IBM® WebSphere®

IBM Business Process Manager V7.5 パフォーマンス・チューニングおよびベスト・プラクティス

チューニングに役立つヒントを学ぶ

最新のベスト・プラクティスを入手する

設定例を確認する

IBM Business Process Management
パフォーマンス・チーム

Redpaper

ibm.com/jp/support/redbooks

International Technical Support Organization

IBM Business Process Manager V7.5 パフォーマンス・チューニングおよびベスト・プラクティス

2012 年 4 月

SG88-4065-00
(英文原典 : REDP-4784-00)

注 : 本書および本書で紹介する製品をご使用になる前に、vii ページの『特記事項』に記載されている情報をお読みください。

第 1 版（2012 年 4 月）

本書は、IBM Business Process Manager V7.5、IBM Integration Designer V7.5、および IBM Business Monitor V7.5 に適用されます。

本書は 2012 年 6 月 13 日に作成または改訂されました。

目次

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品およびプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権（特許出願中のものを含む）を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒 103-8510
東京都中央区日本橋箱崎町 19 番 21 号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾 :

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

商標

IBM、IBM ロゴおよび ibm.com は、International Business Machines Corporation の米国およびその他の国における商標または登録商標です。これらおよび他の IBM 商標に、この情報の最初に現れる個所で商標表示（または）が付されている場合、これらの表示は、この情報が公開された時点で、米国において、IBM が所有する登録商標またはコモン・ロー上の商標であることを示しています。このような商標は、その他の国においても登録商標またはコモン・ロー上の商標である可能性があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

以下は、International Business Machines Corporation の米国およびその他の国における商標です。

AIX®
alphaWorks®
CICS®
Cognos®
DB2®
developerWorks®
IBM®
IMS™
MQSeries®
POWER6®
Redbooks®
Redpaper™
Redbooks (logo) ®
Tivoli®
WebSphere®
z/OS®

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

インテル、Intel ロゴ、Intel Inside ロゴ、Intel Centrino ロゴは、Intel Corporation または子会社の米国およびその他の国における商標または登録商標です。

Windows、および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。

UNIX は The Open Group の米国およびその他の国における登録商標です。

前書き

この IBM® Redpaper™ 資料には、IBM Business Process Manager（BPM）V7.5（全エディション）および IBM Business Monitor V7.5 のパフォーマンス・チューニングのヒントとベスト・プラクティスが記載されています。これらの製品は、主要な一連のサービス指向アーキテクチャー（SOA）およびビジネス・プロセス・マネジメントのテクノロジーに基づいた開発およびランタイムの統合環境を表します。このようなテクノロジーとして、Service Component Architecture（SCA）、Service Data Object（SDO）、Web サービス用 Business Process Execution Language（BPEL）、および Business Processing Modeling Notation（BPMN）があります。

BPM および Business Monitor は双方とも IBM WebSphere® Application Server インフラストラクチャーのコア機能が基盤となっています。このため、BPM ソリューションでは、WebSphere Application Server およびそれに対応するプラットフォーム Java 仮想マシン（JVM）に関するチューニング、構成、およびベスト・プラクティスの情報が役に立ちます。

本書は、IBM 内（開発、サービス、テクニカル・セールスなど）と顧客の両方にわたる、さまざまなグループを対象としています。BPM および Business Monitor を組み込むソリューションの実装を検討している顧客、またはそのような実装の初期段階にある顧客にとって、本書は必ず有益な参照資料になります。本書は、アプリケーションの開発およびデプロイメントの際のベスト・プラクティスとして、またセットアップ、チューニング、および構成に関する情報の参照資料として役に立ちます。

本書は、各製品のパフォーマンスに影響を及ぼす問題の多くを紹介しており、構成とパフォーマンスの設定において合理的な第 1 選択を行うためのガイドとして役立ちます。同様に、これらの製品を使用するソリューションを既に実装している顧客は、本書の情報を使用して、統合ソリューションの全体的なパフォーマンスを向上させる方法について見識を得ることができます。

この資料の作成チーム

本書は、BPM パフォーマンス・チーム、およびテキサス州のオースティン、ドイツのベーブリンゲン、および英国のハースリーのメンバーによって作成されました。

- Mike Collins
- Weiming Gu
- Paul Harris
- Ben Hoflich
- Kean Kuiper
- Sam Massey
- Rachel Norris
- David Ogren
- David Parish
- Chris Richardson
- Randall Theobald

あなたも出版物の著者になれます

スキルにスポットライトを当て、キャリアを発展させ、出版物の著作者になるというすべてを同時にかなえるチャンスです。ITSO 研修プロジェクトに参加し、専門分野についての本の執筆に携わる一方で、最先端の技術を使用するという貴重な体験を得ることができます。このような取り組みは、参加者の技術に関する交流や関係のネットワークを広げるとともに、製品受容と顧客満足度の向上に役立ちます。研修は、2 週間から 6 週間の長さで実施され、実地に参加することも自身の本拠地からリモート研修生として参加することもできます。

研修プログラムの詳細の入手、研修カタログの閲覧、およびオンライン申請については、以下にアクセスしてください。

ibm.com/redbooks/residencies.html

ご意見をお寄せください

皆さまのご意見は重要です。

弊社の資料をできる限り有益なものにしていきたいと願っています。この資料またはその他の IBM Redbooks 出版物に関するご意見は、以下のいずれかの方法でお送りください。

- 以下のアドレスから「**お問い合わせ**」の Redbooks レビュー用オンライン入力フォームを使用

  ibm.com/jp/support/redbooks

- E メールで送信

  redbooks@us.ibm.com

- 郵送

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

IBM Redbooks のその他の関連アドレス

- Facebook:

  http://www.facebook.com/pages/IBM-Redbooks/178023492563?ref=ts

- Twitter:

  http://twitter.com/ibmredbooks

- LinkedIn:

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

- IBM Redbooks 週刊ニュースレターで、Redbooks の新しい資料、研修、およびワークショップに関する情報を入手できます。

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

- Redbooks の最新資料に関する情報は RSS 配信を使用して入手できます。

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

1

概要

本章では、本書で取り上げる製品と全体の文書構造を示します。

1.1 この資料で取り上げる製品

本書では、以下の製品について説明します。

- BPM V7.5（全エディション）

  BPM は、簡潔性、使いやすさ、およびタスク管理機能を組み合わせて、総合的な SOA の一環として、エンタープライズの統合およびトランザクション・プロセス管理の要件に対応します。

  BPM では、以下のように価値が付加されます。

  - ビジネス・プロセスを最適化します。これは、分散プラットフォームおよび IBM z/OS® にわたって、すべてのプロセス参加者に可視性をもたらし、コラボレーションを助長し、継続的なプロセス強化を可能にすることによって実現されます。
  - タスクの実行、作業項目の管理、パフォーマンスのトラッキング、およびイベントへの対応について、すべてリアルタイムでの統合ビューによって効率性を向上します。
  - ユーザーがリアルタイムの分析によってビジネス・プロセスを最適化できるようにします。
  - ユーザーが簡単にプロセスのステップを実行していけるようにガイドするプロセス・コーチングなどの、ビジネス・ユーザーに焦点を当てた設計機能によって、タイム・ツー・バリューを短縮します。
  - 同じプロセス・バージョンを全員が確認できるようにする統合されたモデル駆動型環境によって、確信を持って変更を管理します。
  - IBM WebSphere Lombardi Edition の簡易性、使いやすさ、およびタスク管理機能を、IBM WebSphere Process Server（Advanced Edition のみ）の主要機能と組み合わせます。これらの機能は、エンタープライズの統合およびトランザクション・プロセス管理の要件を総合的な SOA の一環としてサポートします。
  - 最新の WebSphere Process Server（Advanced Edition のみ）および WebSphere Lombardi Edition に対して、以前のバージョンとの互換性を提供します。

- Business Monitor V7.5

  Business Monitor は、包括的なビジネス・アクティビティー・モニタリング（BAM）を提供します。この機能を使用すると、ユーザーはリアルタイムでエンドツーエンドのビジネス処理、トランザクション、およびプロセスを確認できるようになり、プロセスの最適化と効率性の向上に役立ちます。

  Business Monitor では、以下のように価値が付加されます。

  - BPM テクノロジーを使用して実装されているかどうかにかかわらず、異なる環境で実行されるプロセスおよびアプリケーションに、ハイパフォーマンスの BAM ソリューションを提供します。
  - BPM 用の統合 BAM に組み込みツールおよびランタイム・サポートを提供します。
  - 履歴データの拡張分析およびレポート用に、IBM Cognos® Business Intelligence Server V10.1.1 を統合します。
  - Business Process Modeling Notation V2.0（BPMN2）プロセス用に、ダッシュボードを自動生成します。この機能によって、プロセス・インスタンス、重要業績評価指標（KPI）、Cognos レポート、およびアノテーション付き BPMN2 プロセス・ダイアグラムでリアルタイムの可視性を実現できます。
  - 詳細に設定されたセキュリティーを組み込んで、ユーザーによるさまざまな情報深度または詳細へのアクセスを有効にしたり、防止したりします。
  - データのフィルタリングおよびダッシュボードのコントロールとレポートのビジネス・ユーザーによる拡張カスタマイズを可能にします。
  - Web インターフェース、iPad、モバイル・デバイス、および企業ポータルを使用した KPI のビュー、メトリック、およびアラートを有効にします。
  - 分散プラットフォームおよび z/OS で使用できます。

1.2 資料の構造

以下のリストは、本書の各章の概要を示しています。

- 1 ページの第 1 章、「概要」

  この章であり、本書の内容の範囲を示しています。

- 5 ページの第 2 章、「アーキテクチャーのベスト・プラクティス」

  ハイパフォーマンスでスケーラブルなソリューションを生み出すためのアーキテクチャーおよびトポロジーに関する決断事項について、さまざまな提案を提供します。

- 23 ページの第 3 章、「開発のベスト・プラクティス」

  ハイパフォーマンスのシステムに導くために、ソリューション開発者に向けたガイドラインを示します。

- 47 ページの第 4 章、「パフォーマンスのチューニングと構成」

  ビジネス・プロセス・マネジメント・ソリューションを構成する主要ソフトウェア・コンポーネントの構成パラメーターおよび設定について説明します。

- 93 ページの第 5 章、「初期構成設定」

  BPM および Business Monitor を使用する IBM パフォーマンス・チームが実行する代表的な処理で使用されるソフトウェア構成の詳細を示します。

- 101 ページの『関連資料』

  この資料で説明している製品、および WebSphere Application Server、IBM DB2® 、およびその他のアプリケーションなどの関連製品に関するベスト・プラクティス、パフォーマンス情報、および製品情報へのリンクを提供します。

2

アーキテクチャーのベスト・プラクティス

本章では、ハイパフォーマンスでスケーラブルな Business Process Manager（BPM）ソリューションを設計するためのガイダンスを示します。本章では、特に、本書で取り上げる BPM および Business Monitor 製品によって実現されるテクノロジーと機能に関連するベスト・プラクティスに焦点を置きます。ただし、これらの製品は、既存のテクノロジー（WebSphere Application Server や DB2 など）が基になっています。これらのテクノロジーにはそれぞれ固有の関連ベスト・プラクティスがあります。

本書では、これらの BPM 外部のベスト・プラクティスについては列挙していません。これらのその他のテクノロジーに関する参照情報およびリンクについては、101 ページの『関連資料』を参照してください。

2.1 最善のチューニングおよびデプロイメントのガイドライン

本章では、BPM V7.5 ソリューションのアーキテクチャー上のベスト・プラクティスの詳細を示します。開発のベスト・プラクティスおよびパフォーマンスのチューニングと構成については、以降の章で取り上げています。

本書の他の箇所を読まない場合は、以下のチューニングおよびデプロイメントの主要ガイドラインを読んでこれに従ってください。これらのガイドラインは、パフォーマンスを要求される事実上すべての顧客との関わりに関連しています。

- ハイパフォーマンスのディスク・サブシステムを使用します。事実上いずれの実際的なトポロジーでも、適切なパフォーマンスを実現するには、メッセージおよびデータのストアをホストする層に、サーバー・クラスのディスク・サブシステム（複数の物理ディスクを伴った RAID アダプターなど）が必要です。この点を誇張してもし過ぎることはありません。多くの場合、適切なディスク・サブシステムを使用すると、いくつかの要因によってソリューションの全体的なパフォーマンスが向上します。
- 64 ビットの Java 仮想マシン（JVM）を使用し、最適なスループットと応答時間を実現できるように適切な Java ヒープ・サイズを設定します。JVM の verbose ガーベッジ・コレクション・コマンド（**verbosegc**）を使用して取得するメモリー使用量データは、最適な設定の決定に役立ちます。詳しくは、52 ページの 4.3.2、「Java のチューニング・パラメーター」を参照してください。
- ハイパフォーマンスであり、ハイレベルのスループットおよび並行性を処理するよう設計されている業務仕様のデータベースである DB2 など、実動レベルの高品質のデータベースを使用します。DB2 では、最適なスケーリングおよび応答時間を実現できます。
- 最適なパフォーマンスを実現できるようデータベースをチューニングします。データベースについて適切なチューニングおよびデプロイメントを選択することによって、システムの全体的なスループットを大きく向上できます。詳しくは、77 ページの 4.10、「一般的なデータベース・チューニング」を参照してください。
- トレースを無効にします。トレースはデバッグ時には重要ですが、トレースに必要なリソースによってパフォーマンスが大きな影響を受けます。詳しくは、55 ページの 4.4.1、「トレーシングとモニタリングの考慮事項」を参照してください。
- スレッド・プールおよび接続プールを十分な並行性を確保できるように構成します。この構成は、高ボリュームで並行性の高いワークロードの場合に重要です。スレッド・プールの設定が、サーバーで同時に処理できる作業量に直接影響を与えるためです。詳しくは、58 ページの『スレッド・プール・サイズの構成』を参照してください。
- IBM Process Designer と IBM Process Center の間に高速のネットワーク接続を使用します。Process Designer は、Process Center と頻繁に通信するため、ネットワークの待ち時間を最小限に抑えることが重要です。
- Web サービス用 Business Process Execution Language（BPEL）を使用するビジネス・プロセスの場合は、以下を実行します。
  - 可能な場合は、割り込み可能なプロセス（*macroflow* または*実行時間の長いプロセス*とも呼ばれる）ではなく、割り込み不可能なプロセス（*microflow* または*実行時間の短いプロセス*とも呼ばれる）を使用します。macroflow が必要なプロセスは多くあります（例えば、ヒューマン・タスクを採用する場合や状態を持続する必要がある場合）。ただし、極めて大量のパフォーマンス・リソースが macroflow に関連付けられます。ソリューションの一部で macroflow が必要な場合は、microflow を最大限に利用するために、ソリューションを microflow と macroflow の両方に分けます。詳しくは、10 ページの『可能な場合は microflow を選択』を参照してください。
  - タスク・リスト照会およびプロセス・リスト照会の場合は、複合照会テーブルを使用します。照会テーブルは、高ボリュームのタスク・リスト照会およびプロセス・リスト照会の場合に迅速な応答時間になるように設計されています。詳しくは、10 ページの『タスク・リスト照会およびプロセス・リスト照会用の照会テーブルの選択』を参照してください。

– 作業マネージャー・ベースのナビゲーションを使用して、実行時間の長いプロセスのスループットを向上します。この最適化によって、割り振られるオブジェクトの数、データベースから取得されるオブジェクトの数、および Business Process Choreographer メッセージングに送信されるメッセージの数が削減されます。詳しくは、65 ページの 4.5.1、「ビジネス・プロセス用の作業マネージャー・ベース・ナビゲーションのチューニング」を参照してください。

– 非同期呼び出しを不必要に使用しないようにします。モジュールのエッジで同期呼び出しが必要なことはよくありますが、モジュール内では必要になりません。39 ページの『可能な場合に、「優先相互作用形式」を「同期」に設定』で説明されているように、同期優先の対話スタイルを使用します。

– Service Component Architecture（SCA）および BPEL でトランザクション境界を過度に細分化しないようにします。各トランザクション・コミットによって、データベースおよびメッセージングの動作のコストが高くなってしまいます。37 ページの 3.2.6、「トランザクション上の考慮事項」で説明されているように、トランザクションは慎重に設計してください。

2.2 モデリング

ここでは、モデリングのベスト・プラクティスについて説明します。BPM V7.5 Advanced Edition には、2 つのオーサリング環境が用意されています。

- *Process Designer* は、ハイレベルの Business Processing Modeling Notation（BPMN）ビジネス・プロセスのモデリングと実行に使用されます。ここでは対話型操作が関与することが多いです。Process Designer は、BPM V7.5 Standard Edition の唯一のオーサリング・ツールです。
- *Integration Designer* は、自動化されているサービスまたは他のサービスを開始するサービスの作成と実装に使用されます。これらのサービスには、エンタープライズ内に存在する、Web サービス、エンタープライズ・リソース・アプリケーション、IBM CICS® および IBM IMS™ で実行されるアプリケーションなどがあります。Integration Designer は、BPEL ビジネス・プロセスの作成に使用するツールでもあります。

これらのオーサリング環境は両方とも、共有リポジトリーおよびランタイム環境である Process Center と対話します。

異なるロールおよびスキル・セットを持つ 2 人のユーザーが、これらの環境を使用してビジネス・プロセス・マネジメント・アプリケーションを協同で開発します。

- *ビジネス作成者*は、すべてのビジネス・プロセスの作成を担当します。ビジネス作成者は、サービスを使用できますが、実装の詳細やサービスの機能については関心を持ちません。ビジネス作成者は、Process Designer を使用して、ビジネス・プロセス・ダイアグラム（BPD）および拡張統合サービス（AIS）を作成します。
- *統合プログラマー*は、ビジネス作成者が作成するプロセスのサポートに必要なすべての統合作業を担当します。例えば、統合プログラマーは、すべての AIS を実装し、バックエンドの形式と現行アプリケーションの要件との間のマッピングを作成します。統合プログラマーは、Integration Designer を使用します。

このセクションの残りの箇所はユーザー・タイプに基づいてまとめられており、個別のセクションで一般的なベスト・プラクティスと Process Designer（ビジネス作成者向け）および Integration Designer（統合プログラマー向け）のベスト・プラクティスについて説明します。

2.2.1 一般的なベスト・プラクティス

ここでは、BPM V7.5 の要素の設計と構成に使用する一般的な方法の概要を示します。

プロセスの適切な細分度の選択

ビジネス・プロセスとその個々のステップには、ビジネスの重要度を指定する必要があり、プログラミング・レベルの細分度を模倣しないようにする必要があります。ビジネス重要度を指定しないロジックには、Plain Old Java Object（POJO）や Java スニペットなどのプログラミング技術を使用します。この題材については、以下の Web サイトから入手できる IBM developerWorks® の記事『*Software components: Coarse-grained versus fine-grained*』で詳しく説明されています。

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

イベントの賢明な使用

BPM V7.5 におけるイベント生成の目的は、ビジネス・アクティビティーのモニタリングです。イベント生成では、永続的なメカニズムが使用されるため、本質的にハイパフォーマンスのプロセッサー・リソースが与えられます。ビジネス関連性があるイベントにのみ Common Base Event を使用します。データベースに対してイベントを生成しないでください。その代わりに、イベント生成はメッセージング・インフラストラクチャーを通して実行してください。ビジネス・アクティビティー・モニタリングと IT モニタリングを混同しないでください。IT モニタリングには、Performance Monitoring Infrastructure（PMI）の方がはるかに適しています。

一般的に、ほとんどの顧客について、以下の原則があります。

- 顧客は、自分たちのビジネスとプロセスの状態に関心を持っています。このため、状態の変化を表すイベントが重要です。実行時間の長いヒューマン・タスク・アクティビティーの場合、状態における変化はかなり一般的です。実行時間の長いアクティビティーの完了のトラッキングにイベントを使用して、ヒューマン・タスクの状態が変化したときなどを検出します。
- 数秒で完了する実行時間の短いフローの場合は、1 つのフローが完了したことを認知し、その関連データもあれば十分です。通常、microflow 内で、数ミリ秒のイベントと数秒のイベントを区別することには意味がありません。このため、microflow の場合は 2 つのイベント（開始と終了）で十分です。

2.2.2 Process Designer アーキテクチャーのベスト・プラクティス

ここでは、Process Designer のベスト・プラクティスを示します。

Process Designer と Process Center 間での高速接続の使用

オーサリング・タスクの場合、Process Designer は、Process Center と頻繁に対話します。このため、最適な応答時間を確保できるようにネットワーク待ち時間を最小限に抑えます。Process Center を Process Designer ユーザーと物理的に同じ場所に配置します。Process Center を再配置できない場合は、Process Center が物理的に配置されている環境にリモートで接続して、Process Designer をそのメカニズムを通して使用できます。

効率的なタスクの開発

次に、効率的なタスク開発のガイドラインを示します。

- 可能な場合は、「タスクなし」サービスを使用してシステム・レーン・タスクを呼び出し、タスクがデータベースに書き込まれないようにします。
- ビジネス・データ検索を、プロセス・ポータルで必ず検索可能であることがわかっているフィールドのみに制限します。

変数使用の管理

変数は、実行コンテキストの保存時にデータベースに保持されます。これはかなり頻繁に生じることです（つまり、BPD からサービス実行への変更時および各 COACH の実行時）。これらのパーシスタンス処理はコストが高いです。以下の方法で、パーシスタンス処理のコストを最小限に抑えます。

- 使用する変数の数を最小限に抑えます。
- 各変数のサイズを最小限に抑えます。
- 変数（DB 結果セットなど）が不要になったら、これらの変数をヌルに設定します。
- 各タスクに渡される変数の数とサイズを最小限に抑えます。
- 業界標準スキーマを使用している場合は（ACORD や HIPAA など）これらのスキーマに多くのフィールドが含まれていることを認識して、スキーマから求められる変数のみを使用します。必要に応じて、業界標準スキーマに変換したり、業界標準スキーマから変換したりします。ただし、BPD 処理で不要なパーシスタンス処理のコストがかからないように、アプリケーションのエッジでのみ行います。

不要な場合のビジネス・プロセス・ダイアグラムでの自動トラッキングの無効化

BPD に対して、自動トラッキングはデフォルトで有効になっています。この機能によって、主要なビジネス・メトリックの収集、トラッキング、およびレポート作成が有効になるため、この機能は多くの BPD にとって重要です。ただし、イベントが Performance Data Warehouse で処理され、データベースに保持されるため、自動トラッキングによっていくらかのコストが生じます。ビジネス・メトリックのトラッキングとレポート作成が必要でない BPD については、自動トラッキングを無効にしてください。

永久に実行されるビジネス・プロセス・ダイアグラムの回避

BPD によっては、永久に実行されるものがあります。永久に実行する必要があるビジネス・プロセスもありますが、このタイプの BPD は、その機能が完全に必要な場合にのみ設計してください。永久に実行される BPD は新規イベントに対して絶えずポーリングを行い、このときサーバー・プロセッサーのリソースが使用されます。ポーリング以外の通信メカニズム（Java Message Service キューなど）の使用を検討してください。ポーリングが必要な場合は、ポーリングの実行には BPD の代わりに Undercover Agent（UCA）を使用します。また、Performance Data Warehouse への過剰なトラフィックを避けるために、これらの BPD に対して自動トラッキングを無効にします。

効率的な COACH の開発

次に、ハイパフォーマンスの COACH 開発のガイドラインを示します。

- カスタム可視性は慎重に使用します。
- 大規模で複雑な COACH の使用は避けます。
- 大規模な繰り返しテーブルは使用せず、代わりにページに結果を表示します。

大規模な JavaScript スクリプトの最小限の使用

JavaScript は解釈が必要で、Java コードなどのその他のメカニズムより処理が遅いため、大規模な JavaScript ブロックは使用しないようにします。また、大規模な JavaScript ブロックによって、非常に大規模なドキュメント・オブジェクト・モデル（DOM）ツリーが生成される可能性があり、この場合、ブラウザーの処理およびレンダリングのコストが高くなります。さらに、大規模な JavaScript ブロックにより、BPM 層に過剰な数のロジックが配置されることになる傾向が大きいです。このため、JavaScript の使用を抑えるようにソリューションをリファクタリングすることが望ましいです。

2.2.3 Integration Designer のベスト・プラクティス

ここでは、Integration Designer 使用のベスト・プラクティスについて説明します。

可能な場合は microflow を選択

macroflow は、必要な場合（実行時間が長いサービス呼び出しやヒューマン・タスクなど）にのみ使用します。microflow の方が、実行時のパフォーマンスが、より高いです。割り込み不可能な microflow インスタンスは、状態の持続なしに 1 つの J2EE トランザクションで実行されます。ところが、割り込み可能な macroflow インスタンスは通常、複数の J2EE トランザクションで実行され、トランザクション境界でデータベースでの状態の持続が必要になります。

可能な場合は、割り込み不可能なプロセスには同期対話を使用します。割り込み不可能なプロセスは、バッキング・データベース・システムで状態を持続する必要がないため、割り込み可能なプロセスより効率的です。

プロセスが割り込み可能かどうかを判別するには、Integration Designer で、**「プロパティー」→「詳細」**をクリックします。**「長期実行プロセス」**チェック・ボックスが選択されていれば、プロセスは割り込み可能です。

何らかの機能を使用するために割り込み可能なプロセスが必要な場合は、最も頻繁に生じるシナリオを割り込み不可能なプロセスで処理し、例外的なケースを割り込み可能なプロセスで処理するように、プロセスを分けます。

タスク・リスト照会およびプロセス・リスト照会用の照会テーブルの選択

照会テーブルは、高ボリュームのタスク・リスト照会およびプロセス・リスト照会の場合に適切な応答時間になるように設計されています。照会テーブルでの照会のパフォーマンスは、以下の方法で向上できます。

- 作業項目へのアクセスの改善によって、データベース照会の複雑性が軽減されます。
- タスク、プロセス・インスタンス、および作業項目に対して、構成可能なハイパフォーマンスのフィルターを使用します。
- 作業項目で許可をバイパスできるように複合照会テーブルを使用します。
- 照会テーブル定義に、タスク・リストおよびプロセス・リストに示される情報を反映できるように、複合照会テーブルを使用します。

詳しくは、以下を参照してください。

- PA71: Business Process Manager Advanced - Query Table Builder

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

- Business Process Choreographer での照会テーブル（IBM BPM 7.5 インフォメーション・センター）

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r5mx/index.jsp?topic=%2Fcom.ibm.wbpm.bpc.doc%2Ftopics%2Fc6bpel_querytables.html

効率的なメタデータ管理の選択

ここでは、メタデータ使用のベスト・プラクティスについて説明します。

複合データ型名の場合は、Java 言語仕様に従う

WebSphere Process Server では、ビジネス・オブジェクト（BO）タイプ名に、Java クラス名で許容される文字を使用できます（アンダースコア (_) など）。ただし、複合データ型名の内部データ表現では、Java 型が使用されます。このように、有効な Java 命名構文には追加の変換が必要ないため、BO タイプが Java 命名基準に準拠している方がパフォーマンスは高くなります。

XML スキーマ定義では匿名の派生型を使用しないようにする

XML スキーマ定義（XSD）の一部の機能（プリミティブ・ストリング型に対する制限など）によって、新規サブタイプを生成する必要がある変更が型に生じます。これらの型が明示的に宣言されない場合は、実行時に新規サブタイプ（派生型）が生成されます。通常、この状況を避ける方がパフォーマンスは高くなります。可能な場合は、プリミティブ型のエレメントに制限を追加しないようにします。制限を回避できない場合は、制限を組み込むようにプリミティブ型を拡張する新規の具象 SimpleType の作成を検討してください。そうすると、XSD エレメントでパフォーマンスを低下させることなくこの型を使用できるようになります。

1 つの XML スキーマ定義から別の XML スキーマ定義のエレメントを参照しないようにする

XSD 間のエレメントの参照は避けます。

例えば、A.xsd でエレメント AElement を定義すると（例 2-1 を参照）、このエレメントは、別のファイル B.xsd で参照される場合があります（例 2-2 を参照）。

例 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>
```

例 2-2 別のファイルから参照される AElement

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

このように実行すると、通常はパフォーマンスが低下します。型を具象的に定義し、新規エレメントではすべてこの型を使用するようにします。A.xsd は、例 2-3 に示したような形になります。

例 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>
```

B.xsd は、例 2-4 に示したような形になります。

例 2-4 BElement XSD

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

可能な場合、データ・オブジェクト・タイプのメタデータを再利用する

アプリケーション・コード内では、BO の作成時などにタイプがよく参照されます。例えば、メソッド DataFactory.create(String URI, String typeName) のように、BO タイプを名前で参照することができます。

メソッド DataFactory.create(Type type) のように、直接参照でタイプを参照することもできます。1 つのタイプが複数回使用される可能性がある場合は、例えば DataObject.getType() を使用してタイプを保持し、そのタイプを後で再利用すると便利です。

ビジネス・ステート・マシンの選択

ビジネス・ステート・マシン（BSM）には、ビジネス・フロー・ロジックの魅力的な実装方法が用意されています。アプリケーションによっては、ビジネス・ロジックをステート・マシンとしてモデリングする方がより直感的で、結果としての成果物の理解がより簡単になるものがあります。ただし、BSM はビジネス・プロセス・インフラストラクチャーを使用して実装されるため、BSM を選択する際には常にビジネス・プロセス全体にパフォーマンスの影響があります。

BSM またはビジネス・プロセスのいずれかを使用してアプリケーションをモデリングでき、パフォーマンスが判別の要因の 1 つである場合は、ビジネス・プロセスを選択します。また、パフォーマンスの最適化については、BSM よりビジネス・プロセスの方が使用可能なオプションが多くあります。

ビジネス・ステート・マシンでの状態遷移の最小化

可能な場合は、BSM で状態遷移を駆動する外部イベントを最小限に抑えます。外部イベント駆動の状態遷移は、パフォーマンスの観点から考えると負担が大きいです。実際に、1 つの BSM の実行にかかる合計時間は、その BSM の存続期間中に発生する状態遷移の数に比例しています。

例えば、ステート・マシンが状態 A → B → B → B → C（4 つの遷移）で遷移する場合は、状態 A → B → C（2 つの遷移）で遷移する場合の 2 倍の時間がかかります。また、自動状態遷移は、イベント駆動の状態遷移より負担が少なくなります。BSM を設計する際には、これらの指針を考慮してください。

2.3 トポロジー

ここでは、ソリューションに適切なトポロジーを選択するための情報を提供します。

2.3.1 適切なハードウェアのデプロイ

BPM 環境でハイパフォーマンスを実現するために必要なリソースが含まれるハードウェア構成を選択することが重要です。ハードウェア構成の選択において主な考慮事項となる項目は次のとおりです。

- コア

  BPM サーバー（Process Server と Process Center）を、複数のコアがある最新のサーバー・システムにインストールします。各製品が、対称型マルチプロセッシング（SMP）スケーリングにおいては垂直方向にスケーラブルで、クラスタリングにおいては水平方向にスケーラブルです。

