process Manager v8.5 Knowledge Center

  http://www.ibm.com/support/knowledgecenter/#!/SSFPJS_8.5.0/com.ibm.w
  bpm.main.doc/ic-homepage-bpm.html

- Performance tuning resources for WebSphere Process Server and IBM Business Process Manager solutions

  http://www.ibm.com/developerworks/websphere/library/techarticles/111
  1_herrmann/1111_herrmann.html

- Best practices for DB2 for Linux, UNIX, and Windows

  http://www.ibm.com/developerworks/data/bestpractices/db2luw

- Oracle 10g Release 2 Documentation Library (including Performance Tuning Guide)

  http://www.oracle.com/pls/db102/homepage

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, technotes, draft publications and Additional materials, and order hardcopy Redbooks publications, at this website:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

IBM

Redbooks

IBM Business Process Manager V8.5 Performance Tuning and Best Practices

(0.2"spine)
0.17"<->0.473"
90<->249 pages

IBM Business Process Manager V8.5 Performance Tuning and Best Practices

IBM®

Redbooks®

Learn valuable tips for tuning

Get the latest recommendations

See example settings

This IBM Redbooks publication provides performance tuning tips and best practices for IBM Business Process Manager (IBM BPM) V8.5.5 (all editions) and IBM Business Monitor V8.5.5. These products represent an integrated development and runtime environment based on a key set of service-oriented architecture (SOA) and business process management (BPM) technologies. Such technologies include Service Component Architecture (SCA), Service Data Object (SDO), Business Process Execution Language (BPEL) for web services, and Business Processing Modeling Notation (BPMN).

This book targets a wide variety of groups, both within IBM (development, services, technical sales, and others) and customers. For customers who are either considering or are in the early stages of implementing a solution incorporating Business Process Manager and Business Monitor, this document proves a useful reference. The book is useful both in terms of best practices during application development and deployment and as a reference for setup, tuning, and configuration information.

This book talks about many issues that can influence performance of each product and can serve as a guide for making rational first choices in terms of configuration and performance settings. Similarly, customers who already implemented a solution with these products can use the information presented here to gain insight into how their overall integrated solution performance can be improved.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information: ibm.com/redbooks