- メモリー

  BPM サーバーでは、堅固なメモリー・サブシステムおよび十分な物理メモリーの両方がもたらす利益を得ることができます。選択したシステムにサーバー・クラスのメモリー・コントローラーと、可能な限り大きな L2 および L3 キャッシュが備わっていることを確認してください（最小 4 MB の L3 キャッシュが備わったシステムの使用が望ましい）。システムで同時に実行されることが予期される複合アプリケーション（JVM）すべてに対して物理メモリーが十分であることを確認します。64 ビットの JVM（公開されている推奨構成）では、最大ヒープ・サイズが 3 GB 以下の場合、JVM ごとに 4 GB が必要です。ヒープ・サイズが 3 GB を超える場合は物理メモリーを追加します。32 ビットの JVM を使用する場合は、おおよそのガイドラインとして JVM インスタンスごとに 2 GB が必要です。

- ディスク

  メッセージ・ストアおよびデータ・ストア（通常はデータベース層）をホストするシステムに高速ストレージが備わっていることを確認します。高速ストレージとは、ライトバック・キャッシュおよび多数の物理ドライブのディスク・アレイとともに RAID（Redundant Array of Independent Disk）アダプターを使用することです。

- ネットワーク

  ネットワークがシステム・ボトルネックを作成しないように、十分に高速であることを確認します。例えば、専用のギガビット・イーサネット・ネットワークは適切な選択です。

- 仮想化

  IBM AIX® 動的論理パーティショニングまたは VMware 仮想マシンなどの仮想化を使用する場合は注意してください。十分なプロセッサー、メモリー、および入出力リソースが各仮想マシンまたは論理パーティション（LPAR）に割り振られていることを確認します。リソースをオーバーコミットしないようにします。

2.3.2 ローカル・モジュールの同一サーバーへのデプロイ

複数のモジュールを同一の物理サーバーにデプロイすることを計画している場合は、モジュールを同一のアプリケーション・サーバー JVM にデプロイすることによってより高いパフォーマンスを実現できます。これによってサーバーでこの局所性を利用できるようになります。

2.3.3 クラスタリングのベスト・プラクティス

クラスタリングのベスト・プラクティスについては、IBM Redbooks 資料の「IBM Business Process Manager V7.5 Production Topologies」（SG24-7976）を参照してください。この文書で

は、スケーラビリティーと高可用性の両方に適切なトポロジーを選択するための包括的なガイダンスが提供されています。この文書は、以下の場所から入手できます。

http://www.redbooks.ibm.com/redbooks/pdfs/sg247976.pdf

ここでは、この文書の内容を繰り返すことは目的としていません。本書では、パフォーマンスを最大限にするためにトポロジーをスケールアップする際の主要な考慮事項のいくつかを取り上げます。

リモート・メッセージングおよびリモート・サポートのデプロイメント・パターンの使用

スケーリングにおいて柔軟性を最大限にするには、リモート・メッセージングおよびリモート・サポートのデプロイメント環境パターンを使用します。詳しくは、以下の場所から入手できる「*IBM Business Process Manager Advanced インストール・ガイド*」の『デプロイメント環境のトポロジーおよびパターン』を参照してください。

ftp://public.dhe.ibm.com/software/integration/business-process-manager/library/pdf751/imuc_ebpm_dist_pdf_ja.pdf

このトポロジー（以前の「ゴールド・トポロジー」）では、アプリケーションとメッセージング・エンジンで別々のクラスターを使用するよう指示されています。また、サポート・アプリケーション・サーバー（Common Event Infrastructure（CEI）サーバーやビジネス・ルール・マネージャーなど）にクラスターを使用する方法も指示されています。このトポロジーでは、インフラストラクチャーの各エレメントの負荷に対応するためにリソースを独立して制御できます。

柔軟性とコスト：システムに関する多くの選択肢と同様に、柔軟性にはコストが伴います。例えば、このトポロジーでのアプリケーションと CEI サーバーの同期イベント生成はリモート呼び出しであり、ローカル呼び出しより負荷が大きいです。この構成の利点は、アプリケーションとサポート・クラスターを独立してスケーリングできることです。これらの種類のシステム・トレードオフはほとんどのサーバー・ミドルウェアで生じるため、読者がこれらのトレードオフについて理解していることを前提としています。

単一サーバーとクラスター・トポロジーの比較検討

通常、単一サーバー構成からクラスター・トポロジーに移行するかどうかを判断する際に、検討すべき 2 つの主要な問題があります。

- 全体的なパフォーマンスとスループットを向上するためのスケーラビリティーとロード・バランシング
- ハードウェアまたはソフトウェア障害によるサービス損失を防ぐための、別のクラスター・メンバーへのフェイルオーバーを使用した高可用性

互いに排他的ではありませんが、それぞれにいくつかの考慮事項があります。本書では、クラスタリングの高可用性の面ではなく、パフォーマンス（スループット）の面に焦点を当てます。リソースを飽和状態にしているほとんどの単一サーバー・ワークロードは、クラスター・トポロジーへの移行によって活用できます。

2.3.4 サービス・プロバイダーおよび外部インターフェースの評価

BPM V7.5 の一般的な使用パターンの 1 つが、着信要求とビジネスのバックエンド・システム（ターゲット・アプリケーションまたはサービス・プロバイダー）との間の統合層としての使用です。これらのシナリオでは、スループットは、スループット・キャパシティーが最小となっている層によって制限されます。

ターゲット・アプリケーションが 1 つしかない簡単なケースについて考えてみます。BPM V7.5 ベースの統合ソリューションでは、ターゲット・アプリケーションのスループット・

キャパシティーより多いスループット・レートを実現することはできません。スループットを増加できないことは、BPM V7.5 ベースの実装の効率性やそれをホストしているシステムのサイズや速度にかかわらず当てはまります。このため、エンドツーエンド・ソリューションの設計時には、すべてのターゲット・アプリケーションおよびサービス・プロバイダーのスループット・キャパシティーを理解して、この情報を適用することが重要です。

ターゲット・アプリケーションまたはサービス・プロバイダーのスループット・キャパシティーには次の 2 つの重要な側面があります。

- 応答時間（一般的なケースと例外的なケースの両方）
- ターゲット・アプリケーションが同時に処理できる要求数（並行性）

ターゲット・アプリケーションのパフォーマンス面を確立できれば、おおよその最大スループット・キャパシティーを計算できます。同様に、平均スループットがわかっている場合は、これら 2 つの側面のいずれかを大まかに計算できます。例えば、平均応答時間が 1 秒で、1 秒間に 10 個の要求を処理できるターゲット・アプリケーションでは、同時に約 10 個の要求を処理できます（スループット / 応答時間 = 並行性）。

ターゲット・アプリケーションのスループット・キャパシティーは、アプリケーション全体のエンドツーエンドのスループット計画の際に重要です。また、BPM V7.5 ベースのアップストリーム・コンポーネントの並行性レベルのチューニングの際にもターゲット・アプリケーションの並行性を考慮してください。例えば、ターゲット・アプリケーションで同時に 10 個の要求を処理できる場合は、このアプリケーションを始動するプロセス・サーバー・コンポーネントをチューニングします。これらのコンポーネントをチューニングすることによって、BPM V7.5 からの同時要求が少なくともターゲットの並行性機能に適合するようになります。

また、ターゲット・アプリケーションが過負荷にならないようにします。そのように構成してもアプリケーション全体のスループットが増加することにはならないためです。例えば、同時に 10 個の要求のみを処理できるターゲット・アプリケーションに 100 個の要求を送信しても、スループットが増加することはありません。ただし、送信する要求数がターゲットの並行性機能に適合するようにチューニングすることによって、スループットは増加します。

メインライン処理の一部として、または例外ケースで、応答に時間がかかる可能性のあるサービス・プロバイダーでは、応答が必要な同期呼び出しは使用しません。同期呼び出しを使用しないことによって、サービス・プロバイダーが応答するまで、ビジネス・プロセスとそのリソースの関連付けが回避されます。

2.4 Business Space

ここでは、Business Space ソリューションの構築、開発、およびデプロイのベスト・プラクティスを示します。

2.4.1 トポロジーの最適化

Business Space などの Ajax（Asynchronous JavaScript and XML）アプリケーションのパフォーマンスのモデルは、図 2-1 に示しているように、4 層に分類することができます。ハイパフォーマンス・ソリューションを実現するには、各層を最適化する必要があります。トポロジーの最適化の詳細のいくつかについては本書の後半で説明しますが、その説明のコンテキストはここに示します。

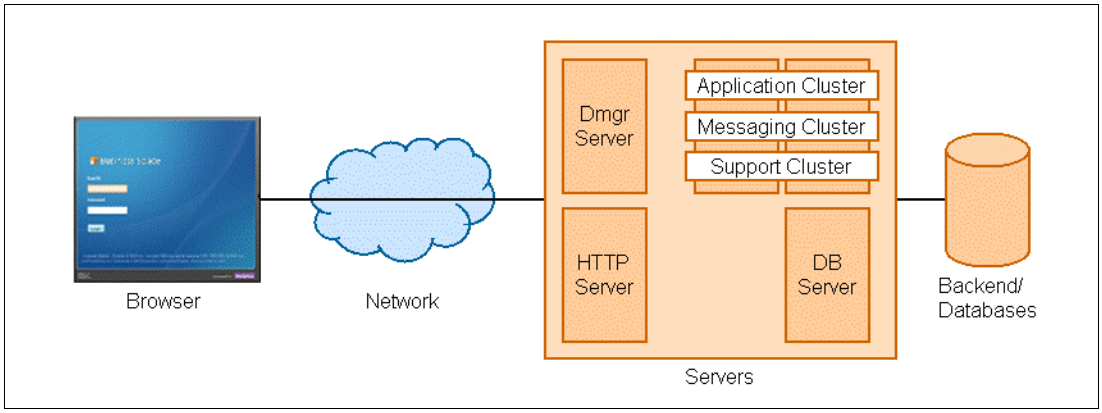

図 2-1 AJAX パフォーマンスの 4 つの層

4 つの各層は、次のとおりです。

- ブラウザー

  定義によれば、Ajax アプリケーションは、クライアント側のブラウザー内で（ある程度まで）処理を実行するアプリケーションです。UI の作成などの通常のクライアント処理はすべてクライアント側で実行されるため、これらのアプリケーションは、HTML のレンダリングのみがブラウザーで実行される従来の Web アプリケーションとは区別されます。16 ページの 2.4、「Business Space」の以下のサブセクションで説明しているように、ブラウザーおよびクライアント・システムの最適化は、ハイパフォーマンスを実現する上での鍵となります。

- ネットワーク

  静的 Web ページと対照的に、Ajax アプリケーションでは、完全なページをすぐにロードするのではなく、必要に応じてページのコンテンツが動的にロードされます。Ajax アプリケーションでは、ペイロードが大きな数個（または 1 つ）の要求ではなく、ペイロードが小さな要求が多数送信されることが多くあります。したがって、ネットワーク遅延により、各メッセージの処理時間が長くなるため、Business Space の応答時間が大きな影響を受ける可能性があります。最後には、ネットワーク遅延によって、ページ全体の応答時間の遅延がもっと大きくなってしまいます。

  ネットワークはサーバーとデータベース間、およびクラスター・セットアップではサーバー間の通信でも役割を果たしているため、図 2-1 は簡略して表してあります。ただし、Ajax アプリケーションの性質により、遅延を分析するにあたっての最初のポイントは通常、ブラウザーとサーバー間です。

- サーバー

  サーバー・インフラストラクチャーでは、接続先クライアントからの要求の処理、ビジネス・アプリケーション（プロセス、ステート・マシン、Web サービス、およびその他のアプリケーション）の実行、およびバックエンド・サービスの統合が行われます。このインフラストラクチャーのセットアップは、選択したトポロジーに大きく依存します。

  以下のサーバーは、Business Space で重要な役割を果たします。

  - HTTP サーバー

    HTTP サーバーは、すべてのトポロジーに含まれているとは限りません。ただし、数千ものユーザーに対応することを目的とした環境では、HTTP サーバーは必要不可欠です。Business Space クライアントからのすべての静的および動的要求が HTTP サーバーに着信されます。このサーバーでは、キャッシュを実行した後に、静的（キャッシュ済み）コンテンツをクライアントに返し、動的要求を WebSphere Process Server REST API に転送できます。さらに、HTTP サーバーでは、ロード・バランシングを指定して、高可用性のシナリオをサポートできます。

  - BPM V7.5 サーバー

    BPM V7.5 サーバーでは、さまざまなビジネス・ロジック（BPEL および BPMN ビジネス・プロセスなど）が実行されます。ただし、ここでは、タスク・リストの照会、タスクの作成と要求、およびその他の機能を含めた、Business Space ウィジェットでの役割を果たす処理のみに焦点を当てます。

- データベース

  WebSphere ソリューションを備えた BPM V7.5 Business Space の場合は通常、2 つのデータベースが重要な役割を果たします。

  - WebSphere データベースを備えた Business Space

    Business Space データベースには構成関連情報が保持され、各ユーザーはこのデータベースにはまれにしかアクセスしません。

  - Business Process Choreographer データベース（BPEDB）

    すべてのタイプのプロセスおよびタスク情報が BPEDB に保管されるため、このデータベースはヒューマン・タスクのウィジェットによって頻繁に更新されたり読み取られたりします。

2.4.2 待ち時間が短く帯域幅が広いネットワークの使用

繰り返しですが、Business Space V7.5 は AJAX アプリケーションの 1 つです。このため、Business Space V7.5 では、ユーザー処置ごとに複数の HTTP 要求が使用されるので、要求ごとのネットワーク遅延を最小限に抑えることが重要です。HTTP 要求によっては大量のデータを返すものがあるため、ネットワーク帯域幅も重要です。

可能な場合は、Business Space クライアントとサーバー間の接続に、高速で高帯域幅（1 GB 以上）の専用ネットワークを使用してください。

2.4.3 ハイパフォーマンスのブラウザーの使用

ブラウザー・テクノロジーの選択は、Business Space のパフォーマンスにとって重要です。Internet Explorer（IE）の場合は、ブラウザーの選択が重要です。最近のバージョンの IE は通常、以前のバージョンの IE よりもパフォーマンスが高くなっています。

2.4.4 ブラウザー・キャッシュの有効化

通常、ブラウザーでは、静的データは最初にサーバーから取得された後にキャッシュされ、これによってキャッシュの準備後のシナリオでは応答時間を大きく向上できます。この向上は特に、待ち時間が比較的長いネットワークの場合に顕著です。ブラウザー・キャッシュが

アクティブで有効になっていることを確認してください。キャッシュの設定はブラウザー固有です。詳しくは、71 ページの 4.9.1、「Internet Explorer 8 でのブラウザー・キャッシュの有効化」を参照してください。

2.4.5 物理的にクライアント近くへのサーバーの配置

ネットワーク待ち時間に影響を与える要因の 1 つが、サーバーとクライアントの物理的距離と、クラスター内でのサーバー間の距離です（例えば、サポート・クラスターと BPM V7.5 アプリケーション・サーバーとの距離）。実用的な場合は、BPM V7.5 サーバーを互いに物理的に近くに、また Business Space クライアントの物理的に近くに配置して、要求の待ち時間を最小限に抑えます。

2.4.6 最新のデスクトップ・ハードウェアの使用

Business Space ソリューションでは、処理のほとんどがクライアント・システムで（ブラウザー・レンダリングを通して）実行されます。このため、十分な物理メモリーと、大規模なキャッシュおよび高速のフロント・サイド・バスを備えた高速プロセッサーを持つ最新のデスクトップ・ハードウェアの導入が不可欠です。パフォーマンス・ツール（Windows タスク・マネージャーまたは vmstat）を使用してクライアント・システムをモニターし、クライアント・システムにハイパフォーマンスを確保するのに十分なプロセッサーおよびメモリー・リソースがあるようにしておきます。

2.5 ラージ・オブジェクト

現場担当者がよく直面する問題の 1 つが、BPM V7.5 サーバーおよび対応する WebSphere アダプターが効果的および効率的に処理できる最大オブジェクト・サイズの特定です。これらの各製品でラージ・オブジェクトの処理に影響を与える要因はたくさんあります。ここでは、関係する要因の説明とこれらの製品の現在のリリース向けの実用的なガイドラインを示します。最大オブジェクト・サイズの特定の問題は、主に 32 ビットの JVM に適用されます。この環境にはヒープ・サイズの制約があるからです。

JVM はラージ・オブジェクトの処理に影響を与える単一の最も重要な要因です。JVM テクノロジーは、Java 5 から続けて変更されており、これは BPM 製品では V6.1.0 で初めて使用されました。BPM V7.5 では、Java 5 の後続リリースである Java 6 JVM が使用されます。本書で示される提案およびベスト・プラクティスは、JVM 1.4.2 を使用していた BPM V6.0.2 およびこれ以前のものとは異なります。

通常、5 MB 以上のオブジェクトは、「大きい」と見なされ、特別な注意が必要です。100 MB 以上のオブジェクトは「非常に大きい」と見なされ、通常は適切に処理するためには大がかりなチューニングが必要です。

2.5.1 ラージ・オブジェクト・サイズの処理に影響を及ぼす要因

通常、インストール環境のオブジェクト・サイズのキャパシティーは、Java ヒープおよびそのヒープに現在のレベルの着信作業によってかかる負荷（つまり、ライブ・セット）のサイズによって異なります。ヒープが大きければ、適切に処理できる BO も大きくなります。

この原則を適用するには、まずオブジェクト・サイズの制限が、JVM についての次の 3 つの基本的な実装ファクトに基づいていることを理解する必要があります。

- Java ヒープ・サイズの制限

Java ヒープ・サイズの制限は、オペレーティング・システムに依存しますが、32 ビットの JVM でヒープ・サイズ制限が 1.4 GB 前後であることも珍しくありません。ヒープ・サイズの上限は 64 ビットの JVM ではもっと高く、通常、最新のハードウェア構成では使用可能な物理メモリーほど制御要因とはなりません。ヒープ・サイズについて詳しくは、56 ページの『Java ヒープ・サイズの最大化』を参照してください。

- メモリー内 BO のサイズ

  BO は、Java オブジェクトとして表される場合、ワイヤー形式で表される場合よりもサイズがずっと大きくなります。例えば、入力 Java Message Service（JMS）メッセージ・キューで 10 MB 使用する BO が、Java ヒープでは 最大 90 MB の割り振りになる場合があります。この状態は、BO がアダプターおよび BPM V7.5 をフローしていくときに発生する大小の Java オブジェクトの多数の割り振りによって生じます。メモリー内での BO の拡張に影響を与える要因は次のようにたくさんあります。

  – 1 バイトのバイナリー・ワイヤー表記は通常、マルチバイト文字表記（Unicode など）に変換されるため、この結果、2 つのうちの 1 つの拡張要因が生じます。

  – BO にはたくさんの小さなエレメントや属性が含まれる場合があり、それぞれの名前、値、およびその他のプロパティーをいくつかの固有 Java オブジェクトによって表す必要があります。

  – 各 Java オブジェクトには、最小のものの場合でも、ほとんどの 32 ビット JVM で 12 バイト長、および 64 ビット JVM ではもっと長くなる内部オブジェクト・ヘッダーのために、リソースに対する固定の要件があります。

  – Java オブジェクトは、8 バイトまたは 16 バイトのアドレス境界を調整するために埋め込まれます。

  – BO がシステムをフローしていくとき、BO は変更されたりコピーされたりするため、エンドツーエンドのトランザクション中のどの時点においても複数のコピーが存在する可能性があります。BO の複数のコピーが存在することは、トランザクションが正常に完了するためには、Java ヒープがこれらすべての BO コピーをホストするのに十分大きくなければならないことを意味します。

- 同時処理されるオブジェクトの数

  同時に処理される要求の数が増えると、正常に処理可能なオブジェクトのサイズが小さくなります。これは、各要求には、システムを進んでいくときの固有メモリー使用プロファイル（ライブ・セット）があるためです。複数のラージ・オブジェクトを同時に処理すると、必要なメモリーの量が劇的に多くなります。これは、要求のライブ・セットの合計が、構成されているヒープに収まる必要があるためです。

2.5.2 ラージ・オブジェクトの設計パターン

以下のセクションでは、ラージ・オブジェクトを適切に処理するための、2 つの実証済み設計パターンについて説明します。どちらの設計パターンも適用できない場合は、64 ビット・モードの使用を検討してください。詳しくは、21 ページの 2.6、「64 ビットに関する考慮事項」を参照してください。

また、IBM WebSphere ESB の場合、以下の Web サイトから、サイズが大きいメッセージのベスト・プラクティスとチューニングの詳細を示した developerWorks の記事を入手できます。

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

バッチ入力：ラージ・オブジェクトを複数のスモール・オブジェクトとして送信

ラージ・オブジェクトを処理する必要がある場合、ソリューション・エンジニアは、割り振られる大きな Java オブジェクトの数を制限する方法を見つける必要があります。オブジェクト数を制限する主要技法として、大きな BO を小さなオブジェクトに分解してそれを個別に送信するものがあります。

ラージ・オブジェクトが実際にはスモール・オブジェクトの集合である場合、ソリューションではそれらのスモール・オブジェクトは、サイズが 1 MB 未満の複合オブジェクトにグループ分けされます。一部の顧客サイトには、このようにして統合されたスモール・オブジェクトが含まれ、良い結果が生み出されます。個々のオブジェクトについて、一時的な依存関係または絶対的な要件が存在する場合、ソリューションはもっと複雑になります。顧客サイトでの実装では、パフォーマンスと安定度の向上が見られるため、この複雑性の処理は努力に値することが明らかです。

一部の WebSphere アダプター（Flat Files アダプターなど）を、SplitCriteria を個々のオブジェクトのサイズに設定して SplitBySize モードを使用するよう構成できます。この場合、ラージ・オブジェクトは、ピーク・メモリー使用量を軽減するためにチャンク（SplitCriteria で指定されているサイズ）に分割されます。

要求チェック・パターン : 入力メッセージのごく一部のみを使用

入力 BO がシステム内でやりとりするには大きすぎる上、プロセスまたはメディエーションには少数の属性しか必要ではない場合、「要求チェック・パターン」と呼ばれるパターンを使用できます。要求チェック・パターンの使用を BO に適用するステップは、以下のとおりです。

1. メッセージからデータ・ペイロードを切り離します。
2. 必須属性を小さめのコントロール BO に抽出します。
3. この大きなデータ・ペイロードをデータ・ストアに保持して、「要求チェック」をこの小さめのコントロール BO にリファレンスとして保管します。
4. メモリー占有スペースが小さいこの小さめのコントロール BO を処理します。
5. ソリューションに大きなペイロード全体が再度必要になった場合は、キーを使用してデータ・ストアからこの大きなペイロードをチェックアウトします。
6. データ・ストアから大きなペイロードを削除します。
7. コントロール BO 内の変更した属性を考慮に入れて、コントロール BO 内の属性を大きなペイロードにマージします。

要求チェック・パターンを使用する場合、ソリューションにカスタム・コードとスニペットを指定する必要があります。開発者の負担が少ないバリアントでは、コントロール BO の生成にカスタムのデータ・バインディングを使用します。この方法は、所定のエクスポート・バインディングおよびインポート・バインディングに限定されます。JVM でのフル・ペイロードの割り振りは引き続き必要です。

2.6 64 ビットに関する考慮事項

BPM アプリケーションは、32 ビットまたは 64 ビットの JVM を使用して実行できます。ただし、32 ビット・モードでは最大ヒープ・サイズが 4 GB アドレス・スペース・サイズによって制限されるため、Process Server と Process Center の両方に 64 ビット・モードを使用することをお勧めします。ほとんどの 32 ビットのオペレーティング・システムで、実際の制限値は 1.5 GB から 2.5 GB です。これと対照的に、64 ビット・モードでは最大ヒープ・サイズは無制限である一方、標準の Java ベスト・プラクティスは引き続き適用されます。

システムで実行されるすべての Java プロセスの最大ヒープ・サイズおよびネイティブ・メモリー使用量の合計が、システムで使用可能な物理メモリーを超えてはいけません。この合計値には、オペレーティング・システムとその他のアプリケーションに必要な追加メモリーも含まれます。Java プロセスには、スレッド、スタック、および JIT（Just In Time）コンパイル済みコードがあります。

BPM V7.5 サーバーは、64 ビット・モードではアクセス可能なメモリー容量が大きいため、64 ビットの JVM インスタンスで最も効率的に実行されます。64 ビット・ランタイム・サーバーのパフォーマンスとメモリー占有スペースは、32 ビット・バージョンの場合とほぼ同じです。ただし、BPM V6.1 および V6.2 では、64 ビット JVM のメモリー占有スペースは、32 ビットの場合のパフォーマンスおよびメモリー占有スペースほど良くありません。

以下のリストに、これらのモードのいずれで実行するかを決定する際に考慮すべき要素の詳細を示します。

- 64 ビット・モードは、ライブ・セットが 32 ビットの制限値に達しているかまたは超えているアプリケーションの場合に良い選択です。このようなアプリケーションでは、OutOfMemoryExceptions が発生するか、ガーベッジ・コレクション（GC）で過度な時間がかかっています。GC の時間の 10% を超える分が過度と見なされます。これらのアプリケーションでは、必要とするよりもヒープを大きく設定して実行できるようにした場合にパフォーマンスが大きく向上します。ただし、Java ヒープ・サイズに対応するにはシステム上に十分な物理メモリーが必要です。
- 64 ビット・モードは、32 ビット・モードで適切に動作しているが、ヒープを拡大してパフォーマンスが向上するようにアルゴリズムで変更できるアプリケーションの場合にも良い選択です。1 つの例として、非常に大きなメモリー内キャッシュの保持を回避するために、このようなキャッシュがスループットを大きく向上させる場合でも、データをデータ・ストアに頻繁に保存するアプリケーションが挙げられます。このようなアプリケーションを再コード化して、64 ビット・ヒープの使用可能スペースの拡大と実行時間の短縮をトレードオフすることによって、パフォーマンスを向上できます。
- 32 ビット・アプリケーションが 1.5 GB から 2.5 GB のヒープに適合しており、このアプリケーションが大きく拡張することは考えにくい場合、64 ビットへの移行によって、スループットに機能低下が生じる可能性があります。このような状況でメモリー制限が大きな要因ではない場合は、32 ビット JVM の方が 64 ビット JVM よりも良い選択と考えられます。

2.7 Business Monitor

ここでは、Business Monitor V7.5 を使用する場合のパフォーマンスのベスト・プラクティスについて説明します。

2.7.1 イベント処理

イベント処理パフォーマンスでの主な要素は、Business Monitor データベースのチューニングです。ディスク読み取りアクティビティーを最小限に抑えるように適切なバッファー・プール・サイズを確保し、データベース・ログの配置も確認します。データベース・ログは、理想的には、データベース表スペースとは物理的に別のディスク・サブシステム上に配置します。

デフォルトで、イベントは、中間キューをバイパスして、CEI からモニター・データベースに送達されます。このデフォルトの送達スタイルでは、フローでの追加のパーシスタンス・ステップが回避されるため、パフォーマンスを向上するためにはこのデフォルトの送達スタイルを使用することをお勧めします。

追加の背景については、IBM Business Monitor V7.5 Information Center の『表ベースのイベント・デリバリーを使用したイベントの受信』を参照してください。

http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r5mx/index.jsp?topic=/com.ibm.wbpm.mon.imuc.doc/inst/cfg_qb.html

2.7.2 ダッシュボード

Business Space およびその Business Monitor ウィジェットのプラットフォーム要件は、Business Monitor サーバーのウィジェットおよびデータベース・サーバーに比べて比較的軽微です。ダッシュボードのハイパフォーマンスを実現するための最も重要な考慮事項は、データベース・サーバーを適切にサイズ設定および構成することです。データベース・サーバーに、予期されるデータ・マイニング照会用の十分なプロセッサー・キャパシティー、バッファー・プール用の十分なメモリー、およびたくさんのディスク・アームを備えるようにします。

2.7.3 データベース・サーバー

イベント処理およびダッシュボードでは両方とも、ハイパフォーマンスを実現できるかどうかは、高速で適切にチューニングされたデータベース・サーバーに依存しています。Business Monitor の設計では、これを使用する顧客はすべてオンサイト・データベース管理者としての優れた技術を持っていることが前提となっています。77 ページの 4.10、「一般的なデータベース・チューニング」を初めとして示されているデータベース・チューニングについての提案を適用することが重要です。

3

開発のベスト・プラクティス

本章では、ソリューション開発者に関連するベスト・プラクティスを示します。BPM V7.5 ソリューションの設計および実装の際の、モデリング、設計、および開発上の選択について説明します。Business Process Manager（BPM）Advanced Edition には、2 つのオーサリング環境が用意されています。これらのオーサリング環境は両方とも、共有リポジトリーおよびランタイム環境である Process Center と対話します。

- Process Designer 環境では、ハイレベルのビジネス・プロセスをモデリングしたり実行したりでき、このとき対話型操作が多く行われます。Process Designer は、BPM V7.5 Standard Edition の唯一のオーサリング・ツールです。
- Integration Designer 環境では、自動化するサービス、または他のサービスを開始するサービス（Web サービス、エンタープライズ・リソース・アプリケーション、CICS および IMS で実行されるアプリケーションなど）を作成したり実装したりできます。これらのサービスおよびアプリケーションは、エンタープライズ内に存在します。Integration Designer は、BPEL（Business Process Execution Language）ビジネス・プロセスの作成に使用するツールでもあります。

異なるロールおよびスキル・セットを持つ 2 人のユーザーが、BPM アプリケーションの開発時に協同で作業します。これらのロールは、Process Designer 環境および Integration Designer 環境に対応しています。

- *ビジネス作成者* は、すべてのビジネス・プロセスの作成を担当します。ビジネス作成者は、サービスを使用できますが、実装の詳細やサービスの機能については関心を持ちません。ビジネス作成者は、Process Designer を使用して、ビジネス・プロセス・ダイアグラム（BPD）および拡張統合サービス（AIS）を作成し、統合プログラマーと共同作業を行います。
- *統合プログラマー* は、ビジネス作成者が作成するプロセスのサポートに必要なすべての統合作業を担当します。例えば、統合プログラマーは、すべての AIS を実装し、バックエンドの形式と現行アプリケーションの要件との間のマッピングを作成します。統合プログラマーは、Integration Designer を使用します。

本章の残りの箇所はユーザー・タイプに基づいてまとめられており、個別のセクションで Process Designer（ビジネス作成者向け）および Integration Designer（統合プログラマー向け）のベスト・プラクティスについて説明します。また、本章では、開発者に向けて、Business Space ソリューション、WebSphere InterChange Server マイグレーション、およびオーサリング環境に関する考慮事項も提示します。

3.1 Process Designer 開発のベスト・プラクティス

Process Designer を使用したハイパフォーマンスのビジネス・プロセスの開発に関するベスト・プラクティスは、以下のとおりです。

3.1.1 終了が意図されていない公開ヒューマン・サービスの変数のクリア

タスクなしヒューマン・サービスのデータは、そのサービスがエンドポイントに達するまでガーベッジ・コレクションに収集されません。単一ページやリダイレクトなど、エンドポイントに達することを意図されていないヒューマン・サービスを開発した場合、Enterprise JavaBeans（EJB）のタイムアウトが発生するまで（デフォルトで 2 時間）メモリーはガーベッジ・コレクションに収集されません。メモリーの使用を軽減するために、COACH のカスタム HTML ブロックで変数をヌルに設定します。

3.1.2 システム・レーンで、またはバッチ・アクティビティーに対して、マルチインスタンス・ループを使用しない

下位 BPD をマルチインスタンス・ループ（MIL）のアクティビティーとして使用するときは注意が必要です。最初のアクティビティーがシステム・レーンではなくユーザー・タスクの場合は問題ありません。ただし、バッチ・アクティビティーまたはシステム・レーン・アクティビティーの実行に MIL を使用しないでください。このパターンによって、BPD で処理する必要のあるトークンの数が過剰になる可能性があります。また、システム・レーンでは MIL のアクティビティーは単一スレッドで実行されるため、マルチプロセッサーのコア・サーバーでは明らかに最適ではありません。

図 3-1 は、不適切な BPD 設計パターンの例を示しています。

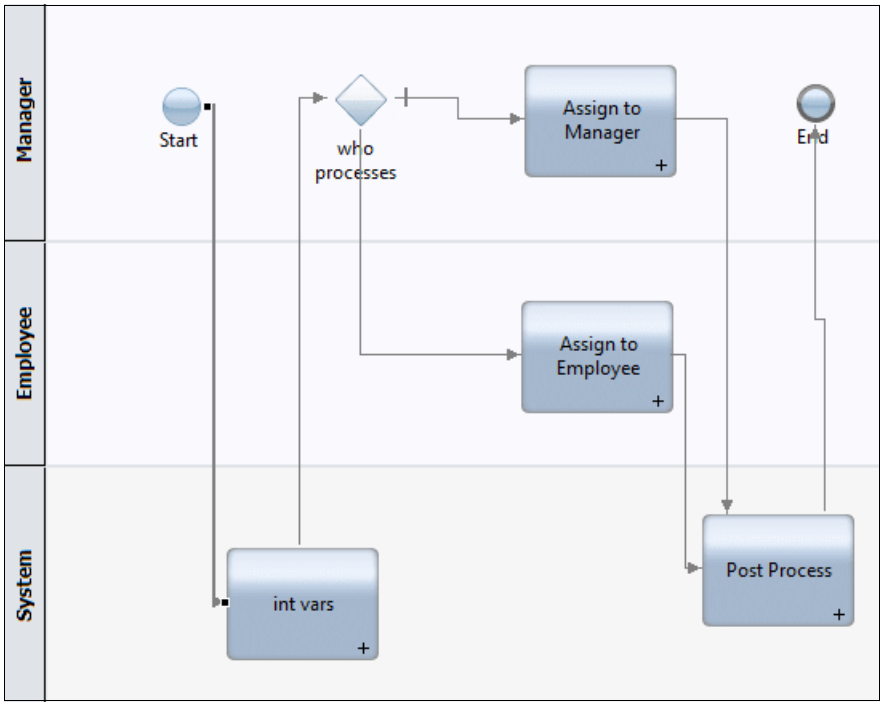

図 3-1　不適切な BPD 設計パターン

図 3-2 は、適切な BPD 設計パターンの例を示しています。

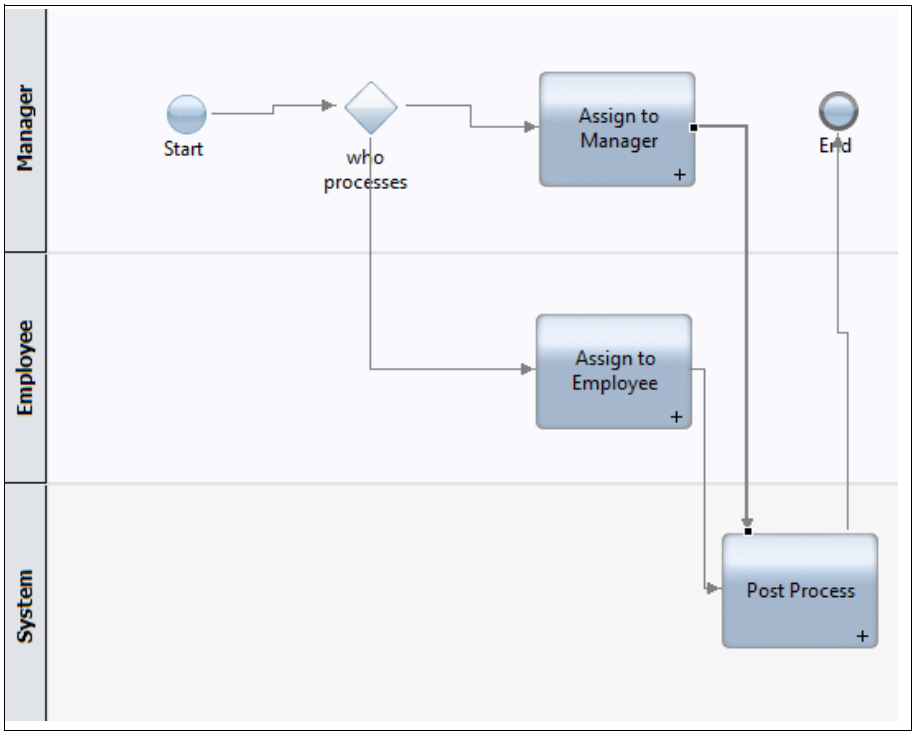

図 3-2　適切な BPD 設計パターン

3.1.3 必要な場合のみの条件付き結合の使用

単純結合では、「and」条件が使用されます。トークンが続行するには、結合に向かっている行すべてにアクティブ・トークンが必要です。

一方、条件付き結合の場合、トークンが続行するには、考えられるトークンすべてが結合に達している必要があります。このように、3 つの着信データ行を使用した条件付き結合があるが、現在それらのうちの 2 つのみにトークンがある場合（またはアップストリームを見ればトークンがある可能性がある場合)、これらの 2 つのトークンが続行するには結合に達している必要があります。この状態を判別するには、考えられるすべてのアップストリーム・パスを BPD エンジンで評価して、トークンが結合に達することができるかどうかを判断する必要があります。この評価は、大規模で複雑な BPD の場合、コストが高くなることがあります。このような場合、この機能は注意して使用してください。

3.1.4 エラー・ハンドリングの指針

サーバー・プロセッサーを過剰に使用する可能性があり、COACH での無限ループさえ招きかねないグローバル・エラー・ハンドリングをサービスで実行しないようにします。

BPD のアクティビティーでエラーをキャッチした場合は、エラーを同じアクティビティーに転送で戻さないでください。そのように戻すと、プロセッサーおよびデータベースの処理能力を大量に使用することになり、BPD とサービス・エンジンの間でサーバーがスラッシングを起こします。

3.1.5 順次システム・レーン・アクティビティーの効率的な使用

各システム・レーン・アクティビティーは、新規イベント・マネージャー・タスクと見なされ、サービス・エンジンにタスク遷移が追加されます。これらのタスク遷移はコストが高いです。BPD で 1 行に複数のシステム・レーン・サービス・タスクが含まれる場合、遷移に必要な余分のリソースを最小限に抑えるために、1 つのシステム・レーン・タスクを使用して、その他のシステム・レーン・タスクをラップします。1 つのシステム・レーンの使用は、1 つの参加者レーン内の複数の連続したタスクにも適用されます。ただし、通常、参加者レーン内の各タスク間では 1 つの処理しか必要でないため、このようなパターンはあまり一般的ではありません。

図 3-3 は、不適切な使用パターン（複数の連続したシステム・レーン・アクティビティーを使用）を示しています。

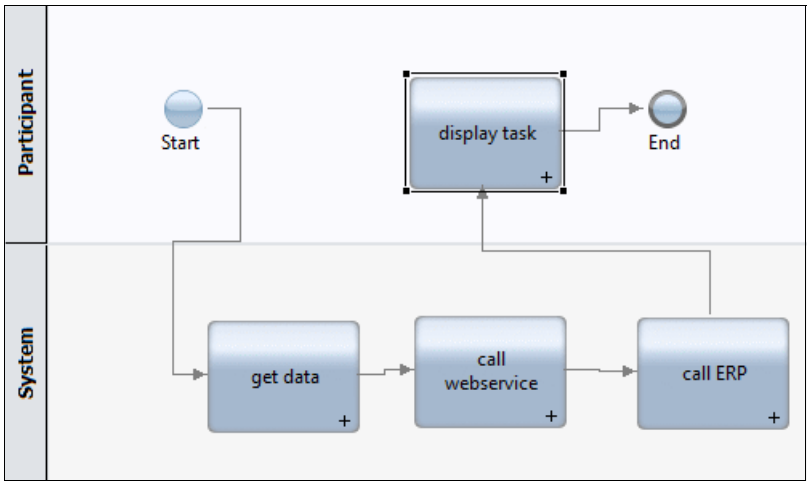

図 3-3　不適切な BPD 使用パターン

図 3-4 は、より適切な使用パターン（図 3-3 の複数のアクティビティーを組み込んだ 1 つのシステム・レーン・アクティビティー）を示しています。

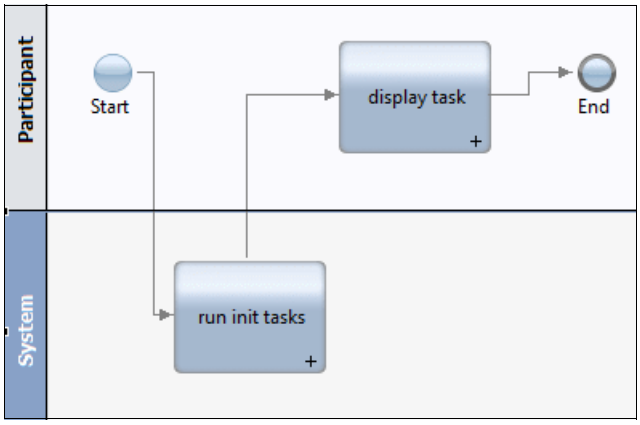

図 3-4　適切な BPD 使用パターン

3.1.6 Process Center のチューニングの確保

少なくとも、以下の方法で Process Center のパラメーターをチューニングします。

- 64 ビット JVM を使用します。
- 引数 **-Xgcpolicy:gencon JVM** を指定して、世代別並列ガーベッジ・コレクターを使用します。
- ピーク・ロードを処理するのに十分な大きさに Java ヒープ・サイズを設定します。Java ヒープ・サイズの設定について詳しくは、52 ページの 4.3.2、「Java のチューニング・パラメーター」を参照してください。
- ORB（Object Request Broker）のスレッド・プール・サイズを 20 以上に設定します。
- Process Server データ・ソース（jdbc/TeamWorksDB）の JDBC（Java Database Connectivity）最大接続数を、セル内の各ノードの ORB スレッド・プールに許可される最大スレッド数の 2 倍以上に設定します。例えば、2 つの Process Center ノードが存在し、各ノードの ORB サービスのスレッド・プール最大数を 20 スレッドに設定した場合、80 接続以上を受け入れるように Process Server データ・ソースを設定します。
- Process Center データベースをチューニングします。詳しくは、77 ページの 4.10、「一般的なデータベース・チューニング」を参照してください。

3.1.7 Process Designer と Process Center 間での高速接続の使用

オーサリング・タスクの場合、Process Designer は、Process Center と頻繁に対話します。これらのコンポーネント間の対話を促進させるステップを実行します。Process Designer と Process Center 間の接続について詳しくは、8 ページの『Process Designer と Process Center 間での高速接続の使用』を参照してください。

3.1.8 Web サービス記述言語の検証による Web サービス統合の遅延の防止

Process Designer Web サービス統合コネクターは、Web サービスを開始するために、実行時に複数のステップを実行します。システムによって、メタデータおよびビジネス・オブジェクト（BO）から SOAP 要求が生成されます。次に、Web サービス記述言語（WSDL）に対して要求が検証され、実際の SOAP 呼び出しが HTTP 経由で実行され、結果が BO に解析して戻されます。これらのステップではそれぞれ待ち時間が生じる可能性があります。このため、実際の Web サービス応答時間が速くなるようにするだけでなく、WSDL に対して要求を迅速に検証できるようにすることも重要です。頻繁に開始される可能性のある Web サービスにとって速度は特に重要です。

検証の遅延には主に 2 つの原因があります。

- WSDL の取得における応答の遅延
- 深くネストされた WSDL include

WSDL のソースがリモート・ロケーションの場合、WSDL を HTTP 経由で取得する際の待ち時間によって全体的な待ち時間が長くなります。このため、遅い接続、遅いプロキシー、または遅いサーバーによって、全体的な Web サービス呼び出しの待ち時間が長くなる可能性があります。WSDL で import および include によって追加の WSDL ファイルまたは XML スキーマ定義（XSD）ファイルがネストされている場合、メイン WSDL の取得後にサブファイルも取得する必要があります。検証は、サブファイル内でネストされているすべての WSDL または XSD を通して再帰処理を続けます。このため、複数レベルのネストがある場合、多くの HTTP 呼び出しで多くの呼び出しを実行して WSDL 文書全体を取得しなければならず、全体的な待ち時間が非常に長くなる可能性があります。

このタイプの待ち時間を軽減するには、WSDL のローカル・コピーをローカル・ファイル・システムまたは HTTP 応答が速い場所に保管します。さらにパフォーマンスを向上するには、すべての include ステートメントを実際のコンテンツに手動で置き換えることによってネストを削除して、このローカル・コピーを「フラット化」できます。

3.2 Integration Designer のベスト・プラクティス

Integration Designer を使用したハイパフォーマンス・ソリューションの開発に関するベスト・プラクティスは、以下のとおりです。

3.2.1 ビジネス・オブジェクト構文解析モードに関する考慮事項

ここでは、BO 構文解析モードを選択する方法とその理由について説明します。このモードには、EAGER 構文解析モードと LAZY 構文解析モードの 2 つのオプションがあります。

概要

WebSphere BPM V7.0.0.1 には、BO LAZY 構文解析モードが導入されています。このモードでは、BO の作成時に BO のプロパティーは、アプリケーションからアクセスされるまで取り込まれません。また、XML 入力の読み取り時に、ストリームは増分解析され、ストリームのプロパティーは、アクセスされるまで実体化しません。一方、EAGER 構文解析モードでは、入力 XML ストリームは即座に全体が解析され（特定のメディエーション・プリミティブの場合に例外あり）、完全なメモリー内データ・オブジェクト表現に実体化します。

LAZY 構文解析では、BPM V7.5 の一部として送達される WebSphere Application Server Feature Pack for Service Component Architecture（SCA）に備わっているサービス・データ・オブジェクト（SDO）の XML カーソル・インターフェース（XCI）実装が使用されます。このモードでは、入力 XML ストリームは、対応する BO の存続期間を通してメモリー内に留まります。カーソル・ベースの XML 解析では、ノードがトラバースされるときにのみ、メモリー内にそれらのノードが作成されます。プロパティーと属性は、アプリケーションが要求したときにのみ、評価されます。この方法での XML ストリームの解析では、EAGER モードで多くのアプリケーションを完全に解析するよりパフォーマンスが高くなります。このパフォーマンスの向上は、アプリケーションの実行中に BO のごく一部しかアクセスされない場合に特に顕著です。

また、LAZY 構文解析では、例えばアウトバウンド・サービス呼び出しの場合など、BO が直列化されるときに XML ストリームが再使用されます。この結果、直列化が速くなります。これは、メモリー内 BO 全体を XML ストリングに直列化する必要がないためです。LAZY 構文解析では、元の入力ストリームの名前空間とは異なる出力ストリームの名前空間が自動的に検出され、修正されます。この直列化の形は、すべてのバインディングおよび Business Process Choreographer（BPC）コンテナーなどの内部 BPM ランタイム・コンポーネントに対して、およびカスタム・コードが BO 直列化を呼び出すときに使用されます。

パフォーマンスをさらに最適化するために、LAZY 構文解析では、BO 内のプロパティーが遅延コピーされる、コピー・オン・ライトと呼ばれるテクノロジーが採用されます。コピー操作が最初に実行されるときに、ターゲット BO は、ソース BO のノード・ツリーを指すだけです。LAZY 構文解析でプロパティーは何もコピーされません。ソース BO とターゲット BO のいずれかの後続の変更によって、BO ツリー内の影響を受けたノードの分割がトリガーされ、必要に応じて BO のプロパティーのみがコピーされます。変更は、対応するノードのローカル・コピーで発生します。コピー・オン・ライトは、アプリケーションに対して透過的です。

解析モードの設定

BPM V7.5 では、Integration Designer V7.5 で新規開発されたアプリケーションの場合、デフォルトの構文解析モードは LAZY モードです。WPS V7.0.0.3 およびこれ以前のリリースから移行されたアプリケーションは、引き続き EAGER 構文解析モードで実行されます。ただし、これらの以前のアプリケーションの場合、Integration Designer を使用して、LAZY 構文解析モードを手動で有効にすることができます。Integration Designer では、XML 構文解析モードを EAGER または LAZY に構成するためのプロパティーを各モジュールおよびライブラリーで使用できます。

図 3-5 は、ライブラリーの作成時に表示される Integration Designer パネルを示しています。この時点で、ライブラリーを EAGER 構文解析モジュールと LAZY 構文解析モジュールのい

ずれに組み込むか、または両方に組み込むかを指定できます。デフォルトは、ライブラリーの最初の作成時に LAZY 構文解析に設定されます。作成するライブラリーに対して両方の構文解析モードを選択した場合は、ライブラリーが配置されるモジュールによって、実行時の実際の構文解析モードが決まります。

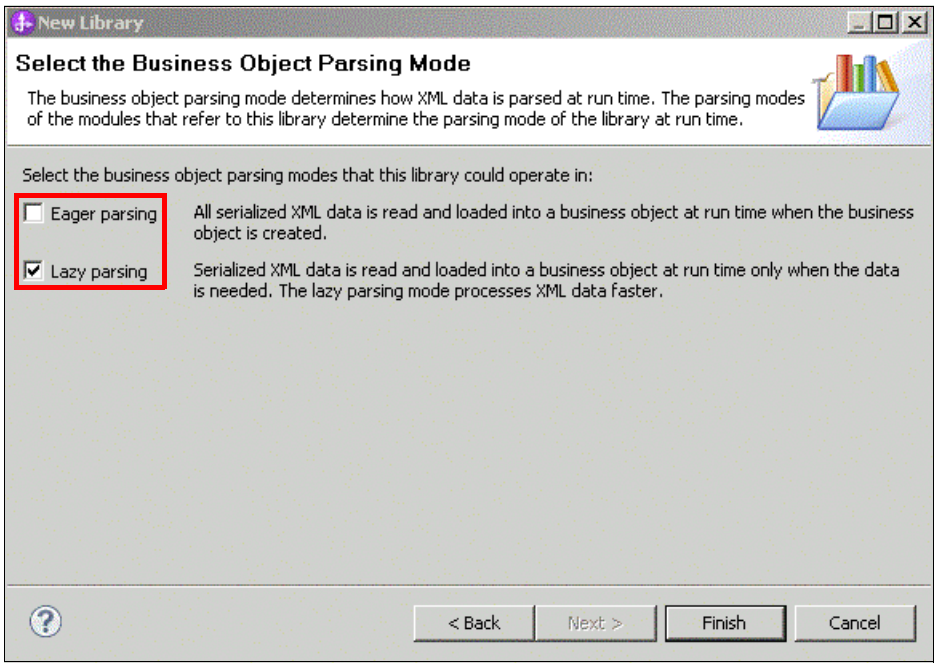

図 3-5　新規ライブラリーの場合の LAZY 構文解析

モジュールの作成時には、LAZY と EAGER のいずれかの構文解析モードを選択する必要があります。デフォルトの選択は LAZY 構文解析モードです。図 3-6 は、モジュール作成時の Integration Designer パネルを示しています。

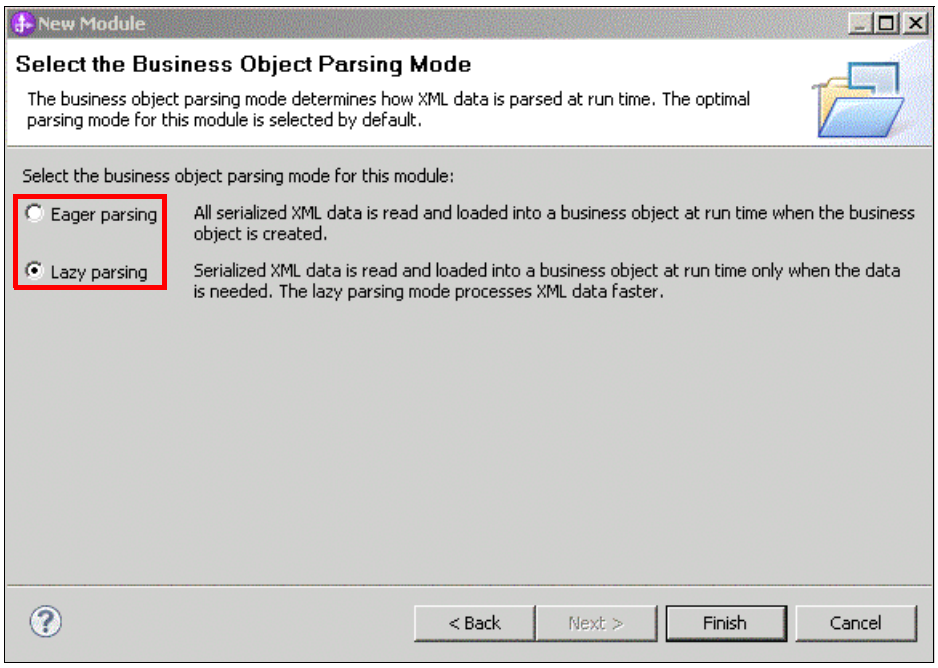

図 3-6　新規モジュールの場合の LAZY 構文解析

BPM V7.5 では、LAZY 構文解析によって、約 3 KB 以上の BO またはメッセージを使用する XML ベースのアプリケーションの場合にハイパフォーマンスを実現できます。LAZY 構文解析のパフォーマンスは通常、その他のシナリオにおける EAGER 構文解析と同じかそれ以上になります。

アプリケーションにおける LAZY 構文解析の利点

ここでは、適切な構文解析モードを選択するための一般的な指針と、一方のモードを選択する場合に従うべきベスト・プラクティスを示します。ただし、アプリケーションの特定の性質に最適なモードを判別するためには、両方の構文解析モードでアプリケーションのベンチマークを行うことをお勧めします。

LAZY 構文解析によってもたらされるパフォーマンスにおける主要な利点は、遅延解析と、アプリケーションによってアクセスされるプロパティーのみが解析されることです。以下のいずれかの特徴を示すアプリケーションは、LAZY 構文解析を利用してパフォーマンスを向上できる可能性があります。数字で示されているように、LAZY 構文解析の使用時は、パフォーマンスの向上がかなり大きなものになる場合があります。

- XML データ・ストリームを使用するアプリケーションのみが、LAZY 構文解析の利点を享受できます。例えば、XML データ・ハンドラーとともに、Web サービス・バインディング、同期または非同期 SCA バインディング、または Java Message Service（JMS）バインディングを使用するアプリケーションは、LAZY 構文解析モードから利点を得られる可能性があります。
- XML データ・ストリームが中規模から大規模のアプリケーションでは、LAZY 構文解析の使用時にパフォーマンス向上が見られる傾向にあります。パフォーマンス上の利点は、XML データ・ストリームのサイズが大きいほど、多くなります。ワークロードを調査したところ、BO が約 3 KB 以上のときに LAZY 構文解析が EAGER モードより優れていました。BO サイズはアプリケーションの構成によって異なるため、これらの調査結果は大まかな目安として使用してください。
- XML データ・ストリームのごく一部にしかアクセスしないアプリケーションでは、LAZY 構文解析の場合にパフォーマンスの向上が見られる傾向にあります。BO へのアクセスは、プロパティー値の読み取り、プロパティー値の更新、または BO へのプロパ

ティーの追加として定義されます。LAZY 構文解析では、データ・ストリームの未変更の部分はシステムを通過でき、必要なリソース使用が最低限に抑えられます。EAGER 構文解析とは対照的に、アプリケーションから BO へのアクセスが少なくなればなるほど、LAZY 構文解析を有効にすることによって達成できるパフォーマンス向上の程度が大きくなります。

- 複数のメディエーション・プリミティブを使用する WebSphere Enterprise Service Bus（ESB）メディエーションでは、LAZY 構文解析の使用時にパフォーマンスの向上が見られる傾向にあります。EAGER 構文解析の使用時は、BO は、次のメディエーション・プリミティブの開始前に直列化され、そのメディエーション・プリミティブの開始後に非直列化されることが多くあります。LAZY 構文解析モードでは、未変更のプロパティーがデータ・ストリームで不必要に直列化されなくなるため、直列化および非直列化ごとのコストが軽減されます。

これらのパラメーターに含まれるアプリケーションは、LAZY 構文解析で利益を得る傾向にあります。これらの属性を持たないアプリケーションでは、パフォーマンスにおける相違はほとんどなく、LAZY 構文解析モードと EAGER 構文解析モードのいずれでも実行できます。例えば、WebSphere アダプターや非 XML データ・ハンドラーなどの非 XML データ・ストリームを解析するアプリケーションは、BPM V7.5 で LAZY 構文解析モードと EAGER 構文解析モードのいずれを選択するかにかかわらず、同じように動作します。この類似性は、BOFactory サービスを使用して BO を直接作成するアプリケーションや小さな XML メッセージ（大まかに、3 KB 以上）を使用するアプリケーションにも当てはまります。

メッセージ・ヘッダーのみにアクセスする、または 1 つのメディエーション・プリミティブのみが含まれる WebSphere ESB メディエーション・フローでは、EAGER 構文解析の使用時にわずかなパフォーマンス向上が見られる場合があります。例えば、ヘッダー・コンテンツに基づいて 1 つのフィルター・メディエーション・ルーティングを使用する ServiceGateway では、パフォーマンスが向上する場合があります。ところが、1 つのフロー・パフォーマンスに複数のメディエーション・プリミティブを追加する処理では、LAZY 構文解析モードの方がパフォーマンスは高くなる傾向にあります。

モジュール間での混合構文解析モード使用の低減

一部のモジュールを EAGER 構文解析を使用するように指定し、その他のモジュールを LAZY 構文解析を使用するように指定する必要がある場合があります。ただし、パフォーマンスを向上するには、互いに頻繁に対話するアプリケーション・モジュールにはこの混合モードは使用しないようにします。例えば、モジュール A が、それぞれのトランザクション上での同期 SCA バインディングを使用してモジュール B を開始する場合、モジュール A とモジュール B の両方で EAGER 構文解析と LAZY 構文解析のどちらか一方を使用するように設定する方が望ましいです。

構文解析モードが異なるモジュール間の対話は、直列化と非直列化を通して実行されるため、パフォーマンスに悪影響を与える可能性があります。混合モードの使用を避けられない場合は、EAGER 構文解析モジュールから LAZY 構文解析モジュールへのアプリケーションの開始の方が、その逆よりも効率的です。EAGER 構文解析モジュールからの出力は XML ストリングに直列化され、LAZY 構文解析を使用するターゲット・モジュールでは、遅延解析の利点を十分に得ながら、効率性を生み出すことができます。

可能な場合の参照による共有ライブラリーの使用

インターフェース、BO、データ・マップ、メディエーション・サブフロー、リレーションシップ、ロール、および Web サービス・ポートの定義を共有して、複数のモジュールのリソースでそれらを使用できるようにしなければならないことがよくあります。これらのリソースはライブラリーに保管されます。ライブラリーは、さまざまな方法でデプロイできます。

- プロセス・アプリケーションを使用したデプロイ
- 依存モジュールを使用したデプロイ
- グローバルにデプロイ

依存関係エディターで選択を行うことができます。ライブラリーをプロセス・アプリケーションと関連付けた場合は、そのアプリケーションをエディターで選択します。ライブラリーは、デプロイ済みのプロセス・アプリケーション内で共有されます。つまり、メモリー内に 1 つのコピーだけが存在します。このタイプのライブラリーを「参照による共有ライブラリー」と呼びます。参照による共有ライブラリーは、
BPM V7.5 の新機能です。

モジュールを使用したライブラリーのデプロイを選択した場合、デプロイメント処理では、モジュールのデプロイ時にモジュールごとにライブラリーのコピーが作成されます。このタイプのライブラリーを「値による共有ライブラリー」と呼びます。モジュールを使用したライブラリーのデプロイは、ライブラリーがプロセス・アプリケーションに関連付けられていない場合のデフォルト設定です。

BPM V7.5 ライブラリー・タイプについて詳しくは、IBM Business Process Manager バージョン 7.5 インフォメーション・センターの『ライブラリー』を参照してください（全プラットフォーム対象）。

http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r5mx/index.jsp?topic=%2Fcom.ibm.wbpm.wid.main.doc%2Fnewapp%2Ftopics%2Fclibrary.html

LAZY 構文解析は、参照による共有ライブラリーを使用するために最適化されます。BO が同期 SCA サービス呼び出しのパラメーターとしてモジュール間を移動するとき、ソース・モジュールとターゲット・モジュールで参照による共有ライブラリーを通して BO スキーマを共有している場合は、LAZY コピーが使用されます。参照による共有の最適化がないと、呼び出し元によって BO が直列化され、呼び出し先によって BO が非直列化されます。

LAZY 構文解析と EAGER 構文解析の動作の相違点

元は EAGER 構文解析を使用して開発されたアプリケーションを LAZY 構文解析を使用するよう変更する場合は、各モードの動作における相違点を理解しておく必要があります。これらの相違点についてはここに概要を示しています。また、1 つのアプリケーションを LAZY 構文解析モードと EAGER 構文解析モード間で切り替えることを計画している場合も、各モード間の動作の相違点を考慮してください。

エラー・ハンドリング

解析対象の XML バイト・ストリームの形式が不適切な場合、解析例外が発生します。

- EAGER 構文解析の場合、BO がインバウンド XML ストリームから解析されるとすぐに例外が発生します。
- LAZY 構文解析の場合、アプリケーションが BO プロパティーにアクセスして、形式が不適切な XML の部分を解析したときに解析例外が潜在的に発生します。

いずれの構文解析モードの場合でも、形式が不適切な XML を適切に処理するには、次のいずれかのオプションを選択します。

- アプリケーション・エッジにメディエーション・フロー・コンポーネントをデプロイして、インバウンド XML を検証します。
- BO のプロパティーがアクセスされるポイントに LAZY エラー検出ロジックを作成します。

例外スタックおよび例外メッセージ

EAGER 構文解析と LAZY 構文解析では、基本となる実装が異なるため、BO プログラミング・インターフェースおよびサービスによって生成されるスタック・トレースは同一の例外クラス名を持ちます。ただし、スタック・トレースに同じ例外メッセージまたは実装固有の例外クラスのラップされたセットが含まれない場合があります。

XML 直列化形式

LAZY 構文解析では、直列化の際にインバウンド・データ・ストリームから未変更の XML をアウトバウンド・データ・ストリームにコピーしようとする高速直列化の最適化を実現できます。高速直列化によってパフォーマンスが向上しますが、LAZY 構文解析で BO 全体が

更新されていた場合、または EAGER 構文解析が使用されていた場合に、アウトバウンド XML データ・ストリームの直列化の結果としての形式が構文上異なる場合があります。

出力 XML の直列化形式は、これらの場合に正確には構文上、等しくならない場合があります。ただし、XML データ・ストリームのセマンティクスは、構文解析モードにかかわらず等しいため、結果としての XML をセマンティクスが等しく、異なる構文解析モードで実行されているアプリケーション間で安全に渡すことができます。

ビジネス・オブジェクト・インスタンス・バリデーター

LAZY 構文解析インスタンス・バリデーターでは、特にプロパティー値のファセット検証において、BO の検証精度を向上できます。精度におけるこれらの向上によって、LAZY 構文解析インスタンス・バリデーターでは、EAGER 構文解析モードでは検出されない機能上の問題がキャッチされ、より詳細なエラー・メッセージが提供されます。

XSL スタイル・シートのマイグレーション

WebSphere Integration Developer V7.0.0.1 またはこれ以前を使用して作成されたアプリケーションは、Integration Designer V7.5 を使用してマイグレーションされ、再ビルドされます。XSLT（Extensible Stylesheet Language Transformation）スタイル・シートは、関連付けられているマップ・ファイルから再生成されます。Integration Designer V7.5 を使用してビルドされた XSL（Extensible Style Sheet Language）スタイル・シートには、空白ディレクティブもインデント・ディレクティブも含まれなくなりました。この変更によって、LAZY 構文解析モードでのパフォーマンスが向上します。ただし、XSLT プリミティブが、手動編集されたスタイル・シートを直接参照している場合、このスタイル・シートは、アプリケーションのビルド時に再生成されません。この場合、空白ディレクティブ（<xsl:strip-space elements="*"/>）がある場合はこれを削除し、必要ない限りインデントが無効（indent="no"）になっているようにすることにより、パフォーマンスが向上します。

プライベート API

BO プログラミング・モデルでは、BO アプリケーション・プログラミング・インターフェース（API）および一連のサービス・データ・オブジェクト（SDO）API がサポートされます。場合によっては、BO に対してサポートされる API には含まれない、追加の実装固有の API がアプリケーションで使用されることがあります。例えば、アプリケーションで Eclipse モデリング・フレームワーク（EMF）API を使用する場合があります。これらの API は以前の BPM リリースの EAGER 構文解析モードでは動作すると思われますが、BO にアクセスするための API としてはサポートされていません。EMF API は、BO の実装に対してはプライベートと見なされるため、アプリケーションでは使用しないでください。

これらの理由により、プライベート API をアプリケーションから削除して、すべての BO アクセスが、サポートされる API セットを使用して実行されるようにします。

サービス・メッセージ・オブジェクトの Eclipse モデリング・フレームワーク API

メディエーション・フロー・コンポーネントでは、com.ibm.websphere.sibx.smobo パッケージに同梱の Java クラスおよび Java インターフェースを使用してメッセージ・コンテンツを操作できます。LAZY 構文解析の場合も、com.ibm.websphere.sibx.smobo パッケージに含まれる Java インターフェースを使用できます。ただし、Eclipse モデリング・フレームワーク（EMF）クラスおよびインターフェースを直接参照するメソッド、または EMF インターフェースから継承されるメソッドは、失敗する可能性が高いです。また、サービス・メッセージ・オブジェクト（SMO）およびそのコンテンツは、LAZY 構文解析の使用時は EMF オブジェクトにキャストできません。

マイグレーション

BPM V7.5 より前に開発されたアプリケーションではすべて、デフォルトで EAGER 構文解析モードが使用されていました。BPM ランタイム・マイグレーションによってこれらのアプリケーションを移行した場合、これらのアプリケーションは引き続き EAGER 構文解析モードで実行されます。

元は EAGER 構文解析を使用して開発されたアプリケーションで LAZY 構文解析を使用できるようにするには、まず、Integration Designer V7.5 を使用して、アプリケーションの成果物

を移行します。マイグレーションの後に、Integration Designer V7.5 を使用して、LAZY 構文解析を使用するようにアプリケーションを構成します。

Integration Designer で成果物を移行する操作について詳しくは、IBM Business Process Manager インフォメーション・センターの『ソース成果物のマイグレーション』を参照してください。

http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r5mx/index.jsp?topic=%2Fcom.ibm.wbpm.wid.imuc.doc%2Ftopics%2Ftmigsrcartwid.html

構文解析モードの設定について詳しくは、IBM Business Process Manager インフォメーション・センターの『モジュールおよびライブラリーのビジネス・オブジェクト構文解析モードの構成』を参照してください。

http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r5mx/index.jsp?topic=%2Fcom.ibm.wbpm.wid.main.doc%2Fnewapp%2Ftopics%2Ftconfigbo.html

3.2.2 サービス・コンポーネント・アーキテクチャーに関する考慮事項

ここでは、BMP パフォーマンスを向上するための SCA オーサリングに関する考慮事項について説明します。

ServiceManager.locateService() の結果のキャッシュ

SCA サービスを検索する Java コードを Java コンポーネントまたは Java スニペット内で記述する場合、サービスの検索は比較的コスト高の操作であるため、後で使用できるように結果をキャッシュすることを検討してください。Integration Designer で生成されたコードでは locateService の結果がキャッシュされることはないため、編集が必要です。

適切な場合のサービス・コンポーネント・アーキテクチャー・モジュール数の軽減

BPM V7.5 コンポーネントをデプロイメントのためにモジュールにアセンブルする場合、多くの要素が関与します。パフォーマンスは主要な要素の 1 つですが、保守容易性、バージョン要件、およびモジュール所有権も考慮する必要があります。また、モジュール数を増やすと、サーバーおよびノードにわたる分散機能を向上できます。モジュール化にもコストがかかることを認識しておくことが重要です。複数のコンポーネントを 1 つのサーバー・インスタンスに一緒に配置する場合、パフォーマンスを向上するには、それらのコンポーネントを 1 つのモジュール内にパッケージ化することが最適です。

ローカル・モジュールでの同期サービス・コンポーネント・アーキテクチャー・バインディングの使用

モジュール間呼び出しの場合、モジュールをローカル（つまり、同一サーバー JVM 内）にデプロイする可能性が高いときは、同期 SCA バインディングの使用をお勧めします。このバインディングは、モジュールの局所性に対して最適化されるため、他のバインディングよりパフォーマンスが高くなるからです。同期 SCA では、異なる BPM V7.5 サーバー・インスタンスのモジュール間で呼び出しが行われる場合は、他のバインディングと同じだけのコストがかかります。

マルチスレッド・サービス・コンポーネント・アーキテクチャー・クライアントを使用した並行性の実現

ローカルで（つまり、同一サーバー JVM の呼び出し元から）開始される同期コンポーネントは、呼び出し元のスレッドのコンテキストで実行されます。このため、呼び出し元では、必要に応じて、マルチスレッドの形で並行性を備える必要があります。

適切なレベルでのサービス品質修飾子の追加

ビジネス・オブジェクト・インスタンス検証などのサービス品質（QoS）修飾子を、インターフェース・レベルまたはインターフェース内の操作レベルで追加できます。QoS 修飾子

には追加のプロセッサー使用が関連付けられるため、インターフェースのすべての操作に必要というわけではない場合は、インターフェース・レベルでは修飾子を適用しないでください。

3.2.3 Business Process Execution Language ビジネス・プロセスに関する考慮事項

ここでは、BPEL ビジネス・プロセスに固有のパフォーマンスのオーサリングに関する考慮事項を示します。

ビジネス・プロセスのアクティビティーに関するモデリングのベスト・プラクティス

パフォーマンスを最適にするには、BPEL ビジネス・プロセスをモデリングする際に以下の指針を使用してください。

- Business Process Choreographer データベース（BPEDB）でイベントをログに記録する必要がある場合は、**「Business Process のみの監査ロギング・プロパティー」**設定を使用します。このプロパティーは、アクティビティー・レベルまたはプロセス・レベルで設定できます。これをプロセス・レベルで設定した場合、その設定はすべてのアクティビティーで継承されます。
- 長期実行プロセスの場合は、Integration Designer で、**「プロパティー」→「サーバー」**をクリックして、プロセスと個々の BPEL アクティビティーの両方に対して **「ビジネス関連データのパーシスタンスおよび照会を可能にする」**をクリアします。このフラグを有効にすると、対象のアクティビティーの実行詳細が BPC データベースに保管されるようになり、データベースの負荷が増し、各プロセス・インスタンスに保管されるデータの量が増えます。この設定は、特定の情報を後で取得する必要がある場合にのみ使用してください。
- 長期実行プロセスの場合は、通常、すべてのアクティビティーに対して**「参加」**を設定することによって、スループット・パフォーマンスが最適になります。詳しくは、38 ページの『非同期ターゲットの両方向同期呼び出しの回避』を参照してください。
- ビジネス・プロセス（プロセスの管理者など）、start アクティビティー、および receive アクティビティーでヒューマン・タスクを指定できます。ヒューマン・タスクは、必要な場合にのみ指定します。複数のユーザーが関与する場合は、グループ・メンバー用の個々の作業項目（担当者割り当て基準 : グループ・メンバー）ではなく、グループ作業項目（担当者割り当て基準 : グループ）を使用します。

長期実行プロセスの両方向同期呼び出しの回避

長期実行ビジネス・プロセス・コンポーネントの設計時には、両方向（要求 / 応答）インターフェースの呼び出し元で、同期セマンティクスが使用されないようにします。これは、この機能によって呼び出し元のリソース（スレッドやトランザクションなど）が、プロセスが完了するまで停止するためです。代わりに、このようなプロセスは、応答を必要としない、非同期呼び出し、または片方向の同期呼び出しによって開始します。また、長期実行ビジネス・プロセスの両方向インターフェースを同期的に呼び出すと、例外の発生時に処理が難しくなります。割り込み不可能なプロセスが、両方向の要求 / 応答セマンティクスを使用して、長期実行プロセスを呼び出し、長期実行プロセスの完了後、呼び出し元のトランザクションがコミットされる前にサーバーに障害が発生したと仮定します。この結果、以下のようになります。

- 呼び出し元が永続メッセージによって開始されていた場合、サーバーの再始動時に、呼び出し元のトランザクションがロールバックされ、再試行されます。ただし、サーバー上の長期実行プロセスの実行結果は、サーバーでの障害発生前にコミットされていたため、ロールバックされません。この結果、サーバー上の長期実行プロセスは 2 回実行されます。手動で修正しない限り、この複製によって、アプリケーションに機能上の問題が発生します。

- 呼び出し元が永続メッセージでは開始されず、長期実行プロセスの応答がまだ送信されていなかった場合は、プロセスは障害が発生したイベント・キューに入る結果となります。

Business Process Execution Language の変数およびビジネス・オブジェクトの数とサイズの最小化

BPEL の変数および BO を定義する際には、以下の指針に従ってください。

- 可能な限り少数の変数を使用して、使用する BO のサイズと数を最小限に抑えます。長期実行プロセスでは、各コミットで、変更済み変数がデータベースに保存されるため (コンテキストを保存するため)、複数の変数または大きな BO ではこのプロセスがコスト高になります。モニター・イベントの発行時には、小さい BO の方が処理がより効率的でもあります。
- 変数をデータ型変数として指定します。このように指定することによって、ランタイム・パフォーマンスが向上します。
- 変換（マップまたは割り当て）を使用して、ビジネス・ロジックに必要なフィールドのみをマッピングすることによって、より小さな BO を作成します。

3.2.4 ヒューマン・タスクに関する考慮事項

BPEL ビジネス・プロセス向けのヒューマン・タスクを開発する際には、以下の指針に従います。

- グループ・メンバー用の個々の作業項目（担当者割り当て基準 : グループ・メンバー）ではなく、大きなグループ用のグループ作業項目（担当者割り当て基準 : グループ）を使用します。
- 可能な場合はカスタム・プロパティーではなく、タスク・オブジェクトのネイティブ・プロパティーを使用します。例えば、カスタム・プロパティーの priority を作成するのではなく、**priority** フィールドを使用します。
- タスクがページ・フローに含まれない場合は、トランザクション動作を**「事後コミット」**に設定します。この設定によって、タスク完了時の API 呼び出しの応答時間が向上します。

3.2.5 ビジネス・プロセスおよびヒューマン・タスク・クライアントに関する考慮事項

以下に、効果的な BPEL ビジネス・プロセスおよびヒューマン・タスク・クライアントの開発に関する考慮事項の詳細を示します。

- タスク詳細およびプロセス詳細を提供する API（htm.getTask() など）を頻繁に呼び出さないでください。これらのメソッドは、例えば単一タスクのタスク詳細を示す場合など、必要なときにのみ使用してください。
- 1 つのクライアント・トランザクションで実行する作業量が多くなりすぎないようにします。
  - 通常、サーブレット・アプリケーションでは、グローバル・トランザクションは使用できません。サーブレットが Human Task Manager（HTM）API および Business Flow Manager（BFM）API を直接呼び出す場合は通常、トランザクション・サイズを検討する必要はありません。
  - Enterprise JavaBeans（EJB）アプリケーションでは、トランザクションにかかる時間が長くなりすぎないようにします。長期実行トランザクションでは、データベース内に長期にわたるロックが作成され、これによって他のアプリケーションおよびクライアントが処理を続行できなくなります。
- ニーズに最適なプロトコルを選択します。

- J2EE 環境では、HTM API および BFM EJB API を使用します。クライアント・アプリケーションが BPM V7.5 サーバーで実行されている場合は、ローカル EJB インターフェースを使用します。
- Web 2.0 アプリケーションでは、REST API を使用します。
- プロセス・コンテナーに対してリモートに実行されるアプリケーションでは、Web サービス API が 1 つのオプションです。

ページ・フロー・パターンに従うクライアントは、以下の問題を考慮する必要があります。

- 可能な場合は、completeAndClaimSuccessor() API を使用します。この API の使用では、応答時間が最適化されます。
- 次に使用可能なタスクをユーザーに割り当てるアプリケーションでは、Human Task Manager EJB インターフェースで **claim(String queryTableName, …)** メソッドを使用できます。このメソッドは、パフォーマンスが最適化されたメカニズムを実装して、要求の競合を処理します。
- システム上の負荷が増すと、非同期サービスの応答時間が長くなるため、ページ・フローの 2 つのステップ間に非同期呼び出しを挿入しないでください。
- 長期実行サブプロセスは非同期メッセージングを使用して開始されるため、可能な場合は、長期実行サブプロセスをページ・フローの 2 つのステップ間で開始しないでください。

タスク・リストおよびプロセス・リストを表示するクライアントでは、以下の要素を考慮する必要があります。

- タスク・リスト照会およびプロセス・リスト照会用の照会テーブルを使用します。
- タスク・リストまたはプロセス・リストに表示されるタスクをループし、各オブジェクトに対して追加のリモート呼び出しを実行することは避けてください。このような処理を行うと、アプリケーションで適切な応答時間および適切なスケーラビリティーが提供されなくなります。
- タスク・リストおよびプロセス・リストの取得時に、すべての情報が 1 つの照会テーブルから取得されるようにアプリケーションを設計します。例えば、タスク・リストまたはプロセス・リストの作成用の入力メッセージを取得するために複数の呼び出しを行わないようにします。

3.2.6 トランザクション上の考慮事項

BPM V7.5 プラットフォームの利点の 1 つは、トランザクション動作の指定を正確に制御できることです。プロセスまたはメディエーションのアセンブリーをモデリングする際、モデル管理者は、アプリケーションのニーズによって指示されているように目的のトランザクション境界を慎重に設計することをお勧めします。これは、システム・リソースの観点から言えば、境界はコストが高いためです。このセクションの目的は、モデル管理者が不必要なトランザクション境界を設計しないように導くことです。以下に、いくつかの指針の詳細を示します。

- 特定の使用シナリオのスループットは、そのシナリオでトラバースされたトランザクション境界の数に反比例するため、トランザクションが少なければ少ないほど、高速になります。
- ユーザー駆動型シナリオでは、応答時間を向上するために、スループットを犠牲にしてさえも、より細分化されたトランザクション境界が必要な場合があります。
- トランザクションは、同期呼び出し間にわたって伝播できますが、非同期呼び出し間にわたって伝播することはできません。
- 両方向非同期ターゲットに同期呼び出しを行わないようにします。呼び出し元トランザクションの障害リカバリーで問題が生じる可能性があります。

Service Component Architecture トランザクション修飾子の利用

SCA アセンブリーでは、トランザクションがコンポーネントにわたって伝搬できるようにすることで、トランザクション境界の数を軽減できます。トランザクション境界の数を削減したいコンポーネントのペアに対して、以下の設定を使用することをお勧めします。

- **SuspendTransaction=** false（呼び出し元コンポーネントのリファレンスに対する設定）
- **joinTransaction=** true（呼び出し先コンポーネントのインターフェースに対する設定）
- **Transaction=** any|global（両方のコンポーネントの実装に対する設定）

これらの設定では、このようなチェーンの最初のコンポーネントがグローバル・トランザクションを開始するか、グローバル・トランザクションに参加することを前提としています。

非同期ターゲットの両方向同期呼び出しの回避

ターゲット・コンポーネントを非同期に開始する必要があり、そのインターフェースが両方向の要求 / 応答スタイルである場合、同期 SCA 呼び出しを使用してそのターゲットを安全に開始することはできません。呼び出し元は、ターゲットに要求を送信した後、ターゲットからの応答を待機します。要求を受信すると、非同期ターゲットは新規トランザクションを開始します。要求を処理すると、ターゲットは応答キューを使用して、呼び出し元に応答を非同期に返します。呼び出し元が要求を正常に送信した後、応答を受信する前にシステム障害が発生した場合、呼び出し元のトランザクションはロールバックされ、再試行されます。この結果、ターゲットが再度開始されます。

長期実行プロセスのアクティビティーでのトランザクション属性の利用

SCA 修飾子は、コンポーネント・レベルのトランザクション動作を制御しますが、長期実行ビジネス・プロセスにおけるトランザクションの追加考慮事項によって、アクティビティーが複数のトランザクションで実行されるようになる場合があります。Java スニペット・アクティビティー、ヒューマン・タスク・アクティビティー、および start アクティビティーのトランザクション動作の設定を使用して、これらのトランザクションのスコープおよびトランザクションの数を変更できます。これらの設定について詳しくは、WebSphere Process Server for Multiplatforms バージョン 7.0 インフォメーション・センターの『ビジネス・プロセスのトランザクションの振る舞い』を参照してください。

http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r0mx/index.jsp?topic=/com.ibm.websphere.bpc.doc/doc/bpc/cprocess_transaction.html

トランザクション動作の設定については 4 つの選択肢があります。

- **事前コミット**
- **事後コミット**
- **参加**
- **所有が必要**

新規ブランチを開始する並列アクティビティーでは、並列性を確保するために、**「事前コミット」**設定を使用します。インフォメーション・センターに記載されているように、他の制約を考慮する必要があります。

ヒューマン・ユーザーに対する応答性を向上するには、インライン・ヒューマン・タスクに**「事後コミット」**を使用します。ヒューマン・ユーザーがタスク完了を発行した場合、その処理を行うスレッド / トランザクションを使用して、プロセス・フロー内のヒューマン・タスク・アクティビティーのナビゲーションが再開されます。ユーザーのタスク完了処理は、プロセス・エンジンがトランザクションをコミットするまで完了しません。

WebSphere Process Server（WPS）6.2.0 以降は、BPEL フローの receive および pick アクティビティーでは、それぞれ固有のトランザクション動作のプロパティー値を定義できるようになりました。設定しない場合、receive または pick アクティビティー開始のデフォルト値は**「事後コミット」**です。可能な場合は、**「参加」**の使用を検討してください。この動作の方が、パフォーマンスが高くなります。

「参加」設定を使用した場合、コミットが遅延され、ユーザーに対する応答時間が強制的に長くなります。**「参加」**設定のみが、新規トランザクション境界を必要としません。他の 3 つの設定では、アクティビティーの実行前とアクティビティーの実行後のいずれかまたは両方に、プロセス・フロー・コンテナーが新規トランザクションを開始する必要があります。

通常、**「参加」**属性によってスループットが最適になるため、可能な場合はこれを使用してください。このことは、同期アクティビティーと非同期アクティビティーの両方に当てはまります。両方向非同期の場合、呼び出しトランザクションは常に要求の送信後にコミットされることを理解しておくことが重要です。**「参加」**設定は、プロセス・エンジンが応答のために開始したトランザクションを参照します。これが設定されていると、次のアクティビティーは同じトランザクションで続行できるようになります。

特殊なケースとして、**「参加」**以外のトランザクション設定が望ましいことがあります。38 ページの『長期実行プロセスのアクティビティーでのトランザクション属性の利用』で示しているように、WebSphere Process Server for Multiplatforms バージョン 7.0 インフォメーション・センターの『ビジネス・プロセスのトランザクションの振る舞い』を参照してください。

「所有が必要」では、新規トランザクションを開始する必要があります。

3.2.7 呼び出しスタイルに関する考慮事項

ここでは、呼び出しスタイルに関する考慮事項を示します。

非同期の慎重な使用

コンポーネントとモジュールを互いに同期的または非同期的にワイヤーする場合があります。この対話スタイルの選択によって、パフォーマンスに深刻な影響を与える可能性があります。この選択を行う場合は、注意が必要です。

可能な場合に、「優先相互作用形式」を「同期」に設定

多くの BPM V7.5 サーバー・コンポーネント・タイプ（インターフェース・マップやビジネス・ルールなど）では、ターゲット・インターフェースの**「優先相互作用形式」**設定に基づいてターゲット・コンポーネントが始動されます。同期コンポーネント間呼び出しの方がパフォーマンスが高いため、可能な場合は**「優先相互作用形式」**を**「同期」**に設定することをお勧めします。この設定を**「非同期」**に変更するのは、特殊なケースのみにしてください。そのようなケースとは、長期実行ビジネス・プロセスを開始する場合などで、もっと一般的に言えば、ターゲット・コンポーネントに非同期呼び出しが必要な場合です。

WebSphere Integration Developer V6.2（現在の Integration Designer）以降、新規コンポーネントがアセンブリー・ダイアグラムに追加されると、その**「優先相互作用形式」**は、追加されたコンポーネントに基づいて**「同期」**、**「非同期」**、または**「任意」**に設定されるようになりました。前のリリースの Integration Designer（当時は WebSphere Integration Developer と呼ばれていた）では、ユーザーが明示的に変更しない限り、**「優先相互作用形式」**はデフォルトの初期設定として**「任意」**に設定されていました。コンポーネントの**「優先相互作用形式」**を**「任意」**に設定した場合、コンポーネントが開始される方法は、呼び出し元のコンテキストによって決まります。呼び出し元が長期実行ビジネス・プロセスの場合、**「優先相互作用形式」**設定の**「任意」**は、非同期として扱われます。呼び出し元が割り込み不可能なビジネス・フローの場合、**「優先相互作用形式」**設定の**「任意」**は、同期として扱われます。

プロセスの呼び出しロジックについて詳しくは、38 ページの『長期実行プロセスのアクティビティーでのトランザクション属性の利用』を参照してください。

以下に、呼び出しスタイルに関するその他の考慮事項の詳細を示します。

- インターフェースの**「優先相互作用形式」**を**「非同期」**に設定する場合は、ダウンストリームの関連事項を認識しておくことが重要です。ダウンストリームを開始するコンポーネントではすべて、**「優先相互作用形式」**が**「同期」**に明示的に設定されていない限り、非同期対話スタイルが継承されます。

- モジュールに対する入力境界では、IBM WebSphere MQ、JMS、または Java EE コネクター・アーキテクチャー（非同期送達セットが備わっている）などの非同期トランスポートを表すエクスポートによって、対話スタイルが**「非同期」**に設定されます。この設定により、**「優先相互作用形式」**が**「任意」**の場合にダウンストリーム呼び出しが非同期になることがあります。
- SCA インポートの場合、**「優先相互作用形式」**を使用して、モジュール間呼び出しを**「同期」**と**「非同期」**のいずれにするかを指定できます。
- WebSphere MQ や JMS などの非同期トランスポートを表すその他のインポートの場合は、**「優先相互作用形式」**を**「非同期」**に設定する必要はありません。そのように設定すると、呼び出しモジュールとトランスポートの呼び出しとの間で、不必要な非同期ホップを招くことになります。

モジュール内のコンポーネント間非同期呼び出しの最小化

非同期呼び出しは、トランザクション、パーシスタンス、およびリカバリー性などのサービス品質セットを充実したものにすることを目的としています。このため、非同期呼び出しは、ターゲットまでのフル・メッセージ・ホップと考えてください。呼び出しの目的のターゲットが同じモジュール内にある場合、同期呼び出しによってパフォーマンスが向上します。

一部のサービス品質（イベント順序付けやストア・アンド・フォワードなど）は、非同期 SCA 呼び出しにのみ関連付けることができます。これらのサービス品質を設定する場合は、非同期呼び出しによるパフォーマンス上の影響を考慮してください。

ファンアウト・ブロックまたはファンイン・ブロックでの同期サービスの非同期呼び出しの回避

決定的な必要性があり、このスタイルの呼び出しについてのパフォーマンス以外の関連事項を理解している場合を除き、同期バインディングを伴うサービス（Web サービスなど）に対して非同期（据え置き応答対話）サービス呼び出しを選択しないでください。

同期サービスを非同期に呼び出す場合のパフォーマンス上の関連事項とは別に、信頼性およびトランザクションの側面を考慮する必要があります。通常、非同期コールアウトは、べき等の照会タイプのサービスの場合にのみ使用します。サービスが 1 回だけ呼び出されることを保証する必要がある場合は、非同期呼び出しを使用しないでください。メディエーション・フローで非同期コールアウトを使用する場合の機能上の適用可能性について完全な指針を提供することは本書の範囲を超えています。

詳しくは、Integration Designer ヘルプ資料および以下のアドレスからアクセス可能な BPM V7.5 インフォメーション・センターを参照してください。

http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r5mx/index.jsp

非同期コールアウトがご使用の構成に適用可能であると仮定すれば、このスタイルでサービスを開始するパフォーマンス上の理由があるかもしれません。ただし、非同期処理では、この方法でサービスを呼び出すことによって追加のメッセージング・リソースが作成されるため、プロセッサー・サイクルの観点において本質的によりコスト高になります。

同期モードまたは非同期モードの選択において、その他の操作上の考慮事項が適用される場合があります。例えば、非同期呼び出しでは、サービス統合バス・メッセージング・インフラストラクチャーが使用されます。つまり、パーシスタンスを確保するためにデータベースが使用されます。同期呼び出しは、JVM ヒープ・サイズおよびスレッド・プールの基本的なチューニングで適切に実行されますが、非同期呼び出しの場合は、SCA 成果物のレビューとチューニングが必要です。この要件には、SCA メッセージング・エンジン（ 54 ページの 4.3.7、「メッセージング・エンジンのプロパティー」を参照）、データ・ソース（ 54 ページの 4.3.6、「Java Database Connectivity データ・ソースのパラメーター」を参照）、およびデータベース自体のチューニングが含まれます。データ・ソースについては、本書に記載されている JMS バインディングに関するチューニングでの考慮事項が同じであるため、これを指針として参照できます。

待ち時間の長い同期サービスが複数呼び出される場合は、非同期呼び出しによってメディエーション・フローの全体的な応答時間を削減できますが、代わりに個々のサービス呼び出しの内部応答時間が長くなってしまいます。この全体的な応答時間の削減と内部応答時間の増大では、非同期コールアウトが、以下の条件の下にフローのファンアウト・セクションにおいて並列待機で構成されていることを前提としています。

- すべてまたはいくつかのメッセージの送出後に非同期応答を確認するようファンアウトを構成した場合に配列の反復が定義されている
- 追加のワイヤーまたはフロー・オーダー・プリミティブが定義されている（デフォルト）

メディエーション・フローのファンアウト・セクションに多くのサービスが存在する場合に、これらのサービスを同期的に呼び出すと、全体的な応答時間が個々のサービス応答時間の合計に等しくなります。

並列待機が構成されている状態でサービスを非同期的に呼び出すと、応答時間が、WebSphere ESB の個々のサービスの最大応答時間以上になります。この応答時間には、メッセージング・エンジン・キュー内の残りのサービス・コールアウト応答が WebSphere ESB によって処理されるときにかかる時間の合計も含まれます。

ファンアウト・ブロックまたはファンイン・ブロックの場合は両方とも、サービスの呼び出し前または呼び出し後のプリミティブ処理時間を追加する必要があります。

メディエーション・フローのファンアウト・セクションまたはファンイン・セクションでサービスを非同期的に呼び出す際に全体的な応答時間を最適化するには、予期される待ち時間がわかっている場合はその順に（最長待ち時間のものが最初）サービスを開始します。

並列処理と追加の非同期処理とのトレードオフを考慮する必要があります。非同期処理が適切であるかどうかは、処理するメッセージのサイズ、ターゲット・サービスの待ち時間、開始するサービスの数、およびサービス・レベル・アグリーメント（SLA）に示されている応答時間に関する要件によります。非同期呼び出しの使用を検討している場合は、待ち時間の長いサービスが含まれるファンアウトなどのメディエーション・フローでパフォーマンス評価を実行することをお勧めします。

サービス参照のデフォルトのサービス品質は**「保証パーシスタント」**です。この設定を**「ベストエフォート」**（非パーシスタント）に変更することによって、非同期処理時間を大きく削減できます。この変更では、パーシスタント・ストアへの入出力が除去されますが、アプリケーションでは要求メッセージまたは応答メッセージの損失の可能性を容認しておく必要があります。サービス統合バスへのこのレベルの信頼性では、負荷のかかったメッセージが廃棄される場合や、チューニングが必要な場合があります。

3.2.8 ラージ・オブジェクトに関する考慮事項

ここでは、ラージ・オブジェクト使用時のパフォーマンス（特にメモリー使用量）のベスト・プラクティスを示します。非常に大きなオブジェクトによって、特に 32 ビット JVM の場合に、Java ヒープ使用量に大きな負荷がかかります。64 ビット JVM の場合は大きなヒープ・サイズを構成できるため、この問題が生じる可能性はもっと低くなりますが、それでもスループット、応答時間、および全体的なシステム・パフォーマンスは過剰なメモリー使用によって影響を受けることがあります。以下のベスト・プラクティスに従って、メモリー負荷を軽減し、OutOfMemory 例外を回避し、全体的なシステム・パフォーマンスを向上してください。

リソースの遅延クリーンアップの回避

リソースの遅延クリーンアップによって、ラージ・オブジェクトの処理時に必要なライブ・セットが多くなります。リソースをクリーンアップできる場合は（例えば、不要になったオブジェクト参照の除去によって）、すぐにそれを実行することが実際的です。

ラージ・ビジネス・オブジェクトの処理時のトレースの回避

トレースおよびロギングによってメモリー・リソース消費が大幅に増加する場合があります。通常のトレース・アクティビティーでは、BO ペイロードがダンプされます。ラージ BO のストリング表現を作成することによって、多くのラージおよびスモール Java オブジェクトが Java ヒープに割り振られることになる場合があります。このため、実稼働環境でラージ BO ペイロードを処理する場合は、トレースを有効にしないようにしてください。

また、条件付き guard ステートメントの外部では、トレース・メッセージを作成しないようにしてください。例えば、例 3-1 のサンプル・コードでは、トレースが無効になっている場合でも、ラージ・ストリング・オブジェクトが作成されます。

例 3-1　ラージ・ストリング・オブジェクトの作成

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

このパターンは常に非効率的ですが、BO サイズが大きい場合はパフォーマンスへの影響がさらに大きくなります。トレースが無効になっている場合に不必要に BO を作成しないようにするには、以下のように if ステートメント内の String 構造を移動します。

```
if (tracing_on) SystemIn(bo.toString());
```

バッファー・ダブリング・コードの回避

入力に基づいてキャパシティーを拡張する Java データ構造（StringBuffer や ByteArrayOutputStream など）を使用する場合のメモリー関連事項を考慮してください。このようなデータ構造では通常、ディスク空き容量が不足したときにキャパシティーが 2 倍になります。このダブリングによって、ラージ・オブジェクトの処理時に大きなメモリー負荷が生じる場合があります。可能な場合は、このようなデータ構造には常に初期サイズを割り当てるようにしてください。

3.2.9 メディエーション・フローに関する考慮事項

ここでは、パフォーマンスを向上するためのメディエーション・フローに関する考慮事項について説明します。

Extensible Stylesheet Language Transformation プリミティブとビジネス・オブジェクト・マップの使用の比較

XSLT プリミティブおよびビジネス・オブジェクト・マップ・プリミティブのいずれかの方法を使用して、メディエーション・フローで変換を実行できます。XSLT 固有の機能が必要でない場合は通常、ビジネス・オブジェクト・マップ・プリミティブを使用する方が処理速度が速いため、適しています。例外として、事実上あまり意味のないメディエーション・フロー・コンポーネントがあり、変換がサービス・メッセージ・オブジェクト（SMO）の /body レベルで実行される場合があります。この場合は、ネイティブ XML バイトを XSLT エンジンに直接渡すことができるため、XSLT の方が速くなります。ネイティブ XML バイトは、XSLT 変換プリミティブがフローの最初にあるか、XSLT 変換プリミティブの前に以下の 1 つまたは複数のプリミティブのみがある場合に使用できます。

- メッセージ・ヘッダー上の経路（Message Filter プリミティブ）
- XSLT プリミティブ（ルートとして /body で変換）
- Xpath ユーザー・プロパティーが指定されていないエンドポイント・ルックアップ
- イベント・エミッター（イベント・ヘッダーのみ）

集約ブロック（ファンアウト / ファンイン）

集約は、WebSphere Enterprise Service Bus（ESB）の重要なパターンの 1 つです。これによって、1 つのインバウンド要求を複数のアウトバウンド・サービス呼び出しにマップできるようになり、それら複数の応答を元の要求に対する 1 つの応答として集約できます。集約設計パターンを使用してメディエーションを開発する前に、考慮が必要なパフォーマンス要素がいくつかあります。以下のサイトで入手可能なベスト・プラクティスに関する

developerWorks のトピック『Aggregation design patterns and performance considerations in WebSphere Enterprise Service Bus V7.5』を参照してください。

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

WebSphere ESB リソースの構成

Integration Designer を使用してリソースを作成する場合、アプリケーション開発者は、事前構成された WebSphere ESB リソースを使用するか、ツールを使用して必要なメディエーション・フロー関連リソースを生成するかを選択する場合があります。両方の方法に、利点と欠点があります。

事前構成されたリソースは、以下の状況で役立ちます。

- 既存のリソースを使用する
- 作成およびチューニングの外部スクリプトを適用する

事前構成されたリソースでは、デプロイメント後の調整も簡単になります。

ツールで作成するリソースは、スクリプトまたは管理コンソールを使用したリソースの作成について追加ニーズがない場合に適しています。ほとんどのパフォーマンス・チューニング・オプションは、今やツールとして公開されているため、変更が可能です。

パフォーマンス・チューニングをビジネス・ロジックから分離することによって、単一スクリプトでの異なるシナリオに対する構成を保守できるようになるため、当社のパフォーマンス・テストでは、事前構成されたリソースを使用しました。

当社のテストでこのパターンに従わなかった唯一のケースが、汎用 JMS バインディングを使用した場合です。このようなシナリオでは、リソースはサード・パーティーの JMS プロバイダー・ソフトウェア（本書ではすべての事例で WebSphere MQ V6.0.2.2）によって既に構成されています。ツールで作成されたリソースは、外部定義されたリソースの検索に使用されました。

3.3 WebSphere InterChange Server のマイグレーションに関する考慮事項

以下に、WebSphere InterChange Server（WICS）から BPM V7.5 へのマイグレーションに関する考慮事項を示します。

- カスタムの IBM WebSphere Business Integration（WBI）アダプターまたは古い WBI アダプターを使用するマイグレーション済みワークロードは、JMS を通して BPM V7.5 と対話することになるため、JCA アダプターの場合より遅くなります。可能な場合は、WBI アダプターの代わりに WebSphere アダプターを使用してください。
- 一部のテクノロジー・アダプター（HTTP サービスや Web サービスなど）は、WebSphere InterChange Server マイグレーション・ウィザードを使用してネイティブの BPM V7.5 SCA バインディングにマイグレーションされ、パフォーマンスが向上します。使用可能な SCA バインディングに自動的にマイグレーションされないアダプターの場合は、SCA バインディングに手動でマイグレーションする開発処理によって、古いアダプターへの依存関係が削除され、パフォーマンスが向上します。
- Integration Designer の WebSphere InterChange Server マイグレーション・ウィザードは、コネクターおよびコラボレーション・モジュールをマージする機能を提供します。このオプションでは、モジュール間の SCA 呼び出しが軽減されることによってパフォーマンスが向上するので、可能な場合はこのオプションを有効にしてください。
- WebSphere InterChange Server コラボレーションは、BPM V7.5 BPEL プロセスにマイグレーションされます。マイグレーションの結果としての BPEL プロセスをさらにカスタマイズして、効率性を高めることができます。

– マイグレーションされた BPEL プロセスでは、デフォルトで補正のサポートが有効になります。マイグレーションされたワークロードで補正を使用しない場合は、このサポートを無効にして、パフォーマンスを向上できます。関連フラグにアクセスするには、Integration Designer でプロセス名を選択し、**「プロパティー」→「詳細」→「補正範囲コンテキストが渡される必要がある」**をクリックします。

– 生成された BPEL フローでは、引き続き WICS API を使用して、BO およびコラボレーション・レベルのタスクが実行されます。ICS API の代わりに BPM API を使用するようにマイグレーション済み BPEL をクリーンアップする開発処理によって、パフォーマンスおよび保守容易性が向上します。

– マイグレーションによって生成された BPEL プロセスを他の成果物に置換できる場合があります。現在、WebSphere InterChange Server コラボレーションはすべて、BPEL プロセスにマイグレーションされます。一部のシナリオでは、その他の BPM V7.5 サーバー成果物（ビジネス・ルールなど）の方が適した選択肢である場合があります。マイグレーションによって生成された BPEL プロセスを調査して、これらのプロセスがご使用のシナリオに最適であることを確認します。

- マイグレーションで生成されたメディエーション・フロー・コンポーネント（MFC）では、MessageLogger 呼び出しを無効にします。Integration Designer V7.5 の WebSphere InterChange Server マイグレーション・ウィザードは、コネクターのマッピング詳細を処理するための MFC を生成します。この MFC には、アプリケーション固有の BO を汎用 BO に変換し、汎用 BO をアプリケーション固有のオブジェクトに変換するマップを呼び出す同期呼び出しおよび非同期呼び出しの処理のためのコードが含まれています。

  生成される MFC には、メッセージをデータベースに記録する組み込み MessageLogger 呼び出しが含まれます。ビジネス・シナリオで必要ない場合はこれらの呼び出しを無効にして、データベースへの書き込みを軽減し、それによってパフォーマンスが向上されます（MessageLogger インスタンスを選択し、「詳細」パネルを選択して、**「使用可能」**チェック・ボックスをクリアします)。

- マイグレーション・ウィザードによって生成される共有ライブラリーを分割することによってメモリー負荷を軽減します。マイグレーション・ウィザードによって、単一共有ライブラリーが作成され、マイグレーションされた BO、マップ、およびリレーションシップがすべてそこに保管されます。すべてのマイグレーション済みモジュールでこのライブラリーがコピーによって共有されます。共有ライブラリーが大きく、多くのモジュールが存在する場合に、この共有によってメモリーの膨張を招くことがあります。解決方法として、機能と使用状況に基づいて共有ライブラリーを複数のライブラリーに手動でリファクタリングし、必要な共有ライブラリーのみを参照するようにモジュールを変更します。

- 元の WebSphere InterChange Server マップに、カスタム・マップ・ステップが多く含まれる場合は、これらのマップ・ステップの再書き込みの開発処理でパフォーマンスが向上します。Integration Designer V7.5 の WebSphere InterChange Server マイグレーション・ウィザードは、BPM V7.5 サーバー・テクノロジーを超えて変換層に ICS API を使用するマップを生成します。BPM V7.5 サーバー API を直接利用することによってこの層を削除すると、変換のコストを削減して、パフォーマンスが向上します。

3.4 オーサリング環境に関する考慮事項

ここでは、一般的にアプリケーション開発者がエンタープライズ・アプリケーションの開発時に直面する、主にアプリケーション・ワークスペースのインポート、ビルド、および公開操作などのアクティビティーのパフォーマンスを向上するための提案を示します。このセクションの焦点は、オーサリング・ツールを使用したハイパフォーマンス・ランタイム・アプリケーションの作成を集中的に取り上げた前のセクションのものとは対照的になっています。

ハードウェア利点の使用

エンタープライズ・アプリケーションのインポートおよびビルドはリソース集中のアクティビティーです。デスクトップ・ハードウェア・アーキテクチャーの近年の改良により、インポート・アクティビティーおよびビルド・アクティビティーの応答性は大きく向上しました。また、入出力集中のアクティビティー（インポートなど）の場合、ディスク・ドライブまたはディスク・サブシステムの高速化により全体的な応答時間が削減されています。

ビジネス・インテグレーション・ライブラリーのモジュール間での共有

BPM V7.5 には、IBM Process Center が備わっており、これによってビジネス・インテグレーション・モジュールをこのバージョンの新機能である BPM Process Application に組み込むことができます。Process Application の利点の 1 つが、Process Application 内のさまざまなモジュール間で、ビジネス・インテグレーション・ライブラリーをモジュールごとに複製するのではなく、共有のスコープまで拡張できることです。この利点によって、以前はモジュール・レベルのスコープだった大きなライブラリーの場合に、複製コピーが不要になったためにサーバー・メモリーを節約できるようになりました。この Process Application レベルのスコープを使用するには、LAZY BO 構文解析モードが必要です。

この Process Application レベルのスコープは、BPM V7.5 の Process Application の新規ライブラリーでデフォルトとなっていますが、既存のライブラリーではモジュール・レベルのスコープが維持されます。可能な場合は常に、これらのライブラリーで Process Application レベルのスコープを使用するよう変更してください。メモリーの節約は、ライブラリーを参照するモジュールの数と相関関係があります。ライブラリーを使用するモジュールの数が多いほど、Process Application レベルのスコープのライブラリーを使用する場合に可能なメモリー節約量が多くなります。

メモリー節約のための Process Application レベルのスコープへのライブラリーの配置

Process Center では、ビジネス・インテグレーション・モジュールをこのバージョンの Integration Designer の新機能である BPM Process Application に組み込むことができます。Process Application の利点の 1 つが、Process Application 内のさまざまなモジュール間で、ビジネス・インテグレーション・ライブラリーをモジュールごとに複製するのではなく、共有のスコープまで拡張できることです。この利点によって、以前はモジュール・レベルのスコープだった大きなライブラリーの場合に、複製コピーが不要になったためにサーバー・メモリーを節約できるようになりました。BPM V7.5 では、この Process Application レベルのスコープを使用するには、LAZY BO 構文解析モードが必要です。

この Process Application レベルのスコープは、BPM V7.5 の Process Application の新規ライブラリーでデフォルトとなっていますが、既存のライブラリーではモジュール・レベルのスコープが維持されます。可能な場合は常に、これらのライブラリーで Process Application レベルのスコープを使用するよう変更してください。メモリーの節約は、ライブラリーを参照するモジュールの数と相関関係があります。ライブラリーを使用するモジュールの数が多いほど、Process Application レベルのスコープのライブラリーを使用する場合に可能なメモリー節約量が多くなります。

3.5 Business Space 開発者に関する考慮事項

Business Space ソリューションの開発時に、開発者がデバッグ、トレース、およびその他のブラウザー機能を実行するためにブラウザーの構成を変更することがよくあります。ただし、これらの構成変更によって、Business Space ソリューションのパフォーマンスに劇的に影響を与える場合があります。ソリューションを実動に展開する前に、またはパフォーマンス測定を行う前に、このセクションを復習して、指針を確認してください。

実動で使用する場合のアドインまたは拡張機能の無効化またはアンインストール

重要な考慮事項の 1 つとして、Internet Explorer（IE）および Firefox の両方に対するアドインまたは拡張機能の使用があります。IE の場合、開発者ツールを有効にすることによってパフォーマンスが劇的に低下します。Firebug および HttpWatch でもある程度パフォーマンスが低下しますが、IE ほどではありません。問題が生じた場合は常に、必要でない限り、アドインまたは拡張機能を無効にするかまたはアンインストールしてください。

実動での Internet Explorer における開発者ツールからの表示モードつまり互換モード設定の回避

IE では、「開発者ツール」を使用して、互換モード、つまり以前のブラウザー・バージョンに等しい表示モードでブラウザーを表示できます。このオプションは、ソリューションの開発やデバッグ（例えば、互換性のテスト用に IE 8 ブラウザーを IE 7 ブラウザーのように動作させる）の際に便利な場合があります。ただし、一般的により新しいバージョンの IE は以前のバージョンよりパフォーマンスが優れているので、これらの設定は使用しないでください。

ブラウザー・ツールの使用

以下の見解は、応答時間測定値の取得、要求数のカウント、およびキャッシュの分析に使用可能なブラウザー・ツールに関連しています。

- Firebug の**「接続」**タブは、要求のタイミングの取得、要求 / 応答ヘッダーの分析、および要求数のカウントの際に便利です。ただし、このタブでは、ブラウザー・キャッシュで満たされた要求がコード 200 応答としてレポートされます。それでもなお、対象の要求がキャッシュされることは、その要求に表示される**「キャッシュ」**タブから判別できます。このタブには、要求がブラウザー・キャッシュで満たされたことが示されます。この結果を別の文書（E メールなど）にコピーして貼り付けた場合、**「キャッシュ」**タブはコピーされません。このため、コード 200 応答によって誤解を招き、キャッシュが実際には有効な場合にそうではないとの誤った結論を導く可能性があります。
- Fiddler も強力なツールで、IE ブラウザーと Firefox ブラウザーの両方をサポートできる利点があります。ただし、Fiddler はブラウザーとサーバー間のプロキシーとして動作し、キャッシュされた要求はブラウザーで内部処理されるため、これらの要求が Fiddler に表示されることはありません。結果レポートがないことによって、ブラウザー・キャッシュで満たされた要求がいずれであるかを判別できなくなりますが、それでも Fiddler はサーバーに実際に送信された要求の分析に役立ちます。
- HttpWatch は、IE ブラウザーと Firefox ブラウザーの両方でサポートされており、Fiddler のような制限事項がありません。結果をスプレッドシートまたは文書に簡単にコピーして貼り付けることができ、キャッシュされた要求を簡単な方法で表示できます。

4

パフォーマンスのチューニングと構成

パフォーマンスを最適化するには、デフォルト設定と異なる内容でシステムを構成する必要があります。この章では、システム・チューニング時に考慮すべきいくつかの領域を列挙しています。これには、Business Process Manager（BPM）製品やシステム内の他の製品（DB2 など）が含まれます。これらの各製品の資料では、パフォーマンス、キャパシティー・プランニング、および構成について説明されているとともに、さまざまな動作環境で高パフォーマンスを実現するための指針が記載されています。

システム管理者は、多数の構成パラメーターを使用できます。この章では、パフォーマンスに影響を与えることが確認されているいくつかの特定のパラメーターを紹介しますが、すべての使用可能なパラメーターを扱っているわけではありません。構成パラメーターと使用可能な設定の完全なリストについては、該当する製品資料を参照してください。

この章の最初のセクションでは、導入済みシステムのチューニング手法を説明します。その次のセクションに記載された基本的なチューニング・チェックリストでは、主要なコンポーネントとそれらに関するチューニング概念を列挙しています。サブセクションでは、チューニングをさらに詳しく解説して、いくつかのチューニング・パラメーターとそれらの推奨設定（適宜）を記載しており、システムの主要領域を対象にした高度なチューニング指針を示しています。多くのチューニング・パラメーターの代表値は、93 ページの第 5 章、「初期構成設定」に示しています。

重要 : この章で示している指針に従っても満足できるパフォーマンスが得られるという保証はありませんが、これらのパラメーターを不適切な値に設定すると、パフォーマンスが低下する可能性があります。

101 ページの『関連資料』では、特定の構成をチューニングする際に役に立つ可能性のある関連資料を紹介しています。

4.1 パフォーマンス・チューニング手法

BPM 環境のパフォーマンス・チューニングを行うにあたっては、システム全体を考慮したアプローチを採用することをお勧めします。トレーニングと経験を必要とするシステム・パフォーマンス・チューニングについては、ここでは詳しく説明していません。代わりに、チューニングに関するいくつかの重要な側面に焦点を当てています。

チューニングは、次のように導入トポロジーのあらゆる要素を包含しています。

- 物理ハードウェア・トポロジーの選択
- オペレーティング・システムのパラメーター
- BPM サーバー、WebSphere Application Server、およびメッセージング・エンジンの設定

チューニング手法は、次のような反復手順として説明できます。

1. 一連の適切な初期パラメーター設定を選択して、システムを稼働させます。
2. システムをモニターして、パフォーマンスが制限されているかどうかを示すメトリックを確認します。
3. モニタリング・データを参考にして、さらなるチューニング変更を加えます。
4. 完了するまで繰り返します。

以下のサブセクションでは、これらの手順を説明しています。

4.1.1 一連の適切な初期パラメーター設定の選択

パラメーターを体系的に設定するには、49 ページの 4.2、「チューニング・チェックリスト」のチューニング・チェックリストに従ってください。

具体的な初期値については、93 ページの第 5 章、「初期構成設定」を参照して、内部パフォーマンス評価で使用されるさまざまなワークロードの設定を確認してください。これらの値を初期値として検討できます。

4.1.2 システムのモニタリング

システム・コンポーネントをモニターして、次のようにシステムの正常性とさらなるチューニングの必要性を判定することをお勧めします。

- トポロジー内の各物理マシンについて（Web サーバーやデータベース・サーバーなどのフロントエンドとバックエンドのサーバーを含む）、適切な OS ツール（**vmstat**、**iostat**、**netstat** など）を使用して次のプロセスをモニターします。
  - プロセッサー・コア使用量
  - メモリー使用量
  - ディスク使用量
  - ネットワーク使用量
- 物理マシン上で開始された各 Java 仮想マシン（JVM）プロセスについて（プロセス・サーバーやメッセージング・エンジン・サーバーなどのコンポーネント）、次の操作を実行します。
  - **ps** などのツールを使用して、プロセスごとのコア使用量とメモリー使用量を確認します。
  - 詳細ガーベッジ・コレクション（verbosegc）統計データを収集して、Java メモリー使用量に関する情報を取得します。
- BPM サーバーまたはメッセージング・エンジンの JVM ごとに、IBM Tivoli® Performance Viewer を使用して次のタイプのデータをモニターします。
  - 各データ・ソースのデータ接続プール使用量
  - 各スレッド・プール（Web コンテナー、デフォルト、および作業マネージャー）のスレッド・プール使用量

4.1.3 モニタリング・データを参考にしたさらなるチューニング変更

モニタリング・データを正しく活用して BPM のチューニング内容を決定するには、スキルと経験が必要です。一般にこのチューニング・フェーズでは、アナリストが収集されたモニタリング・データを参照して、パフォーマンスのボトルネックを発見して、さらなるチューニングを加える必要があります。このチューニング・フェーズの特徴は、直前のフェーズで収集されたモニタリング・データに基づいて実行される点です。

パフォーマンスのボトルネックとしては、例えば以下の状況が考えられますが、これに限定されるものではありません。

- プロセッサー・コア、ディスク、メモリーなどの物理リソースの過剰な使用。これらの問題を解決するには、物理リソースを追加するか、使用可能なリソース間で再ロード・バランシングします。
- 仮想リソースの過剰な使用。これらのリソースの例としては、ヒープ・メモリー、接続プール、スレッド・プールなどが挙げられます。この場合は、チューニング・パラメーターを使用してボトルネックを解消してください。

4.2 チューニング・チェックリスト

このチェックリストは、BPM ソリューションをチューニングする際の指針として活用できます。これらの各項目については、この章の残り部分で詳しく説明しています。

- 一般的なチューニング可能事項
  - すべてのサーバーに 64 ビット JVM を使用します。
  - 可能な場合はトレーシングとモニタリングを無効にします。
  - すべてのデータベースを DB2 などの高性能データベース管理システム（DBMS）上に配置して、これらのデータベースを適切にチューニングします。
  - セキュリティーが必要な場合は、Java2 セキュリティーではなくアプリケーション・セキュリティーを使用します。
  - パフォーマンス測定に適したハードウェア構成を使用します（例えば、ノートブックとデスクトップは現実的なパフォーマンス評価に適していません）。
  - ハードウェア仮想化が使用されている場合は、十分なプロセッサー、メモリー、および入出力のリソースが各仮想マシンに割り振られていることを確認します。供給能力を超えるリソースを割り振ることは避けてください。
  - 実動サーバーを開発モードで実行したり開発プロファイルを使用して実行したりしてはいけません。
  - 外部サービス・プロバイダーや外部インターフェースをチューニングして、これらがシステム・ボトルネックの原因となることを防止します。
  - メッセージ駆動型 Bean（MDB）のアクティベーション・スペックを構成します。
  - クラスター化に対応するように構成します（必要に応じて）。
  - スレッド・プール・サイズを構成します。
  - 接続プール・サイズと準備済みステートメントのキャッシュ・サイズについてデータ・ソースの設定を構成します。Common Event Infrastructure データ用に非 XA データ・ソースを使用することを検討してください（そのデータが重大なデータでない場合）。
  - データ・プール内の最大接続数をすべての最大スレッド・プール・サイズの合計以上の値に増やします。
- Business Process Choreographer（BPC ビジネス・プロセス用）

- 長時間実行されるプロセスに対して作業マネージャー・ベースのナビゲーションを使用して、メッセージ・プール・サイズとトランザクション間キャッシュ・サイズを最適化します。
- 照会テーブルを使用して、照会応答時間を最適化します。
- 次の Business Flow Manager リソースを最適化します。
  - データベース接続（Business Process Choreographer データベース）
  - アクティベーション・スペック（**BPEInternalActivationSpec**）
  - Java Message Service（JMS）データ・ソース接続（BPECF と BPECFC）
- Business Process Choreographer データベース（BPEDB）のデータベース構成を最適化します。
- DB2 設計アドバイザーなどのデータベース・ツールを使用して、タスク・リスト照会やプロセス・リスト照会から得られた SQL ステートメントの索引を最適化します。
- 不要なオブザーバーを無効にします（監査ロギングなど）。

► Business Processing Modeling Notation（BPMN）ビジネス・プロセス

- twproc データベースのログ・ファイル・サイズを大きくします（16,384 ページに設定）。
- 次のコマンドを実行して twproc データベースのファイル・システム・キャッシングを有効にします。

  db2 alter tablespace userspace1 file system caching
- SIBOWNER テーブルを自動 runstats 実行の対象から除外します。
- データベース統計情報が最新であることを確認します。
- 新しい索引を作成します。詳しくは、次のアドレスにある『Saved search performance degradation with WebSphere Lombardi Edition』という技術情報を参照してください。

  http://www-01.ibm.com/support/docview.wss?uid=swg21474536

► メッセージングとメッセージ・バインディング

- アクティベーション・スペックを最適化します（JMS、MQJMS、WebSphere MQ）。
- キュー接続ファクトリーを最適化します（JMS、MQJMS、WebSphere MQ）。
- 接続プール・サイズを構成します（JMS、MQJMS、WebSphere MQ）。
- サービス統合バスのデータ・バッファー・サイズを構成します。

► データベース

- データベースの表スペースとログを高速ディスク・サブシステム上に配置します。
- ログを表スペースコンテナーとは別個のデバイス上に配置します。
- 表上に現行索引を保持します。
- データベース統計情報を更新します。
- ログ・ファイル・サイズを正しく設定します。
- バッファー・プール・サイズ（DB2）またはバッファー・キャッシュ・サイズ（Oracle）を最適化します。

► Java

- ヒープ・サイズと新世代領域サイズを設定して、メモリーを効率的に管理します。
- 適切なガーベッジ・コレクション・ポリシーを選択します（通常は **-Xgcpolicy:gencon**）。

► Business Monitor

- Common Event Infrastructure を構成します。
- メッセージ消費バッチ・サイズを設定します。
- 重要業績評価指標（KPI）のキャッシングを有効にします。
- テーブル・ベースのイベント送達を使用します。
- データ移動サービスを有効にします。

4.3 一般的なチューニング・パラメーター

ここでは、BPMN ビジネス・プロセスと Business Process Execution Language（BPEL）ビジネス・プロセスの両方について、BPM V7.5 ソリューションをチューニングするために一般に使用されるパフォーマンス・チューニング・パラメーターを紹介します。

4.3.1 トレーシングとロギングのフラグ

トレーシングとロギングは、システムのセットアップ時や問題のデバッグ時に必要になることがよくあります。ただしこれらの機能は、多くの場合に大切なパフォーマンス・リソースを必要とします。パフォーマンスの評価時や実稼働環境では、これらのリソースの使用量をできる限り少なくしてください。ここでは、本書で取り上げている製品で使用されるトレーシング・パラメーターを紹介します。設定の中には、これらの製品のすべてまたは一部に共通しているものも、特定の製品に固有のものもあります。特に記載がない限り、これらのパラメーターはすべて管理コンソールを使用して設定できます。

トレーシングを有効または無効にするには、サブスクリプションのプロパティーで**「トラブルシューティング」→「ログおよびトレース」**をクリックします。ログ詳細レベルを変更するサーバーを選択して、**「ログ詳細レベルの変更」**をクリックします。**「構成」**フィールドと**「実行時」**フィールドを両方とも * =all=disabled に設定します。

Performance Monitoring Infrastructure（PMI）レベルを変更するには、**「モニターおよびチューニング」→「Performance Monitoring Infrastructure」**をクリックします。ログ詳細レベルを変更するサーバーを選択して、**「なし」**をクリックします。

また、Cross-Component Tracing（XCT）は問題判別に役立ち、Service Component Architecture（SCA）コンポーネント情報をログ・エントリーと相関付けることを可能にします。ただし、XCT を実稼働環境内やパフォーマンス・データの取得時に使用しないでください。次の 2 つのレベルの XCT 設定が可能です。

- 有効にする
- データ・スナップショット付きで有効にする

どちらの設定でも、大量のパフォーマンス・リソースが使用されます。「Enable with data snapshot」の設定では、スナップショットをファイルに保存するために追加の入出力が必要になるため、大きな負荷が生じます。

XCT を有効または無効にするには、**「トラブルシューティング」→「クロス・コンポーネント・トレース」**をクリックします。**「構成」**タブまたは**「実行時」**タブで、**XCT** の設定を次の 3 つのオプションから選択します。

- 有効にする
- 無効にする
- データ・スナップショット付きで有効にする

「**実行時**」タブで加えた変更は直ちに反映されます。「**構成**」タブで加えた変更は、サーバーの再始動後に反映されます。

詳しくは、次のアドレスにある『Managing Log Level Settings in TeamWorks』という技術情報を参照してください。

http://www-01.ibm.com/support/docview.wss?uid=swg21439659

4.3.2 Java のチューニング・パラメーター

ここでは、Java 仮想マシン（JVM）のよく使用されるチューニング・パラメーターをいくつか紹介します。完全なリストについては、JVM サプライヤーによって提供されている JVM チューニング・ガイドを参照してください。

JVM パラメーターを変更するには、まず「**サーバー**」→「**アプリケーション**」→「**Performance Monitoring Infrastructure**」をクリックして JVM 管理ウィンドウに移動します。JVM チューニング・パラメーターを変更するサーバーを選択します。次に、「**サーバー・インフラストラクチャー**」→「**Java およびプロセス管理**」→「**プロセス定義**」→「**追加プロパティー**」→「**Java 仮想マシン**」をクリックします。このパネルで JVM パラメーターを変更します。

Java ガーベッジ・コレクション・ポリシー

IBM JVM がインストールされたプラットフォーム上のデフォルトのガーベッジ・コレクション（GC）アルゴリズムは、世代別同時コレクターです（JVM 管理コンソール・パネルの「汎用 JVM 引数」で -Xgcpolicy:gencon によって指定）。IBM の社内評価の結果から、このガーベッジ・コレクション・ポリシーは通常、チューニングされた新世代領域サイズを使用することでパフォーマンスを向上させることがわかっています（次のセクションを参照）。

Java ヒープ・サイズ

デフォルトの Java ヒープ・サイズを変更するには、初期ヒープ・サイズと最大ヒープ・サイズを管理コンソールの JVM ウィンドウで明示的に設定します。64 ビット JVM（BPM V7.5 サーバーの推奨モード）は、32 ビット JVM よりはるかに大きなヒープ・サイズをサポートしています。この機能を使用して Java ヒープ内のメモリー負荷を軽減する一方で、JVM ヒープ・サイズと他のすべてのメモリー要件を補助するための十分な物理メモリーが存在することを常に確認してください。

「**世代別並行ガーベッジ・コレクター**」をクリックした場合は、Java ヒープは、新しいオブジェクトが割り振られる新世代領域（nursery）と、長期間にわたって存続しているオブジェクトが配置される旧世代領域（tenured space）に分割されます。合計ヒープ・サイズは、新世代領域と旧世代領域の合計です。新世代領域サイズは、合計ヒープ・サイズとは独立して設定できます。通常は、新世代領域サイズは、合計ヒープ・サイズの 1/4 から 1/2 の間に設定します。関連するパラメーターは次のとおりです。

- Xmns<*size*>: 新世代領域の初期サイズ
- Xmnx<*size*>: 新世代領域の最大サイズ
- Xmn<*size*>: 新世代領域の固定サイズ

4.3.3 メッセージ駆動型 Bean の ActivationSpec

管理コンソールでは、次のように複数のショートカットを介して MDB の **ActivationSpec** チューニング・パラメーターにアクセスできます。

- 「**リソース**」→「**リソース・アダプター**」→「**J2C アクティベーション・スペック**」をクリックします。アクセスするアクティベーション・スペックの名前を選択します。
- 「**リソース**」→「**リソース・アダプター**」→「**リソース・アダプター**」をクリックします。アクセスするリソース・アダプターの名前を選択します。次に、「**追加プロパティー**」→

「J2C アクティベーション・スペック」をクリックします。希望のアクティベーション・スペックの名前を選択します。

メッセージ駆動型 Bean（MDB）の ActivationSpec 内の次のカスタム・プロパティーは、パフォーマンスに大きな影響を与えます。これらのプロパティーについて詳しくは、57 ページの『メッセージ駆動型 Bean の ActivationSpec プロパティーのチューニング』を参照してください。

- **maxConcurrency**
- **maxBatchSize**

4.3.4 スレッド・プール・サイズ

BPM サーバーは、スレッド・プールを使用して同時タスクを管理します。管理コンソールでスレッド・プールの「最大サイズ」プロパティーを設定するには、**「サーバー」→「アプリケーション・サーバー」**をクリックして、管理するスレッド・プールが配置されているサーバーの名前を選択します。**「追加プロパティー」→「スレッド・プール」**をクリックしてから、スレッド・プール名をクリックします。

通常は、次のスレッド・プールをチューニングする必要があります。

- Default
- ORB.thread.pool
- WebContainer

また、BPEL プロセスの作業マネージャーによって使用されるスレッド・プールは、コンソール内で別個に構成します。そのためには、**「リソース」→「非同期 Bean」→「作業マネージャー」**をクリックします。作業マネージャー名を選択して、**「スレッド・プールのプロパティー」**をクリックします。

通常は、次の作業マネージャーをチューニングする必要があります。

- DefaultWorkManager
- BPENavigationWorkManager

4.3.5 Java Message Service の接続プール・サイズ

管理コンソールから JMS 接続ファクトリーと JMS キュー接続ファクトリーにアクセスするには、以下の複数の方法があります。

- **「リソース」→「リソース・アダプター」→「J2C 接続ファクトリー」**をクリックして、ファクトリー名を選択します。
- **「リソース」→「JMS」→「接続ファクトリー」**をクリックして、ファクトリー名を選択します。
- **「リソース」→「JMS」→「キュー接続ファクトリー」**をクリックして、ファクトリー名を選択します。
- **「リソース」→「リソース・アダプター」→「リソース・アダプター」**をクリックして、リソース・アダプター名を選択します（SIB JMS リソース・アダプターなど）。次に、**「追加プロパティー」→「J2C 接続ファクトリー」**をクリックして、ファクトリー名を選択します。
- 接続ファクトリーの管理パネルで、**「追加プロパティー」→「接続プール・プロパティー」**をクリックします。接続プールの最大サイズとして**「最大接続数」**プロパティーを設定します。

4.3.6 Java Database Connectivity データ・ソースのパラメーター

データ・ソースには、次のいずれかの方法でアクセスできます。

- **「リソース」→「JDBC」→「データ・ソース」**をクリックして、データ・ソース名を選択します。
- **「リソース」→「JDBC プロバイダー」**をクリックして、Java Database Connectivity（**JDBC）プロバイダー名**を選択してから、**「追加プロパティー」→「データ・ソース」**をクリックして、データ・ソース名を選択します。

接続プール・サイズ

データ・ソースの接続プールの最大サイズは、「最大接続数」プロパティーの値によって制限されます。このプロパティーを構成するには、データ・ソース・ウィンドウで**「追加プロパティー」→「接続プール・プロパティー」**をクリックします。

データ・プール内の最大接続数をすべての最大スレッド・プール・サイズの合計以上の値に増やします。

通常は、次のデータ・ソースをチューニングする必要があります。

- BPEDB および関連するメッセージ・エンジン・データベース（BPEL ビジネス・プロセス用）用の Business Process Choreographer（BPC）データ・ソース
- BPMN データベースおよび関連するメッセージ・エンジン・データベース（BPMN ビジネス・プロセス用）用の BPMN データ・ソース
- SCA アプリケーション・バス・メッセージング・エンジンのデータ・ソース
- SCA システム・バス・メッセージング・エンジンのデータ・ソース
- Common Event Infrastructure（CEI）バス・メッセージング・エンジンのデータ・ソース

準備済みステートメントのキャッシュ・サイズ

データ・ソースから、データ・ソースの準備済みステートメントのキャッシュ・サイズを構成できます。**「追加プロパティー」→「WebSphere Application Server データ・ソース・プロパティー」**をクリックします。

BPM サーバーの場合は、BPEDB データ・ソースをより大きな値にチューニングしてください。最初は、この値を 300 に設定します。

4.3.7 メッセージング・エンジンのプロパティー

メッセージング・エンジンの次の 2 つのカスタム・プロパティーは、メッセージング・エンジンのパフォーマンスに影響を与える可能性があります。

- sib.msgstore.discardableDataBufferSize
  - このプロパティーは、ベスト・エフォート非永続メッセージについてはデータ・バッファー内に留まります。
  - デフォルト値は 320 KB です。
  - このバッファーがいっぱいになると、メッセージは破棄されて、新しいメッセージをバッファーに書き込めるようになります。
- sib.msgstore.cachedDataBufferSizeCachedDataBufferSize
  - このプロパティーは、ベスト・エフォート非永続メッセージ以外のメッセージについてはメモリー・キャッシュ内に留まります。
  - デフォルト値は 320 KB です。

これらのプロパティーにコンソール内でアクセスするには、まず**「統合サービス」→「バス」**をクリックして、バス名を選択します。次に、**「メッセージング・エンジン」**をクリックして、メッセージング・エンジン名を選択します。最後に、**「追加プロパティー」→「カスタム・プロパティー」**をクリックします。

4.3.8 実動モードでの実動サーバーの実行

BPM サーバーを開発モードで実行することで、バイトコード検証を無効にする JVM 設定を使用してサーバーの起動時間を短縮できるとともに、JIT（Just-In-Time）コンパイル時間を短縮できます。この設定は、最適な実行時パフォーマンスをもたらすように設計されていないため、実動サーバーでは使用しないでください。サーバーの**「開発モードでの実行」**チェック・ボックスが選択解除されていることを確認します。この設定は、管理コンソール内のサーバーの構成ウィンドウにあります。**「サーバー」→「アプリケーション・サーバー」**をクリックします。設定を変更するサーバーをクリックして、**「構成」**をクリックします。

実動テンプレートまたは開発テンプレートを使用してサーバー・プロファイルを作成することもできます。実動サーバーには実動プロファイル・テンプレートを使用してください。

4.4 高度なチューニング

ここでは、高度なチューニングのヒントを記載しています。

4.4.1 トレーシングとモニタリングの考慮事項

さまざまなシステム・コンポーネントごとに異なるレベルでトレーシングとモニタリングを構成できることは、システムの解析時やデバッグ時に役に立ちます。BPM 製品セットが提供する幅広いモニタリング機能には、CEI や監査ロギングを通じたビジネス・モニタリングと、PMI やアプリケーション応答測定（ARM）インフラストラクチャーを通じたシステム・パフォーマンス・モニタリングの両方が含まれます。これらの機能は、実行中のソリューションのパフォーマンスを把握するのに役立ちますが、システムの全体的なパフォーマンスとスループットを低下させる要因にもなり得ます。

トレーシングとモニタリングがパフォーマンスに与える影響 : トレーシングとモニタリングは慎重に使用してください。最適なパフォーマンスを確保するために、できる限りこれらを無効にしてください。

ほとんどのトレーシングとモニタリングは、管理コンソールを通じて制御されます。管理コンソールを通じて、PMI のモニタリング、ロギング、およびトレーシングの設定に対して、適切なレベルのトレーシングとモニタリングが設定されていることを確認します。

管理コンソールを使用して、Business Flow Manager と Human Task Manager で**「監査ロギング」**チェック・ボックスと**「Common Event Infrastructure のロギング」**チェック・ボックスが選択解除されていることを確認します（ただしこれらの機能がビジネス上の理由から必要な場合は除く）。

Integration Designer も、イベント・モニタリングを制御するために使用されます。お使いのコンポーネントとビジネス・プロセスの**「イベント・モニター」**タブをチェックして、イベント・モニタリングが適切に適用されていることを確認してください。

4.4.2 ラージ・オブジェクトのチューニング

ここでは、ラージ・オブジェクト使用時のパフォーマンスのチューニングについて説明します。

Java ヒープ・サイズの最大化

ラージ・オブジェクトの処理に影響を与える主な要因の 1 つは、Java ヒープの最大サイズです。ここでは、Windows と AIX という 2 つの代表的なプラットフォームでヒープ・サイズをできる限り大きく設定する方法を説明します。

- Windows（32 ビット）

  32 ビット版の Windows オペレーティング・システムではアドレス・スペースが制限されているため、32 ビット JVM について確保可能な最大ヒープは 1.4 GB ～ 1.6 GB です。

- AIX（32 ビット）

  32 ビット版の AIX システムでは、Java 5 と Java 6 の JVM は通常は 2 GB ～ 2.4 GB の範囲のヒープをサポートしています。
  32 ビット・システムで使用可能な 4 GB のアドレス・スペースは他のリソースと共有されるため、ヒープ・サイズの実際の上限は、これらのリソースで使用されるメモリーに依存します。これらのリソースとしては、スレッド・スタック、JIT コンパイル済みコード、ロード済みクラス、共有ライブラリー、OS システム・サービスによって使用されるバッファーなどが挙げられます。ヒープ・サイズが大きいと、他のリソース用に予約されているアドレス・スペースが圧迫されて、実行時エラーが発生する可能性があります。

最大ヒープ・サイズの設定は 32 ビット JVM のみに適用されます。64 ビット JVM を使用している場合は、ヒープ・サイズは使用可能な物理メモリーの量によってのみ制限されます。BPM V7.5 サーバーの推奨構成は 64 ビットです。

詳しいヒープ設定手法については、88 ページの 4.13、「高度な Java ヒープ・チューニング」を参照してください。

ラージ・オブジェクト処理時の他の処理の抑制または回避

大きめのオブジェクト・サイズに対応するための方法の 1 つは、JVM 内の同時処理を制限することです。他の BPM サーバー・アクティビティーや WebSphere Adapters アクティビティーの処理と並行して、連続して受信される最大級のラージ・オブジェクトを処理することを想定してはいけません。ラージ・オブジェクトを考慮する際の動作上の前提は、すべてのオブジェクトが大きかったり非常に大きかったりするわけではないこと、およびラージ・オブジェクトは頻繁に受信されるわけではなく、おそらく 1 日に 1 ～ 2 回であることです。複数の非常に大きいオブジェクトが同時に処理されている場合は、エラーの可能性が大幅に高まります。

通常受信される小さめのオブジェクトのサイズと数は、システム内の Java ヒープ・メモリー消費量に影響を与えます。一般に、ラージ・オブジェクトの処理時のシステム負荷が大きいほど、メモリー問題が発生する可能性が高まります。

アダプターの場合は、同時処理の量を調整するには、**pollPeriod** パラメーターと **pollQuantity** パラメーターを設定します。大きめのオブジェクト・サイズに対応するには、**pollPeriod** に比較的大きい値を設定して（10 秒など）、**pollQuantity** には比較的小さい値を設定することで（1 秒など）、発生する同時処理の量をできる限り少なくします。これらの設定はピーク・スループットに対しては最適ではないため、アダプター・インスタンスが、小さめのオブジェクトと散発的なラージ・オブジェクトの組み合わせについてハイスループットをサポートする必要がある場合は、兼ね合いを考慮して設定する必要があります。

4.4.3 並行性を最大化するためのチューニング

サーバー・クラス・ハードウェア上のほとんどの大規模デプロイメント環境では、多くの処理が同時実行されます。チューニングによって並行性を最大化することで、サーバーはそのコアをフルに使用するのに十分な負荷を受け入れるようになります。構成のチューニングが不十分であることを示す現象の 1 つは、コアがフルに使用されていないにもかかわらず、追加の負荷が発生してもコアの使用量が増えないことです。これらの処理を最適化して並行性を最大化するための一般的な指針は、実行フローをたどって、ボトルネックを 1 つずつ取り除くことです。

同時処理量が増えるほど、サーバー上のリソース要件（メモリー量やスレッド数）も増大します。同時処理量の増大と他のチューニング目的とのバランスを考慮する必要があります。これらの目的としては、ラージ・オブジェクトの処理、多数のユーザーの処理、応答時間の短縮などが挙げられます。

並行性のためのエッジ・コンポーネントのチューニング

並行性を確保するためにエッジ・コンポーネントをチューニングするプロセスの最初のステップは、BPM ソリューションのエッジ・コンポーネントで複数のビジネス・オブジェクトが同時に処理されるようにすることです。入力ビジネス・オブジェクトの出所がアダプターである場合は、アダプターが入力メッセージの同時配信に対応するようにチューニングされていることを確認します。

入力ビジネス・オブジェクトの出所が WebServices エクスポート・バインディング、または Java Server Pages（JSP）またはサーブレットからの直接呼び出しである場合は、WebContainer スレッド・プールが適切なサイズに設定されていることを確認します。例えば、100 件の未完了要求を BPM サーバーで同時に処理することを可能にするには、WebContainer スレッド・プールの最大サイズを 100 以上に設定する必要があります。

入力ビジネス・オブジェクトの出所がメッセージングである場合は、ActivationSpec（MDB バインディング）およびリスナー・ポート（WebSphere MQ または MQJMS バインディング）をチューニングする必要があります。

メッセージ駆動型 Bean の ActivationSpec プロパティーのチューニング

各 JMS エクスポート・コンポーネントについて、1 つの MDB とその対応する ActivationSpec が、module name/export component name_AS という JNDI（Java Naming and Directory Interface）名内に存在します。JMS エクスポート MDB の **maxConcurrency** のデフォルト値である 10 を使用した場合は、最大で 10 個のビジネス・オブジェクトを JMS キューから MDB スレッドに同時配信できます。100 個の同時配信が必要な場合は、この値を 100 に変更します。

Tivoli Performance Viewer を使用して、**maxConcurrency** パラメーターをモニターできます。MDB によって処理されている各メッセージについて、トランザクション内でロック中としてマークされた 1 つのメッセージがキューに含まれています（このメッセージは onMessage の完了後に削除されます）。これらのメッセージは使用不能として分類されています。UnavailableMessageCount という PMI メトリックは、各キュー・ポイント上の使用不能なメッセージの数を示します。この値を確認するには、リソース名を選択してから、**「SIB サービス」→「SIB メッセージング・エンジン」**をクリックします。バス名を選択して、**「宛先」→「キュー」**をクリックします。

いずれかのキューに maxConcurrency の値以上の使用不能メッセージが含まれている場合は、そのキュー内のメッセージの数は現在、MDB の最大並行性を上回っているということです。この場合は、その MDB について **maxConcurrency** の値を大きくしてください。

アクティベーション・スペック内の最大バッチ・サイズもパフォーマンスに影響を与えます。デフォルト値は 1 です。最大バッチ・サイズの値によって、1 ステップでメッセージング層から取得されてアプリケーション層に配信されるメッセージの数が決定されます。このバッチ・サイズ値は、この作業が単一のトランザクション内で実行されることを意味しない

ため、この設定はトランザクション・スコープに影響を与えません。特に大規模なマルチコア・システムの場合に、パフォーマンスとスケーラビリティーを高めるには、SCA モジュールおよび実行時間の長いビジネス・プロセスに関連付けられたアクティベーション・スペックについて、この値を大きくします（8 など）。

スレッド・プール・サイズの構成

スレッド・プールのサイズは、サーバーが複数のアプリケーションを同時実行する能力に直接的な影響を与えます。並行性を最大化するためには、スレッド・プール・サイズを最適な値に設定する必要があります。**maxConcurrency** パラメーターまたは **Maximum sessions** パラメーターの値を大きくした場合は、JMS または WebSphere MQ のキューからの複数ビジネス・オブジェクトの同時配信が可能になるだけです。BPM サーバーで複数の要求を同時に処理するためには、対応するスレッド・プール・サイズを大きくして、これらの MDB スレッドの同時実行量を増やす必要があります。

MDB の作業は、デフォルトのスレッド・プールから割り振られたスレッドにディスパッチされます。アプリケーション・サーバー内のすべての MDB は、このスレッド・プールを共有します（異なるスレッド・プールが指定されている場合は除く）。このことは、デフォルト・スレッド・プール・サイズを、どの個別 MDB の **maxConcurrency** よりも大きくする（場合によっては大幅に大きくする）必要があることを意味します。

WebContainer スレッド・プール内のスレッドは、受信される HTTP サービス要求と Web サービス要求を処理するために使用されます。このスレッド・プールは、サーバー上に導入されたすべてのアプリケーションによって共有されるため、通常はデフォルトよりも大きい値にチューニングする必要があります。

オブジェクト・リクエスト・ブローカー（ORB）のスレッド・プールのスレッドは、ORB 要求（リモート EJB 呼び出しなど）を実行するために使用されます。このスレッド・プールのサイズは、特定のヒューマン・タスク・マネージャー API などのインターフェースを通じて送られてくる要求を処理するのに十分な大きさである必要があります。

メッセージ駆動型 Bean の専用スレッド・プールの構成

デフォルトのスレッド・プールは、多くの WebSphere Application Server タスクによって共有されます。このため、必要に応じて JMS MDB の実行を専用スレッド・プールに分散することをお勧めします。JMS MDB スレッド用に使用されるスレッド・プールを変更するには、次の手順を実行します。

1. サーバー上にスレッド・プールを作成します（MDBThreadPool など）。そのためには、**「サーバー」→「サーバー・タイプ」→「WebSphere Application Server」→「サーバー」→「スレッド・プール」**をクリックします。次に、**「新規作成」**をクリックします。
2. **「リソース」→「リソース・アダプター」→「リソース・アダプター」**をクリックして、サービス統合バス（SIB）JMS Resource Adapters の管理コンソールをサーバー・スコープで開きます。このアダプターが表示されていない場合は、**「設定」**に移動して、**「ビルトイン・リソースを表示」**チェック・ボックスを選択します。
3. スレッド・プールの別名を **Default** から **MDBThreadPool** に変更します。
4. ノード・スコープとセル・スコープで、SIB JMS Resource Adapters について手順 2 と 3 を繰り返します。
5. サーバーを再始動して、これらの変更内容を有効にします。

非同期 SCA 呼び出しのための SCA モジュール MDB は、Platform Messaging Component SPI Resource Adapter という別個のリソース・アダプターを使用します。希望に応じて、同じ手順に従ってスレッド・プールを異なるスレッド・プールに変更してください。

専用スレッド・プールを使用する場合でも、このリソース・アダプターと関連付けられたすべての MDB は 1 つのスレッド・プールを共有します。ただしこれらの MDB は、同様に

Default スレッド・プールを使用する他の WebSphere Application Server タスクと競合する必要はありません。

Java Message Service と Java Message Service キュー接続ファクトリーの構成

複数の同時実行されているスレッドは、JMS やデータベースの接続プールなどのリソースでボトルネックになる可能性があります（このようなリソースが適切にチューニングされていない場所）。**「最大接続数プールサイズ」**は、このプール内で作成できる物理接続の最大数を指定します。これらの物理接続は、バックエンド・リソース（DB2 データベースなど）とつながります。スレッド・プールの上限に達したら、要求側は新しい物理接続を作成できないため、現在使用中の物理接続がプールに戻されるまで待つ必要があります。物理接続がプールに戻されない場合は、**ConnectionWaitTimeout** 例外が発行されます。

例えば、**「最大接続数」**の値を 5 に設定した場合に、5 つの物理接続が使用中の場合は、プール・マネージャーは、**「接続タイムアウト」**で指定された時間だけ物理接続が解放されるのを待ちます。基盤リソースへの接続を待っているスレッドは、接続が解放されてプール・マネージャーによってそのスレッドに割り振られるまで、ブロックされた状態になります。指定された時間内に接続が解放されない場合は、ConnectionWaitTimeout 例外が発行されます。

「最大接続数」を 0 に設定した場合は、接続プールは無制限に拡大できるようになります。ただしこの値に設定した場合は、**「接続タイムアウト」**の値が無視されます。

接続ファクトリーのチューニングに関する一般的な指針として、接続ファクトリーの最大接続プール・サイズを「同時スレッド数 × スレッドあたりの同時接続数」と一致させる必要があります。

それぞれの JMS、WebSphere MQ、または MQJMS インポートについて、アプリケーション導入時に作成された 1 つの接続ファクトリーが存在します。JMS 接続ファクトリーと関連付けられた接続プールの**「最大接続数」**プロパティーを、インポート・コンポーネントで同時実行されているすべてのスレッドに接続を提供するのに十分な大きさの値に設定します。例えば、100 個のスレッドが特定のモジュールで実行されることが想定される場合は、**「最大接続数」**プロパティーの値を 100 に設定します。デフォルト値は 10 です。

接続ファクトリーの構成パネルで、**「追加プロパティー」→「接続プール・プロパティー」**をクリックします。**「最大接続数」**プロパティーの値を接続プールの最大サイズに設定します。

データ・ソース・オプションの構成

データ・ソースの「最大接続数」プロパティーは、すべてのスレッドからデータベースへの同時アクセスを可能にするのに十分な大きさの値に設定します。通常は、いくつかのデータ・ソースが BPM サーバー上で構成されています（例えば、BPEDB データ・ソース、TWPROC データ・ソース、TWPERFDB データ・ソース、WPSDB データ・ソース、メッセージング・エンジン・データベース・データ・ソースなど）。各データ・ソースの**「最大接続数」**プロパティーを、他のシステム・リソースの最大並行性と一致する値に設定します。詳しくは、57 ページの 4.4.3、「並行性を最大化するためのチューニング」を参照してください。

データ・ソースの準備済みステートメントのキャッシュ・サイズの設定

BPC コンテナーは、準備済みステートメントを多用します。このため、データベースにアクセスするためにステートメントを繰り返し準備する必要がないように、ステートメントのキャッシュ・サイズを十分な大きさに設定してください。

BPEDB の準備済みステートメント・キャッシュ・サイズは、300 以上に設定してください。

4.4.4 メッセージングのチューニング

ここでは、メッセージングのパフォーマンス・チューニングについて説明します。

メッセージング・エンジンのデータ・ストアまたはファイル・ストアの選択

メッセージング・エンジンの永続性は、データベースによってサポートされています。ただし、スタンドアロンの BPM 構成では、BPE バスと SCA バスの永続ストレージがファイル・システム（ファイル・ストア）によってサポートされていることがあります。プロファイルの作成時にファイル・ストアを選択する必要があります。プロファイル管理ツールを使用して、新しいスタンドアロン・エンタープライズ・サービス・バス・プロファイルまたはスタンドアロン・プロセス・サーバー・プロファイルを作成します。

そのためには、**「プロファイル作成オプション」→「詳細プロファイル作成」→「データベース構成」**をクリックして、**「メッセージング・エンジン (ME) でこのファイル・ストアを使用する」**チェック・ボックスを選択します。このプロファイルを使用する場合は、ファイル・ストアは BPE と SCA のサービス統合バス用に使用されます。

データ・バッファー・サイズの設定（廃棄可能またはキャッシュ格納式）

DiscardableDataBufferSize は、ベスト・エフォート非永続メッセージの処理時に使用されるデータ・バッファーのサイズです（バイト単位）。廃棄可能なデータ・バッファーの目的は、メッセージ・データをメモリー内に保持することです。このサービス品質ではこのデータはデータ・ストアに一切書き込まれないからです。大きすぎてこのバッファーに格納できないメッセージは破棄されます。

CachedDataBufferSize は、ベスト・エフォート非永続メッセージを除くすべてのメッセージの処理時に使用されるデータ・バッファーのサイズです（バイト単位）。キャッシュ格納式データ・バッファーの目的は、通常であればデータ・ストアから読み取られる必要がある可能性のあるメモリー内データをキャッシュに格納することでパフォーマンスを最適化することです。

DiscardableDataBufferSize と **CachedDataBufferSize** を管理コンソールで設定するには、**「サービス統合バス」**をクリックして、バス名を選択します。**「メッセージング・エンジン」**をクリックして、メッセージング・エンジン名を選択します。**「追加プロパティー」→「カスタム・プロパティー」**をクリックして、**DiscardableDataBufferSize** と **CachedDataBufferSize** の値を入力します。

メッセージング・エンジンのデータ・ストア用としての高性能データベース管理システムの使用

パフォーマンスを高めるには、メッセージング・エンジンのデータ・ストア用に DB2 などの実動レベルの品質を備えたデータベースを使用してください。詳細プロファイル作成オプションを使用して、プロファイル作成時にデータベースを選択できます。

DB2 データベースの作成とデータ・ストア・スキーマのロード

メッセージング・エンジンごとに 1 つの DB2 データベースを使用する代わりに、すべてのメッセージング・エンジンを同じデータベースに配置して、表 4-1 に示すように別々のスキーマを使用してこれらのメッセージング・エンジンを分離しています。

表 4-1 メッセージング・エンジンのスキーマ

スキーマ	メッセージング・エンジン
SCASYS	box01-server1.SCA.SYSTEM.box01Node01Cell.Bus
SCAAPP	box01-server1.SCA.APPLICATION. box01Node01Cell.Bus
CEIMSG	box01-server1.CommonEventInfrastructure_Bus

BPCMSG	box01-server1.BPC.box01Node01Cell.Bus

すべてのメッセージング・エンジンを同じデータベースに配置するには、次の手順を実行します。

1. 次のコマンドを実行して、メッセージング・エンジンごとに 1 つのスキーマ定義を作成します。

   ***WAS Install*\bin\sibDDLGenerator.bat -system db2 -version 8.1 -platform windows -statementend ; -schema BPCMSG -user *user* >createSIBSchema_BPCMSG.ddl**

この構文で（Windows オペレーティング・システムで使用）、**WAS Install** は BPM のインストール・ディレクトリーを表し、***user*** はユーザー名を表します。

2. このコマンドをスキーマごとまたはメッセージング・エンジンごとに繰り返し実行します。
3. データベースを複数のディスクに分散するには、作成したスキーマ定義を編集して、使用されるスキーマに応じた名前を付けた表スペース内に各表を配置します。例えば、SCAAPP は SCANODE_TS になり、CEIMSG は CEIMSG_TS になります。編集後のスキーマ定義は 例 4-1 のような内容になります。

   例 4-1 スキーマ定義

```
CREATE SCHEMA CEIMSG;
CREATE TABLE CEIMSG.SIBOWNER (
  ME_UUID VARCHAR(16),
  INC_UUID VARCHAR(16),
  VERSION INTEGER,
  MIGRATION_VERSION INTEGER
) IN CEIMSG_TB;
CREATE TABLE CEIMSG.SIBCLASSMAP (
  CLASSID INTEGER NOT NULL,
  URI VARCHAR(2048) NOT NULL,
  PRIMARY KEY(CLASSID)
) IN CEIMSG_TB;
Ö.
```

   ここでは、各種の表について表スペースを分けることができます。最適な分散は、アプリケーションの構造と負荷の特性によって異なります。この例では、データ・ストアあたり 1 つの表スペースという形にしています。

4. すべてのスキーマ定義を作成して、表の表スペースを定義したら、SIB という名前のデータベースを作成します。
5. システムによって管理された表スペースに対して次のコマンドを実行して、表スペースを作成して、コンテナーを使用可能な複数のディスクに分散します。

   DB2 CREATE TABLESPACE CEIMSG_TB MANAGED BY SYSTEM USING('<*path*>\CEIMSG_TB')

   可能な場合は、データベース・ログを別個のディスクに配置します。
6. 前述の 4 つのスキーマ定義をデータベースにロードすることで、データベースのスキーマを作成します。

データベースのチューニングと DB2 固有のチューニングについて詳しくは、以下のセクションを参照してください。

- 77 ページの 4.10、「一般的なデータベース・チューニング」
- 79 ページの 4.11、「DB2 固有のデータベース・チューニング」

メッセージング・エンジンのデータ・ソースの作成

管理コンソールを使用して、各メッセージング・エンジンのデータ・ソースを作成して、各メッセージング・エンジンが新しいデータ・ストアを使用するように構成します。

そのためには、次の手順を実行します。

1. 非 XA データ・ソース用の DB2 Universal JDBC Driver Provider タイプの JDBC プロバイダーを作成します（存在していない場合）。BPC が DB2 に対して正しく構成されていた場合は、XA DB2 JDBC Driver Provider が存在するはずです。
2. 4 つの新しい JDBC データ・ソースを作成します。1 つは CEI 用の XA データ・ソースとして、残りの 3 つは単一フェーズ・コミット式（非 XA）データ・ソースとして作成します。

表 4-2 では、これらのデータ・ソースの新しい名前を示しています。

表 4-2 新しいデータ・ソース

データ・ソースの名前	JNDI 名	JDBC プロバイダーのタイプ
CEIMSG_sib	jdbc/sib/CEIMSG	DB2 Universal (XA)
SCAAPP_sib	jdbc/sib/SCAAPPLICATION	DB2 Universal
SCASYSTEM_sib	jdbc/sib/SCASYSTEM	DB2 Universal
BPCMSG_sib	jdbc/sib/BPCMSG	DB2 Universal

データ・ソースを作成する際は、次の手順を実行します。

1. **「コンテナー管理パーシスタンス (CMP) 内でこのデータ・ソースを使用する」**チェック・ボックスを選択解除します。
2. コンポーネント管理認証別名を設定します。
3. データベースの名前を、メッセージング（サービス統合バスなど）用として以前に作成されたデータベースと同じ名前に設定します。
4. タイプ 2 またはタイプ 4 のドライバー・タイプを選択します。DB2 は、JDBC Universal Driver タイプ 2 の接続を使用してローカル・データベースにアクセスして、タイプ 4 の接続を使用してリモート・データベースにアクセスすることをお勧めします。タイプ 4 のドライバーを使用する場合は、データベースに対してホスト名と有効なポートを構成する必要があります。

メッセージング・エンジンのデータ・ストアの変更

管理コンソールを使用して、メッセージング・エンジンのデータ・ストアを変更します。

1. ナビゲーション・パネルで、**「統合サービス」→「バス」**をクリックして、表示されている各バス / メッセージング・エンジンのデータ・ストアを変更します。
2. 各データ・ストアの新しい JNDI 名とスキーマ名を入力します。テーブルは既に作成されているため、**「テーブルの作成」**チェック・ボックスを選択解除します。

   サーバーはメッセージング・エンジンを直ちに再始動します。SystemOut.log ファイルには、この変更の結果が表示されるとともに、メッセージング・エンジンが正常に始動したかどうかが示されます。
3. サーバーを再始動して、すべてのシステムが更新後の構成を使用して起動されることを確認します。

最後に残っている作業は、データベースのチューニングです。データベースのチューニングと DB2 固有のチューニングについて詳しくは、以下のセクションを参照してください。

- 77 ページの 4.10、「一般的なデータベース・チューニング」
- 79 ページの 4.11、「DB2 固有のデータベース・チューニング」

4.4.5 クラスター化トポロジーのチューニング

クラスター化トポロジーを導入する目的の 1 つは、増大する負荷が原因でパフォーマンスが低下しているシステム・コンポーネントにリソースを追加することです。理想的には、必要な負荷に合わせてトポロジーを自由に拡張できることが望ましいです。BPM Network Deployment（ND）インフラストラクチャーはこの機能を提供します。ただし、効果的な拡張を行うには、標準のパフォーマンス・モニタリング技法とボトルネック分析技法が必要です。

以下では、クラスター化トポロジーを拡張またはチューニングする際のいくつかの考慮事項とチューニング指針を説明しています。この説明では、クラスター・メンバーの追加はサーバー・ハードウェアの追加も意味することを前提にしています。

- 複数のクラスター・メンバー（JVM）を単一の物理システム上に導入する場合は、システム全体のリソース使用量（コア、ディスク、ネットワークなどのコンポーネント）をモニターすることが重要です。また、各クラスター・メンバーによって使用されるリソースもモニターしてください。モニタリングによって、特定のクラスター・メンバーに起因するシステム・ボトルネックを検知できます。
- クラスターのすべてのメンバーがボトルネック化している場合は、適切な物理ハードウェアによってサポートされた 1 つ以上のメンバーをクラスターに追加することで拡張できます。
- 単一のシングルトン・サーバーまたはクラスター・メンバーがボトルネックとなっている場合は、次のような追加要因を考慮する必要があります。

  – “One of N” ポリシー（イベント順序を保持するために）を使用したクラスター内のメッセージング・エンジンはボトルネックになる可能性があります。この問題を修正するには、次の拡張手法を使用できます。

    - アクティブなクラスター・メンバーをより強力なハードウェア・サーバー上でホストするか、既存のサーバーから外部の負荷を除去します。
    - メッセージング・エンジン・クラスターが複数のバスにサービスを提供しており、メッセージング・トラフィックがこれらのバス間で分散されている場合は、バスごとに別々のメッセージング・エンジン・クラスターを使用することを検討してください。
    - 特定のバスがボトルネックとなっている場合は、そのバス上の宛先が順不同イベントを許容できるかどうかを検討してください。この場合は、クラスター・ポリシーを変更することで、分割宛先を使用したワークロード・バランシングを許可できます。バスの分割には、メッセージング・エンジンのクラスター・メンバー間でのワーク・バランシングに関する考慮事項も含まれます。

  – データベース・サーバーがボトルネックとなる可能性があります。この問題を修正するには、次の手法を検討してください。

    - そのデータベース・サーバーが複数のアクティブなデータベース（BPEDB、twproc、twperfdb、MEDB など）をホストしている場合は、各データベースを別々のサーバー上でホストすることを検討してください。
    - 単一のデータベースが負荷を増大させている場合は、より強力なデータベース・サーバーを使用することを検討してください。
    - 上記の項目以外に、データベースのパーティショニング機能とクラスター化機能を使用できます。

4.4.6 Web サービスのチューニング

Web サービス・インポート・バインディングのターゲットが、同じアプリケーション・サーバー上でローカルにホストされている場合は、Web コンテナーによって提供される最適化された通信パスを利用することでパフォーマンスをさらに高めることができます。通常は、Web サービス・クライアントからの要求は、クライアントとサービス・プロバイダー間のネットワーク接続を介して送信されます。ただし、ローカル Web サービス呼び出しの場合は、WebSphere Application Server はネットワーク層を完全に迂回する直接通信チャネルを提供します。この最適化を有効にするには、次の手順を実行します。管理コンソールを使用して以下の値を変更します。

1. **「アプリケーション・サーバー」**をクリックして、希望のサーバーの名前を選択します。**「コンテナー設定」→「Web コンテナー設定」「Web コンテナー」→「追加プロパティー」→「カスタム・プロパティー」**をクリックします。Web コンテナーの enableInProcessConnections というカスタム・プロパティーを true に設定します。

   Web コンテナー・ポートのホスト名にワイルドカード (*) を使用しないでください。代わりにホスト名または IP アドレスを指定します。このプロパティーにアクセスするには、次のようにクリックします。

   「アプリケーション・サーバー」→「メッセージング・エンジン」→「コンテナー設定」→「Web コンテナー設定」→「Web コンテナー」→「追加プロパティー」→「Web コンテナー・トランスポート・チェーン」→「WCInboundDefault」→「TCP インバウンド・チャネル (TCP_2)」→「関連項目」→「ポート」→「WC_defaulthost」→「ホスト」

2. Web サービス・クライアント・バインディング内でホスト名の代わりに localhost を使用します。実際のホスト名を使用すると（そのホスト名が localhost の別名として設定されている場合でも同様）、この最適化が無効になります。ホスト名にアクセスするには、まず**「エンタープライズ・アプリケーション」**をクリックして、アプリケーション名を選択します。次に、**「モジュールの管理」**をクリックして、アプリケーションの EJB jar を選択します。最後に、**「Web サービス・クライアント・バインディング」→「優先ポート・マッピング」**をクリックして、バインディング名を選択します。URL 内で localhost（localhost:9080 など）を使用します。

3. サーバーのホスト名と IP アドレスのエントリーがお使いのサーバーのネーム解決用 hosts ファイルに存在しないことを確認します。hosts ファイルにエントリーが含まれている場合は、ネーム解決のためにプロセッサー使用量が増大することで、この最適化が阻害されます。

4.4.7 パワー・マネジメントのチューニング

パワー・マネジメントは、プロセッサー・テクノロジーで一般的になりつつあります。現在の Intel コア・プロセッサーと Power コア・プロセッサーはどちらもこの機能を備えています。この機能は明白な利点をもたらしますが、システムで大きな負荷が発生している場合はシステム・パフォーマンスを低下させる可能性もあるため、十分に考慮した上でパワー・マネジメントを有効にするかどうかを判断してください。例えば、IBM POWER6 ハードウェアを使用している場合は、省電力モードが有効になっていないことを確認してください（このモードを希望する場合は除く）。AIX でこの設定を変更または確認する方法の 1 つは、ハードウェア管理コンソール（HMC）の「パワー・マネジメント」ウィンドウを使用することです。

4.4.8 AIX のスレッディング・パラメーターの設定

IBM JVM のスレッディング・コンポーネントと同期コンポーネントは、AIX POSIX 準拠のスレッド実装に基づいています。以下の環境変数設定は、多くの場合に Java のパフォーマンスを向上させます。これらの変数は、Java スレッドから AIX ネイティブ・スレッドへのマッピングを制御して、マッピング情報を無効にして、相互排他（mutex）ロックでのスピニングをサポートします。

- **export AIXTHREAD_COND_DEBUG=OFF**

- **export AIXTHREAD_MUTEX_DEBUG=OFF**
- **export AIXTHREAD_RWLOCK_DEBUG=OFF**
- **export AIXTHREAD_SCOPE=S**
- **export SPINLOOPTIME=2000**

4.5 Business Process Execution Language ビジネス・プロセスのための Business Process Choreographer のチューニング

ここでは、Business Process Choreographer の高度なチューニング・ヒントを記載しています。

4.5.1 ビジネス・プロセス用の作業マネージャー・ベース・ナビゲーションのチューニング

作業マネージャー・ベースのナビゲーションは、BPEL ビジネス・プロセス用のデフォルト・ナビゲーション・モードです（もう一方は JMS ベースのナビゲーション）。作業マネージャー・ベースのナビゲーションは、パフォーマンスを最適化するための 2 つの方法を提供して、(JMS ベースの）プロセス・ナビゲーションのサービス品質を永続メッセージングと一致したものに保ちます。

- 作業マネージャー・ベースのナビゲーション

  WorkManager は、J2EE スレッドのスレッド・プールです。WorkManager プロセス・ナビゲーションは、WebSphere Application Server の基盤機能を利用して、JMS プロバイダーによって提供されるメッセージングを使用することなく、ナビゲーションの準備ができているビジネス・フロー・アクティビティーの処理を開始します。

- InterTransactionCache

  このキャッシュは、作業マネージャー・ベース・ナビゲーション・モードの一部として、プロセス・インスタンス状態情報をメモリー内に保持して、BPE データベースから情報を取得する必要性を低減します。

いくつかのパラメーターによって、これら 2 つの最適化機能の使用が制御されます。管理コンソールでこれらのパラメーターのうちの最初のセットにアクセスするには、**「アプリケーション・サーバー」**をクリックして、希望のサーバー名を選択します。次に、**「ビジネス・インテグレーション」→「Business Process Choreographer」→「Business Flow Manager」→「ビジネス・プロセスのナビゲーションのパフォーマンス」**をクリックします。

この機能をクラスター・レベルで有効にすると、個別サーバーの設定がオーバーライドされます。つまり、クラスター化環境では、最も簡単なのはこの機能をクラスター・レベルで有効にすることです。

主要なパラメーターは次のとおりです。

- **高度なパフォーマンスの最適化を使用可能にする**

  このプロパティーを選択すると、作業マネージャー・ベースのナビゲーションと InterTransactionCache という両方の最適化機能が有効になります。

- **作業マネージャー・ベースのナビゲーションのメッセージ・プール・サイズ**

  このプロパティーは、直ちに処理できないナビゲーション・メッセージ用のキャッシュのサイズを指定します（作業マネージャー・ベースのナビゲーションが有効になっている場合)。このキャッシュのデフォルト・サイズはメッセージ・サイズです (BPENavigationWorkManager のスレッド・プール・サイズの 10 倍)。このキャッシュが上限に達した場合は、JMS ベースのナビゲーションが新しいメッセージ用に使用されます。最適なパフォーマンスを得るためには、このメッセージ・プール・サイズが十分に大きい値に設定されている必要があります。

- **トランザクション間キャッシュ・サイズ**

このプロパティーは、BPE データベースにも書き込み済みであるプロセス状態情報を保管するためのキャッシュのサイズを指定します。この値は、並列実行されるプロセス・インスタンスの数の 2 倍に設定してください。このプロパティーのデフォルト値は、BPENavigationWorkManager のスレッド・プール・サイズです。

また、これらの設定を使用して作業マネージャー用のスレッドの数を調整するには、**「リソース」→「非同期 Bean」→「作業マネージャー」→「BPENavigationWorkManager」**をクリックします。

57 ページの 4.4.3、「並行性を最大化するためのチューニング」で概要を示している方法を使用して、スレッドの最小数をデフォルト値である 5 より大きくして、スレッドの最大数をデフォルト値である 12 より大きくします。スレッド・プール・サイズを変更した場合は、作業要求キュー・サイズも変更して、スレッドの最大数の 2 倍になるように設定します。

4.5.2 Java Message Service ナビゲーション用のビジネス・プロセス・コンテナーのチューニング

JMS ベースのナビゲーションが構成されている場合は、ビジネス・プロセスの効率的なナビゲーションのために次のリソースを最適化する必要があります。

- アクティベーション・スペック : BPEInternalActivationSpec

  「最大並行エンドポイント数」パラメーターは、すべてのプロセス・インスタンスにまたがるプロセス・ナビゲーションのために使用される並列性を指定します。このパラメーターの値を大きくすると、同時実行されるビジネス・プロセスの数が増えます。管理コンソールでこのリソースにアクセスするには、**「リソース」→「アクティベーション・スペック」→「BPEInternalActivationSpec」**をクリックします。

- JMS 接続ファクトリー : BPECFC

  接続プール・サイズを、BPEInternalActivationSpec 内のスレッドの数の 1.1 倍に設定します。管理コンソールでこのリソースにアクセスするには、**「リソース」→「JMS」→「接続ファクトリー」→「BPECFC」→「接続プール・プロパティー」**をクリックします。この接続ファクトリーは、作業マネージャー・ベースのナビゲーションが使用中の際にも使用されますが、エラー状況の場合やサーバーで大きな過負荷が生じている場合のみです。

4.5.3 タスク・リスト照会とプロセス・リスト照会のチューニング

プログラマーは、標準の照会 API（query() と queryAll()）および関連する REST サービス・インターフェースと Web サービス・インターフェースを使用して、BPM アプリケーションでタスク・リスト照会とプロセス・リスト照会を作成します。タスク・リスト照会とプロセス・リスト照会は、queryEntities() と queryRows() という照会テーブル API によっても作成されます。すべてのタスク・リスト照会とプロセス・リスト照会は、結果的に BPC データベースに対する SQL 照会を発行します。これらの SQL 照会は、応答時間を最短化するために次のように特別なチューニングを必要とすることがあります。

- 最新のデータベース統計情報は SQL 照会の応答時間を短縮するために重要です。
- データベースでは、SQL 照会をチューニングするためのツールが用意されています。ほとんどの場合は、索引を追加することで照会の処理速度が向上しますが、それに伴ってプロセス・ナビゲーションのパフォーマンスが影響を受ける可能性があります。DB2 の場合は、DB2 設計アドバイザーのガイドに従って索引を選択できます。

4.5.4 Business Process Choreographer API 呼び出しのチューニング

BPC API 呼び出しは、BPM サーバーの実行時外の要求によってトリガーされます。例として挙げられるのは、リモート EJB 要求、Web サービス要求、HTTP 経由の Web 要求、SCA 層を介して送信されてくる要求、JMS 要求などです。これらの各通信メカニズムと関連付

けられた接続プールは、チューニングを必要とすることがあります。これらの接続プールをチューニングする際は、次のヒントを参考にしてください。

- タスク・リスト照会とプロセス・リスト照会の API 呼び出しは、データベースのチューニング状況やデータベース内のデータ量によっては、応答時間が通常より長くなることがあります。
- 並行性（並列性）が、負荷を処理してプロセッサーを使用するのに十分なレベルであることを確認してください。ただし、API 呼び出し実行の並列性を必要以上に高めると、応答時間が長くなることがあります。また、並列性を高めると、BPC データベースに過剰な負荷が生じる可能性があります。API 呼び出しの並列性をチューニングする際は、チューニングの前後に応答時間を測定して、必要に応じて並列性を調整してください。

4.5.5 並行性のための中間コンポーネントのチューニング

入力ビジネス・オブジェクトが単一のスレッドによって全体的に処理される場合は、通常はエッジ・コンポーネントのチューニングで十分です。しかし多くの場合は、エンドツーエンドの実行パス中に複数のスレッド切り替えが存在します。したがって、実行パスの各非同期セグメントについて十分な並行性を確保するために、システムをチューニングすることが重要です。

SCA コンポーネントの非同期呼び出しは MDB を使用して、関連付けられた入力キュー内に届く着信イベントを listen します。各 SCA モジュールは MDB およびその対応するアクティベーション・スペック（JNDI 名 : sca/*module name*/ActivationSpec）を定義します。SCA モジュールの MDB は、そのモジュール内のすべての非同期 SCA コンポーネント（SCA エクスポート・コンポーネントを含む）によって共有されます。ActivationSpec の maxConcurrency プロパティーを構成する際は、この共有状態を考慮に入れてください。SCA モジュールの MDB は、JMS エクスポートのものと同じデフォルト・スレッド・プールを使用します。

長時間実行されるビジネス・プロセス内の非同期性は、トランザクション境界で発生します（トランザクション境界に影響を与える設定について詳しくは、37 ページの 3.2.6、「トランザクション上の考慮事項」を参照してください）。BPE は、内部 MDB とその ActivationSpec を BPEInternalActivationSpec として定義しています。SCA モジュールと JMS エクスポートの MDB と同じ指針に従って、**maxConcurrency** パラメーターをチューニングする必要があります（詳しくは 57 ページの 4.4.3、「並行性を最大化するためのチューニング」を参照してください）。唯一の注意点は、単一の BPM サーバーについて 1 つの BPEInternalActivationSpec しか存在しないことです。

4.6 Business Processing Modeling Notation ビジネス・プロセスのチューニング

ここでは、一般的なチューニング指針に加えて、BPMN ビジネス・プロセスに固有のチューニング指針を記載しています。ここで紹介するチューニングの多くは、Performance Server と SIB のデータベースに当てはまります。

4.6.1 データベースのチューニング

以下では、パフォーマンスを高めるためのデータベース・チューニング方法を列挙しています。

- twproc データベースのログ・ファイル・サイズを大きくします（16,384 ページに設定）。
- 次のコマンドを実行して twproc データベースのファイル・システム・キャッシングを有効にします。
  db2 alter table space userspace1 file system caching
- 次の技術情報に従って、SIBOWNER テーブルを自動 runstats 実行の対象から除外します。

http://www-01.ibm.com/support/docview.wss?uid=swg21452323

- データベース統計情報が最新であることを確認します。

  データベースが高いスループット率で使用されている場合は、ピーク・スループットへの影響を避けるために、データベースの自動保守タスクをいくつかまたはすべて無効にすることを検討してください。ただし、これらの機能を無効にした場合は、必ず runstats を定期的に実行してデータベース統計情報を更新してください。

- 次の技術情報に従って、新しい索引を作成します。

  http://www-01.ibm.com/support/docview.wss?uid=swg21474536

  DB2 を使用している場合は、次のコマンドを使用して索引を作成します（他のデータベースにも同様のコマンドが用意されています）。

  db2 "CREATE INDEX IDX_SOAB_BPD_INSTANCE ON LSW_BPD_INSTANCE (SNAPSHOT_ID ASC, BPD_INSTANCE_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS”

  db2 "CREATE INDEX IDX_SOAB_BPD_INSTANCE_VAR ON LSW_BPD_INSTANCE_VARIABLES (BPD_INSTANCE_ID ASC, VARIABLE_TYPE ASC, ALIAS ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS"

4.6.2 ビジネス・プロセス定義のキュー・サイズとワーカー・スレッド・プール

高いスループット率で実行している場合は、必要に応じて**「BPD Queue Size」**パラメーターと**「Worker Thread Pool」**パラメーターをそれぞれのデフォルト値より大きい値に設定してください（「BPD Queue Size のデフォルト値は 20、「Worker Thread Pool」のデフォルト値は 40）。これらの値は、プロセス・サーバーの構成ディレクトリー内の 80EventManager.xml ファイルで設定されています。具体的な構成ディレクトリーは次のとおりです。

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

これらの値を変更するには、このファイルを直接編集します。以下に、これらのパラメーターのチューニングに関する指針を示します。

- 目安として、物理プロセッサー・コアあたり 10 という BPD キュー・サイズ（**bpd-queue-capacity**）を最初に設定します（例えば 4 プロセッサー・コア構成の場合は 40）。その後で、システムのパフォーマンスに基づいて、必要に応じてチューニングを加えます。
- ワーカー・スレッド・プール・サイズ（**max-thread-pool-size**）は、4 つのキュー・サイズ（BPD、同期、非同期、およびシステム）すべての合計より常に大きくする必要があります。

4.7 WebSphere ESB のチューニング

以下では、WebSphere ESB のチューニングに関連する追加の構成オプションを紹介します。推奨される一連の初期値については、99 ページの 5.2、「WebSphere ESB の設定」を参照してください。

4.7.1 永続メッセージングを使用している場合のデータベースのチューニング

永続メッセージングを使用している場合は、データベースの構成は重要です。高速なディスク・アレイを備えたリモート DB2 インスタンスをデータベース・サーバーとして使用してください。データ・ソースのステートメント・キャッシュと接続プーリングをチューニングすることで利点が得られる場合があります。

DB2 のチューニングについて詳しくは、以下のセクションを参照してください。

- 77 ページの 4.10、「一般的なデータベース・チューニング」
- 79 ページの 4.11、「DB2 固有のデータベース・チューニング」

101 ページの『関連資料』で紹介している関連資料を参照してください。

4.7.2 Common Event Infrastructure のイベント配布の無効化

イベントを管理するイベント・サーバーは、イベントを配布してこれらのイベントをイベント・データベースに記録するように構成できます。一部のメディエーションでは、イベントがデータベースに記録されることのみが必要です。このような場合は、イベント配布を無効にすることでパフォーマンスが向上します。イベント・サーバーは他のアプリケーションによって使用されている可能性があるため、イベント配布を無効にする前に、イベント配布を必要とするイベント・モニタリングがこれらのアプリケーションのいずれによっても使用されていないことを確認することが重要です。

管理コンソールでイベント配布を無効にするには、**「統合サービス」→「Common Event Infrastructure」→「イベント・サービス」→「イベント・サービス」→「デフォルトの Common Event Infrastructure イベント・サーバー」**をクリックします。**「イベント配布の使用可能化」**チェック・ボックスを選択解除します。

4.7.3 WebSphere Service Registry and Repository のキャッシュ・タイムアウトの構成

WebSphere Service Registry and Repository（WSRR）は、エンドポイント・ルックアップのために WebSphere ESB によって使用されます。WSRR にアクセスする際は（例えばエンドポイント・ルックアップのメディエーション・プリミティブを使用して）、レジストリー内の結果は WebSphere ESB のキャッシュに格納されます。管理コンソールでキャッシュ内のエントリーの存続時間を構成するには、**「統合サービス」→「WSRR 定義」**をクリックして、WSRR 定義名を入力します。次に、**「キャッシュのタイムアウト」**をクリックして、キャッシュ内のレジストリー結果の値を設定します。

このタイムアウトが十分に大きい値であることを確認してください。デフォルトのタイムアウトである 300 秒は、パフォーマンスの観点からは適切な値です。値が小さすぎると、WSRR に対するルックアップが頻繁に実行されて、大きな負荷が発生したり（特に結果リストを取得する場合）、レジストリーがリモート・マシン上にある場合はネットワーク待ち時間が発生したりする可能性があります。

4.8 Business Monitor のチューニング

ここでは、Business Monitor の高度なチューニング・ヒントを記載しています。

4.8.1 Java ヒープ・サイズの構成

ほとんどの Java 実装環境でのデフォルトの最大ヒープ・サイズは、この構成内のサーバーの多くにとって小さすぎます。Business Monitor ランチパッドでは、デフォルトより大きいヒープ・サイズを設定して Business Monitor とその必須サーバーがインストールされますが、これらのサイズがお使いのハードウェアやワークロードにとって適切であることを確認してください。

4.8.2 Common Event Infrastructure の構成

デフォルトでは、CEI に着信したイベントは、登録済みのコンシューマー（この場合は特定のモニター・モデル）と追加のデフォルト・キューに送達されます。管理コンソールを使用して「すべてのイベント」イベント・グループを削除してこのような二重保管を回避することで、パフォーマンスを向上させることができます。**「統合サービス」→「Common Event Infrastructure」→「イベント・サービス」→「イベント・サービス」→「デフォルトの Common Event Infrastructure イベント・サーバー」→「イベント・グループ」**をクリックして、このイベント・グループを削除します。

CEI は、登録済みコンシューマーへの永続イベント送達に加えて、イベントをデータベースに明示的に保管する機能を提供します。データベースへのイベント保管は多大なプロセッサー・リソースを必要とするため、この追加機能が不要な場合はデータベースにイベントを保管することは避けてください。管理コンソールで CEI のデータ・ストアを構成するには、**「統合サービス」→「Common Event Infrastructure」→「イベント・サービス」→「イベント・サービス」→「デフォルトの Common Event Infrastructure イベント・サーバー」**をクリックします。**「データ・ストアの使用可能化」**チェック・ボックスを選択解除します。

4.8.3 メッセージ消費バッチ・サイズの構成

複数のイベントを大きなバッチで処理する方が、1 つずつ処理するよりもはるかに効率的です。特定の限度までは、バッチ・サイズが大きくなるほど、イベント処理のスループットは高まります。しかし、これには利点と欠点が存在します。イベントの処理とモニター・データベースへの永続化は、1 つのトランザクションとして実行されます。バッチ・サイズが大きいほど、スループットは向上しますが、ロールバックが必要になった場合の負荷も大きくなります。ロールバックが頻繁に実行される場合は、バッチ・サイズを小さくすることを検討してください。サーバー・スコープで管理コンソールを使用してキャッシュ・サイズを小さくできます。**「アプリケーション」→「モニター・モデル」**をクリックして、バッチのバージョンを選択します。次に、**「ランタイム構成」→「チューニング」→「メッセージ消費のバッチ・サイズ」**をクリックして、希望のバッチ・サイズを設定します。デフォルト値は 100 です。

4.8.4 重要業績評価指標のキャッシングの有効化

完了済みのプロセス・インスタンスがデータベース内に蓄積するにつれて、合計の重要業績評価指標（KPI）値の計算に伴う負荷は大きくなります。KPI キャッシュを使用してこれらの計算のリソース使用量を低減できますが、その代わりに結果内で不整合状態が発生します。更新間隔は、WebSphere 管理コンソールで構成できます。**「アプリケーション」→「モニター・モデル」**をクリックして、バージョンを選択します。次に、**「ランタイム構成」→「KPI」→「KPI キャッシュ・リフレッシュ間隔」**をクリックします。

デフォルト値である 0 を使用すると、キャッシュが無効になります。

4.8.5 表ベースのイベント送達の使用

CEI からモニター・モデルにイベントを送達するには、次の 2 つの方法があります。

- JMS キュー経由
- データベースの表経由

モニター・モデルのアプリケーション・インストール時に、いずれかの方法を選択できます。信頼性、パフォーマンス、およびスケーラビリティーの観点から、表ベースのイベント送達（キュー・バイパスとも呼ばれます）を選択することをお勧めします。

4.8.6 データ移動サービスの有効化

デフォルトでは、イベント処理とダッシュボード・レポートには同じテーブルが使用されます。オプションの予約済みサービスであるデータ移動サービス（DMS）を有効にすることで、別個のテーブル・セットに対してダッシュボード・レポートを切り替えることができます。DMS は、サーバーのイベント処理テーブルからダッシュボード・レポート・テーブルにデータを定期的にコピーもします。この処理とレポート・モードによって次のコンポーネントが最適化されます。

- 各テーブルの索引
- 迅速な挿入 / 更新 / 削除のためのサーバーのイベント処理テーブル
- 一般的なダッシュボード関連照会のためのダッシュボード・レポート・テーブル

どのような実稼働環境でも、DMS を有効にすることをお勧めします。

4.9 Business Space のチューニング

Business Space は AJAX（Asynchronous JavaScript and XML）アプリケーションであるため、そのコードの大部分は JavaScript を通じてブラウザーで実行されます。したがって Business Space のパフォーマンスは、ブラウザーとその実行場所であるクライアント・システムに大きく依存します。Business Space で応答時間を短縮するには、必要に応じてサーバーとクライアントの設定を調整してください。ここでは、Business Space ソリューションの主要なチューニング・パラメーターを紹介します。

4.9.1 Internet Explorer 8 でのブラウザー・キャッシュの有効化

デフォルトでは、Business Space では、ブラウザーがスタイル・シートや JavaScript ファイルなどの静的情報を 24 時間だけキャッシュに保存することが許可されます。ただし、ブラウザーがデータをキャッシュに保存しないように構成されている場合は、同じリソースが何度も繰り返し読み込まれるため、ページの読み込み時間が長くなります。キャッシングを有効にするには、お使いのブラウザーで次の手順を実行します。

- Internet Explorer
  a. **「ツール」→「インターネット オプション」→「全般」→「閲覧の履歴」**にアクセスします。
  b. キャッシュ・サイズを表す**「使用するディスク領域」**の値を 50 MB 以上に設定します。
- Firefox
  a. **「ツール」→「オプション」→「詳細」→「ネットワーク」→「オフラインデータ」**にアクセスします。
  b. オフライン記憶域を 50 MB 以上に設定します。

ログアウト時にブラウザー・キャッシュが消去されることの防止

最新のブラウザーには、終了時にキャッシュを消去するためのオプションが用意されていますが、このオプションが有効になっていないことを確認します。以下では、Internet Explorer 8 と Firefox の手順を示しています。

次の手順に従って、Internet Explorer 8 でキャッシュが消去されないようにします。

1. **「ツール」→「インターネット オプション」**をクリックします。
2. **「全般」**タブで、**「終了時に閲覧の履歴を削除」**チェック・ボックスが選択解除されていることを確認します（図 4-1 を参照）。

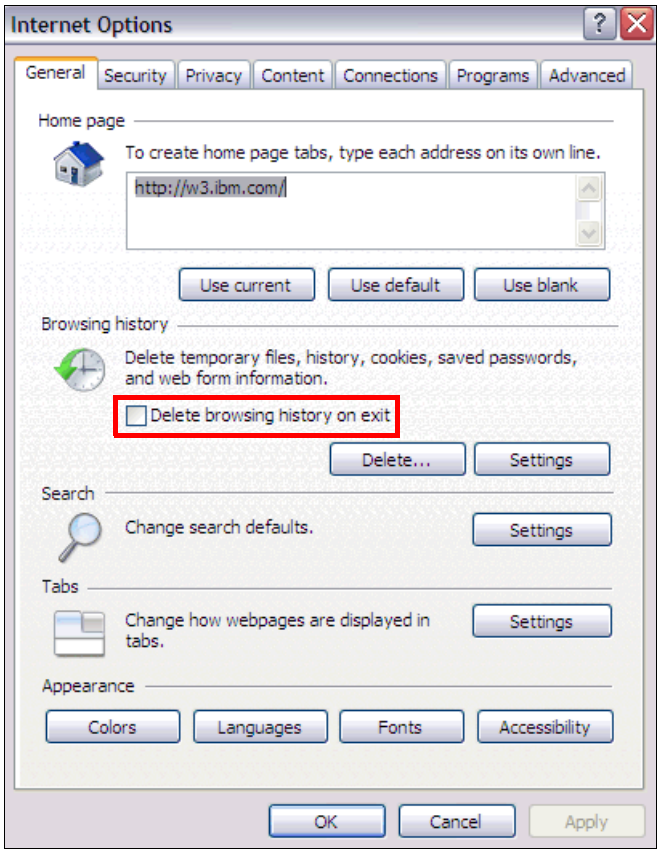

図 4-1　Internet Explorer 8 の「閲覧の履歴を削除」オプション

3. **「詳細設定」**タブをクリックして、**「ブラウザーを閉じたとき、[Temporary Internet Files] フォルダーを空にする」**チェック・ボックスが選択解除されていることを確認します（図 4-2 を参照）。**「OK」**をクリックして設定を保存します。

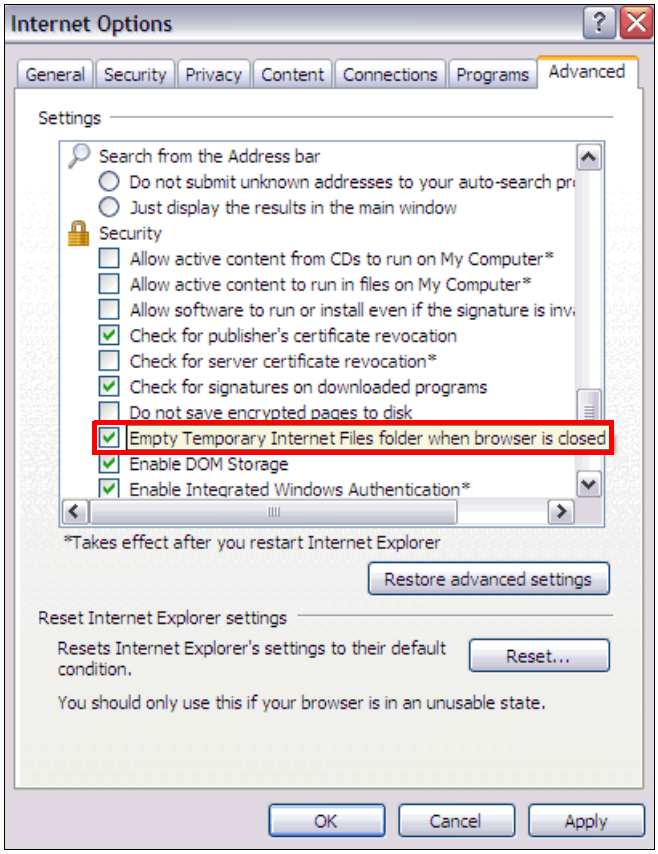

図 4-2　Internet Explorer 8 の「[Temporary Internet Files] フォルダーを空にする」オプション

4. ブラウザーを再起動して、これらの変更内容を有効にします。

4.9.2 Firefox 3.6 でのブラウザー・キャッシュの有効化

次の手順に従って、Firefox 3.6 でキャッシュが消去されないようにします。

1. **「ツール」→「オプション」**をクリックします。
2. **「プライバシー」**タブをクリックして、**「Firefox に　履歴を記憶させる」**と表示されていることを確認します（図 4-3 を参照）。**「OK」**をクリックします。この設定によってキャッシングが有効になります。

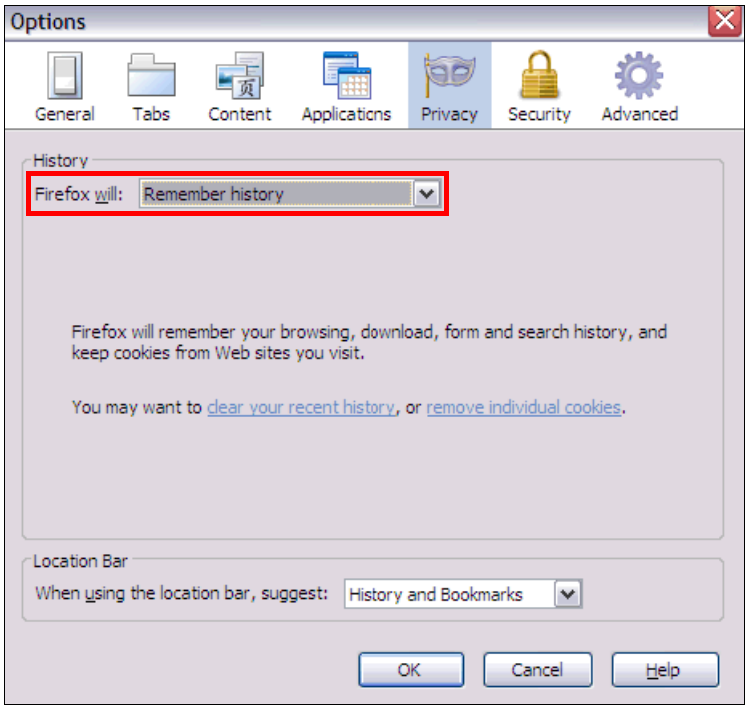

図 4-3　Firefox 3.6 の「履歴を記憶させる」オプション

「履歴を一切記憶させない」を選択した場合は、キャッシングは無効になるため、この設定を変更する必要があります。

「記憶させる履歴を詳細設定する」を選択した場合は、追加のオプションによってキャッシングを有効にできます（図 4-4 を参照）。

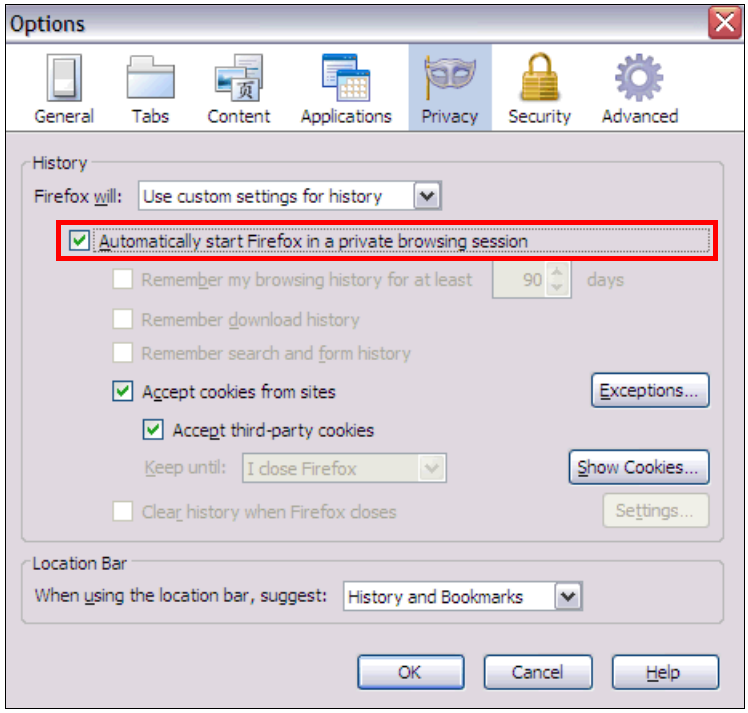

図 4-4　Firefox 3.6 の「記憶させる履歴を詳細設定する」オプション

以下のオプションによっても Firefox でキャッシングを有効にできます。

- 「**常にプライベートブラウジングモード**」チェック・ボックスを選択した場合は、キャッシングは有効になります。ただし、ブラウザーを閉じると、すべて消去されて（キャッシュだけでなく閲覧の履歴も）、ブラウザーをまったく使用していなかったのと同じ状態になります。
- 上記のチェック・ボックスが選択されていない場合は、「**Firefox の終了時に履歴を消去する**」チェック・ボックスが選択解除されていることを確認します（図 4-5 を参照）。

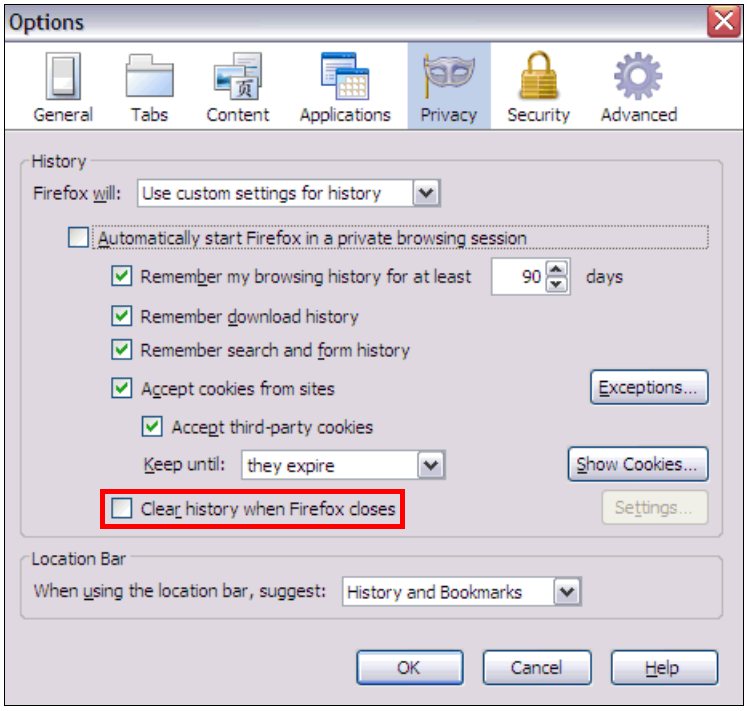

図 4-5　Firefox 3.6 の「履歴を消去する」オプション

3. ブラウザーを再起動して、これらの変更内容を有効にします。

4.9.3 複雑なタスク・メッセージのパフォーマンスの最適化

「タスク情報」ウィジェットを通じて、大きくて複雑な入力メッセージが付随するタスクを開くときは、ブラウザーでそのタスクをレンダリングして表示するのに長い時間がかかることがあります。この問題が発生している場合は、次の技術情報を参照してください。

http://www-01.ibm.com/support/docview.wss?uid=swg21468931&wv=1

4.9.4 HTTP サーバーのチューニング

実稼働環境内の Business Space は通常、HTTP サーバーまたは HTTP サーバー・プラグインが含まれたトポロジーを使用して導入されます。AJAX クライアントとしての Business Space は、当然ながら複数の要求をサーバーに送信して、これらの要求はこの HTTP サーバーによって処理されます。ユーザー応答時間を短縮するためには、HTTP サーバーをチューニングして、想定される負荷が HTTP サーバーによって効率的に処理されるようにしてください。

BPM V7.5 の一部として含まれている IBM HTTP Server のチューニングについて詳しくは、次の『*IBM HTTP Server Performance Tuning*』というページを参照してください。

http://publib.boulder.ibm.com/httpserv/ihsdiag/ihs_performance.html

Business Space のチューニングについて詳しくは、次の『*Scalability and Performance of Business Space and Human Task Management Widgets in WebSphere Process Server v7*』というホワイト・ペーパーを参照してください。

http://www-01.ibm.com/support/docview.wss?uid=swg27020684&wv=1

4.9.5 フェデレーション・モード非使用時のパフォーマンスの最適化

WebSphere Process Server（WPS）V7.0 と組み合わせたヒューマン・ワークフロー・ウィジェットのパフォーマンス特性に精通している場合は、BPM V7.5 で「タスク」、「プロセス」、「タスク定義」などのヒューマン・ワークフロー・ウィジェットを使用しているときに、最初はパフォーマンスの低下を感じる可能性があります。このパフォーマンス低下が明白になるのは、新たにインストールした BPM V7.5（マイグレーションしたものではなく）で Business Space を使用している場合のみです。これらのパフォーマンス考慮事項が当てはまるのは、単一のアプリケーション・サーバーやクラスターで、ビジネス・プロセス定義（BPD）のプロセスやヒューマン・サービスのみと組み合わせてヒューマン・タスク管理ウィジェットを使用する場合です。

このパフォーマンス低下の原因は、フェデレーション・サービス層の追加および取得対象アイテムの数の増加です。フェデレーション・サービス層の追加は、BPM V7.5 での使いやすさを向上させるものであり、すべてのタスクを単一の統一タスク・リストに表示することを可能にします。

BPM アップグレード時の WPS のパフォーマンス : BPM V7.5 を使用した場合の WPS のパフォーマンス低下は、WPS V7 からのアップグレードやマイグレーションには当てはまりません。

管理コンソールで Business Space のターゲット・サービスの構成を確認するには、次の手順を実行します。

1. IBM Business Process Manager Integrated Solutions Console を開きます。
2. ナビゲーション・バーの**「サーバー」**タブをクリックします。
3. **「WebSphere」**をクリックします。
4. **「アプリケーション・サーバー」**というサーバー・タイプをクリックして、Business Space を実行する場所となるアプリケーション・サーバーを選択します。
5. **「構成」**タブの「ビジネス・インテグレーション」セクションで、**「Business Space の構成」**エントリーをクリックします。
6. 「Business Space の構成」ページの「追加プロパティー」セクションで、**「REST サービス・エンドポイント登録」**エントリーをクリックします。
7. 「REST サービス・エンドポイント登録」ページで、「プロセス・サービス」と「タスク・サービス」という REST エンドポイント・タイプの「サービス・エンドポイント・ターゲット」の値を確認します。デフォルトの初期設定は、**「フェデレーテッド REST サービス」**です。

フェデレーテッド REST エンドポイントの代わりに BPC REST エンドポイントを使用することでパフォーマンスを向上させるには、次の手順を実行します。

1. 次の手順に従って、管理コンソールで Business Space のターゲット・サービスの構成を変更します。
2. 「プロセス・サービス」を、お使いのプロセスとヒューマン・タスクを実行しているアプリケーション・サーバーの Business Process Choreographer REST サービス・エンドポイントに変更します。
3. 「タスク・サービス」を、お使いのプロセスとヒューマン・タスクを実行しているアプリケーション・サーバーの Business Process Choreographer REST サービス・エンドポイントに変更します。
4. **「OK」**をクリックして構成の変更内容を保存します。
5. Business Space からログアウトして、再びログインします。

単一のサーバーやクラスターの BPD プロセスおよびヒューマン・サービスの使用 : 単一のアプリケーション・サーバーまたはクラスターの BPD プロセスおよびヒューマン・サービスのみを使用するには、「プロセス・サービス」と「タスク・サービス」を、お使いの BPD プロセスおよびヒューマン・サービスを実行しているアプリケーション・サーバーのビジネス・プロセス定義エンジン REST サービスに変更します。

4.10 一般的なデータベース・チューニング

ここでは、データベースの一般的なチューニング・ヒントを記載しています。

4.10.1 最適化のための十分な統計情報の提供

データベースでは通常、最適なデータ・アクセス手法を決定する際に幅広い選択肢が用意されています。データの形態を示す統計情報を参考にして、低負荷のデータ・アクセス手法が選択されます。統計情報は、テーブル内と索引内に保持されます。統計情報の例としては、テーブル内の行数や特定列内の非重複値の数などが挙げられます。

統計情報の収集には大きな負荷が伴うことがありますが、幸いにも、多くのワークロードについては、一連の代表的な統計情報によって長期間にわたる高いパフォーマンスが実現されます。データの母集団が大幅に変化した場合は、必要に応じて統計情報を定期的に更新してください。

4.10.2 高速なディスク・サブシステム上へのデータベース・ログ・ファイルの配置

データベースは、高可用性、トランザクション処理、およびリカバリー可能性をサポートできるように設計されています。パフォーマンス上の理由から、表データに加えられた変更内容は、直ちにディスクに書き込まれない可能性があります。これらの変更内容は、データベース・ログに書き込まれている場合はリカバリーできます。データベース・ログ・ファイルが更新されるのは、ログ・バッファーがいっぱいになったとき、トランザクションのコミット時、および一部の実装については最大時間間隔の経過後です。

上記の理由から、データベース・ログ・ファイルは頻繁に使用される可能性があります。より重要なこととして、ログへの書き込み時はコミット操作が保留されるため、アプリケーションは書き込みが完了するまで同期的に待機状態になります。したがって、データベース・ログ・ファイルへの書き込みアクセスのパフォーマンスは、システム・パフォーマンス全体にとって非常に重要です。このため、データベース・ログ・ファイルは、ライトバック・キャッシュを備えた高速なディスク・サブシステム上に配置することをお勧めします。

4.10.3 表スペース・コンテナーとは別個のデバイス上へのログの配置

すべてのデータベース・ストレージ構成を対象にした基本的な方針として、データベース・ログを専用の物理ディスク（理想的には専用のディスク・アダプター）上に配置してください。このように配置することで、表スペース・コンテナーに対する入出力とデータベース・ログに対する入出力の間のディスク・アクセス競合が軽減されるとともに、ログ・ストリームのほとんど順次的なアクセス・パターンが保持されます。このように分離することで、アーカイブ・ログが使用されているときのリカバリー可能性も向上します。

4.10.4 十分な物理メモリーの提供

メモリー内のデータへのアクセスは、ディスクからのデータ読み取りよりはるかに高速です。64 ビット・ハードウェアは普及しており、メモリー価格は下落し続けているため、パフォーマンスにとって重要な多くのワークロードについて定常状態でほとんどのディスク読み取りを回避するために、十分なメモリーを用意することが適切です。

データベース・マシン上の仮想メモリー・ページングを回避するように注意してください。データベースは、自身が一切ページングされないことを前提にして自身のメモリーを管理しており、オペレーティング・システムによってデータベースの一部のページがディスクにスワッピングされる場合は、データベースは適切に連携動作しません。

4.10.5 ダブル・バッファリングの回避

データベースは頻繁にアクセスされるデータをメモリー内に保持しようとするため、ほとんどの場合は、ファイル・システム・キャッシングを使用しても何の利点も得られません。ただし通常は、直接入出力（direct I/O）を使用することでパフォーマンスが向上します。直接入出力では、データベースによって読み取られたファイルはファイル・システム・キャッシュを経由せず、そのデータの 1 つのコピーのみがメモリー内に保持されます。直接入出力を使用すると、システムはデータベースに通常より多くのメモリーを割り振ることができるとともに、ファイル・システムのキャッシュ管理に伴うリソース使用が回避されます。

AIX などの一部のオペレーティング・システムでは、同時入出力（concurrent I/O）を使用することでさらなる利点が得られます。同時入出力を使用すると、ファイルごとのロックが回避されて、同時制御の責任がデータベースに移管されて、場合によっては、アダプターやデバイスに対して通常より有用な作業を割り当てることができます。

この指針の重要な例外は、データベース自体によってバッファリングされないラージ・オブジェクト（LOB、BLOB、CLOB など）について発生します。この場合は、理想的にはラージ・オブジェクトをサポートするファイルのみについて、ファイル・システム・キャッシングを有効にすると利点が得られる可能性があります。

4.10.6 必要に応じた表索引の詳細化

BPM 製品は通常、それらの製品で使用されるデータベース・テーブルの適切な索引セットを提供しています。一般に、索引を作成する際は、照会の処理負荷と、データを挿入、更新、または削除するステートメントの処理負荷の間のトレードオフを考慮する必要があります。照会が多用されるワークロードについては、高速なデータ・アクセスを可能にするために、必要に応じてさまざまな索引を提供することが適切です。多くの更新が実行されるワークロードについては、多くの場合、行が変更されるたびに複数の索引を変更する必要性が生じる可能性があるため、定義される索引の数をできる限り少なくする方が望ましいです。索引は、頻繁に使用されない場合でも最新状態に保たれます。

したがって、索引の設計には妥協が伴います。デフォルトの索引セットは、状況によっては、BPM 製品によって生成されるデータベース・トラフィックに対して最適ではない可能性があります。データベースのプロセッサー使用量またはディスク使用量が多い場合や、データベースの応答時間に問題がある場合は、索引を変更することを検討するとよいでしょう。

DB2 データベースと Oracle データベースは、特定のワークロードのコンテキストで索引を分析することで、索引に関する作業を支援します。これらのデータベースは、索引の追加、変更、または削除を提案します。ただし注意点として、ワークロードによってすべての関連するデータベース・アクティビティーがキャプチャーされない場合は、ある索引が必要であるにもかかわらず使用されていないと見なされる可能性があり、その結果としてその索引を削除することが提案されます。その索引がなくなると、結果的に将来のデータベース・アクティビティーで問題が発生する可能性があります。

4.10.7 完了済みのプロセス・インスタンスのアーカイブ

時間の経過に伴って、完了済みのプロセス・インスタンスがサーバーのデータベース内に蓄積します。この蓄積によって、測定対象のソリューションのパフォーマンス特性が変化する可能性があります。したがって、データベースへのデータ取り込みが確実に制御されるために、完了済みのプロセス・インスタンスをアーカイブすることが望ましいです。完了済みのプロセス・インスタンスと Performance Data Warehouse イベントをアーカイブしない場合は、これらが時間の経過とともに無制限に増大して、全体的なシステム・パフォーマンスが低下します。

この問題の症状の 1 つは、ビジネス・プロセス定義（BPD）テーブルとタスク・テーブルで照会時間が長くなることです。もう 1 つの症状は、プロセス・サーバーのデータベース・テーブルが過大なディスク・スペースを占有することです。これらの症状はどちらも、完了済みの BPD インスタンスがシステムから自動的に削除されないことに起因します。BPD インスタンスが完了したら、そのインスタンスは通常は不要になるため、Process Server データベースから削除できます。BPM で用意されている LSW_BPD_INSTANCE_DELETE というストアード・プロシージャーを使用して、古いインスタンスを削除できます。アーカイブ・プロシージャーについては、次の技術情報を参照してください。

http://www-01.ibm.com/support/docview.wss?uid=swg21439859

4.11 DB2 固有のデータベース・チューニング

DB2 の包括的なチューニング・ガイドを提供することは、本書の範囲外です。ただし、いくつかの基本原則を知っておけば、DB2 環境のパフォーマンスを向上させるのに役立つ可能性があります。以下のセクションでは、これらの原則を説明して、詳細情報を紹介します。

DB2 の最新資料一式については（データベース・チューニング指針を含む）、DB2 インフォメーション・センターを参照してください。

http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp

もう 1 つのリファレンス情報として、次のアドレスにある『*Best practices for DB2 for Linux, UNIX, and Windows*』というページを参照してください。

http://www.ibm.com/developerworks/data/bestpractices/db2luw/

4.11.1 データベース統計情報の更新

DB2 は、**RUNSTATS** コマンドを必要に応じてバックグラウンドで実行する自動表保守機能を備えています。**RUNSTATS** を実行することで、正しい統計情報が確実に収集されて保持されます。この機能は、**auto_runstats** というデータベース構成パラメーターによって制御され、DB2 9.1 以降で作成されたデータベースについてはデフォルトで有効になります。DB2 コントロール・センターで、データベース・レベルの自動保守の構成ウィザードを参照してください。

データベース内のすべての表で統計情報を手動で更新する方法の 1 つは、**REORGCHK** コマンドを使用することです。動的 SQL（JDBC によって作成されるものなど）では、新しい統計情報が直ちに反映されます。静的 SQL（ストアード・プロシージャー内のものなど）は、新しい統計情報のコンテキストで明示的に再バインドされる必要があります。例 4-2 には、DBNAME というデータベース上で基本的な統計情報を収集するためのステップを実行する DB2 コマンドの例を示しています。

例 4-2 DBNAME データベース上の基本的な統計情報の収集

```
db2 connect to DBNAME
db2 reorgchk update statistics on table all
db2 connect reset
db2rbind DBNAME all
```

システムが比較的アイドル状態のときは、安定したサンプルが取得されるようにするために、およびカタログ表で発生する可能性のあるデッドロックを回避するために、**REORGCHK** と **rebind** を実行してください。

追加の統計情報を収集すると役に立つため、注目する必要のあるすべての表について次のコマンドを使用することも検討してください。

runstats on table <*schema*>.<*table*> with distribution and detailed indexes

4.11.2 バッファー・プール・サイズの正しい設定

バッファー・プールとは、処理時にデータベース・ページが読み込まれたり、変更されたり、保持されたりするメモリー領域です。バッファー・プールは、データベースのパフォーマンスを向上させます。目的のデータ・ページがバッファー・プール内に既に存在する場合は、そのページをディスクから直接読み取る必要がある場合よりも高速にそのページにアクセスできます。したがって、DB2 のバッファー・プールのサイズはパフォーマンスにとって非常に重要です。

バッファー・プールによって使用されるメモリー量は次の 2 つの要因によって決まります。

- バッファー・プール・ページのサイズ
- 割り振られたページの数

バッファー・プール・ページは、作成時に 4、8、16、または 32 KB という固定サイズに設定されます。最もよく使用されるバッファー・プールは IBMDEFAULTBP であり、そのページ・サイズは 4000 MB に設定されています。

すべてのバッファー・プールは、データベース・マシン上で割り振られたデータベース・グローバル・メモリー内に存在します。これらのバッファー・プールと他のデータ構造体およびアプリケーションは、いずれも使用可能メモリーを使い果たすことなく、共存する必要があります。一般に、バッファー・プールのサイズが大きいほど、入出力アクティビティーが低減されて、一定のレベルまではパフォーマンスが向上します。そのレベルに達すると、追加のメモリーを割り振っても、パフォーマンスは向上しなくなります。

DB2 9.1 以降は、バッファー・プール・サイズの管理を含むセルフチューニング・メモリー管理機能を備えています。**self_tuning_mem** というデータベース・レベル・パラメーターは、デフォルトで有効になっており、メモリー管理をグローバルに制御します。個別のバッファー・プールに対してセルフチューニングを有効にするには、CREATE 時または ALTER 時に **SIZE AUTOMATIC** コマンドを使用します。

適切なバッファー・プール・サイズ設定を手動で選択するには、システム・ツールや DB2 のバッファー・プール・スナップショットを使用して、データベース・コンテナーの入出力アクティビティーをモニターします。システム上のページング・アクティビティーの原因となる大きいバッファー・プール・サイズを設定しないように注意してください。

4.11.3 正しい表索引付けの管理

DB2 設計アドバイザーは、スキーマの変更（索引の変更を含む）について提案します。設計アドバイザーを開くには、次の手順を実行します。

1. コントロール・センターで、左側の列のデータベースを右クリックします。
2. 表示されるメニューで「**DB2 Design Advisor**」をクリックします。

4.11.4 ログ・ファイルの適切なサイズ設定

循環ロギングを使用している場合は、バッファー・プール内のダーティー・ページがあまり頻繁にクリーニングされなくても済むように、十分な大きさの使用可能ログ・スペースが存在することが重要です。データベースに対する変更は直ちにログに書き込まれますが、適切にチューニングされたデータベースは、単一ページに対する複数の変更を結合してから、最終的にその変更されたページをディスクに書き戻します。当然ながら、ログのみに記録された変更内容を循環ロギングによって上書きすることはできません。DB2 はこの状況を検知して、新しいログ・ファイルへの切り替えを可能にするために必要なダーティー・ページの即時クリーニングを強制的に実行します。このメカニズムは、ログに記録された変更内容を保護しますが、必要なページがクリーニングされるまですべてのアプリケーション・ロギングを中断します。

DB2 は、**softmax** というデータベース・レベル・パラメーターの制御下でページ・クリーニングを先行的に実行することで、ログ・ファイルの切り替えに伴う中断を回避しようとします。**softmax** のデフォルト値である 100 を使用した場合は、現在のログ先頭と、ダーティー・ページの変更内容を記録している最も古いログ・エントリーの間のサイズ・ギャップが 1 つのログ・ファイルの 100% を超えたときに、バックグラウンドでのクリーニングが開始されます。極端なケースでは、この非同期ページ・クリーニングがログ・アクティビティーについて行けずに、ログ切り替えに伴う中断が発生してパフォーマンスが低下することがあります。

使用可能なログ・スペースを大きくすると、非同期ページ・クリーニングはより長い時間をかけてバッファー・プールのダーティー・ページを書き出すことができ、ログ切り替えに伴う中断が回避されます。クリーニング間の間隔が長くなると、単一ページの複数の変更を結合してからそのページを書き込むことが可能になり、その結果としてページ・クリーニングの効率性が高まることで、要求される書き込みスループットが低減されます。

使用可能なログ・スペースは、ログ・ファイル・サイズと 1 次ログ・ファイル数の積によって決まります（このサイズと数はデータベース・レベルで設定されます）。**logfilsiz** パラメーターは、各ログ・ファイル内の 4 K ページの数を示します。**logprimary** パラメーターは、1 次ログ・ファイルの数を指定します。コントロール・センターでは、データベース・ロギングの構成ウィザードも提供されています。

初期値として、10 という 1 次ログ・ファイル数を設定してみてください。この数は、通常の動作時であれば 1 次ログ・ファイルのラップアラウンド（先頭からの上書き）が少なくとも 1 分間は実行されない十分に大きい値です。

1 次ログ・ファイルのサイズを大きくすると、データベースのリカバリーが影響を受けます。**softmax** に定数値が指定されている場合は、ログ・ファイル・サイズが大きいほど、リカバリーにかかる時間が長くなる可能性があります。**softmax** パラメーターの値を小さくすることでこの問題を緩和できますが、ページ・クリーニングを積極的に行うほど効率性が低下する可能性もあることに留意してください。**softmax** パラメーターの値を大きくすると、書き込み結合の機会が増える一方で、リカバリー時間が長くなるというデメリットがあります。

softmax のデフォルト値である 100 を使用した場合は、データベース・マネージャーは、リカバリー時に単一のログ・ファイルのみの処理が必要になるようにページをクリーニングしようとします。最高のパフォーマンスを得るためには、次のコマンドを実行してこの値を 300 に引き上げることをお勧めします。この値を使用した場合は、リカバリー時に 3 つのログ・ファイルの処理が必要になる可能性があります。

db2 update db config for <yourDatabaseName> using softmax 300

4.11.5 大規模システムのスループットを向上するためのインライン長の設定

データベース表をチューニングする際は、高スループット率での実行時に可能な限りパフォーマンスを向上させるために、ラージ・オブジェクト（LOB）列のインライン長を設定することを検討してください。このパラメーターを使用すると、DB2 は、設定されたインライン長に収まる LOB をデータベース行と同じページに格納することで、データベース・アクセスを最適化できます。LOB をインライン化することで、照会操作と取得操作のパフォーマンスを向上させることができます。このパラメーターを設定すると、ログ・トラフィックが増大します。

LOB のインライン化は、複数の列に適用できます。特に重要なことは、LSW_BPD_INSTANCE_DATA 表の DATA 列のインライン長を設定することです。IBM 社内のパフォーマンス評価の結果から、この設定によって大規模な実動システムのスループットを大幅に向上可能であることがわかっています。

4.11.6 ラージ・オブジェクトが含まれた表スペースに対するシステム管理ストレージの使用

DB2 9.5 以降では、パフォーマンスにとって重要な LOB データが格納された REGULAR 表スペースまたは LARGE 表スペースを作成する際は、**MANAGED BY SYSTEM** を指定して、システム管理ストレージ（SMS）でのキャッシュされた LOB の処理を活用することをお勧めします。

この考慮事項は、次の BPM 関連製品に当てはまります。

- Business Process Choreographer データベース（BPEDB）
- BPMN データベース（twproc と twperfdb）
- サービス統合バスのメッセージング・エンジン・データ・ストアをサポートするデータベース

ラージ・オブジェクトが格納された表スペースに対する SMS の使用に関する背景情報については、78 ページの 4.10.5、「ダブル・バッファリングの回避」を参照してください。ここでは、詳細な説明が記載されています。

DB2 の表スペースは、多くの場合にシステム・パフォーマンスを向上させることができる **NO FILE SYSTEM CACHING** を使用して構成できます。**MANAGED BY SYSTEM** が指定された表スペースでは、キャッシングに関して LOB データを望ましい形で特例的に処理する SMS が使用されます。**NO FILE SYSTEM CACHING** が有効になっている場合でも（デフォルトでまたは指定に応じて）、LOB データへのアクセス時には、ファイル・システム・キャッシュが使用されます。

MANAGED BY DATABASE が指定された表スペースでは、キャッシングに関して LOB データと非 LOB データを区別しないデータベース管理ストレージ（DMS）が使用されます。具体的には、**NO FILE SYSTEM CACHING** は、LOB にアクセスする際に、直接ディスクに対して読み取りおよび書き込みが行われることを意味します。無条件にディスクから LOB を読み取ると、ディスク使用率が高まり、データベース・パフォーマンスが低下する可能性があります。

バージョン 9.1 以降の DB2 では、自動ストレージ（**AUTOMATIC STORAGE YES**）を使用するデータベースがデフォルトで作成されます。自動ストレージを使用するデータベースを作成すると、そのデータベースはディスク・スペースを管理して、ストレージ・パスと呼ばれる使用可能なファイル・システム・スペースの 1 つ以上のプールから、そのデータベース自体に対してディスク・スペースを割り当てます。自動ストレージが有効になっている場合は、**CREATE TABLESPACE** ではデフォルトで自動ストレージが使用されます（**MANAGED BY AUTOMATIC STORAGE**）。非一時的な表スペースである **REGULAR** と **LARGE** については、自動ストレージはファイル上の DMS を使用して実装されます。

バージョン 9.5 より前の DB2 では、表スペースのデフォルトのキャッシング方式は **FILE SYSTEM CACHING** でした。バージョン 9.5 では、直接入出力または同時入出力がサポートされているプラットフォームについては、デフォルトのキャッシング方式は **NO FILE SYSTEM CACHING** に変更されました。バージョン 9.5 のデフォルト設定を使用すると、**AUTOMATIC STORAGE YES** が指定されたデータベースと、**MANAGED BY AUTOMATIC STORAGE** が指定された表スペースが得られ、多くの場合は **NO FILE SYSTEM CACHING** が有効になっています。このような表スペースは、DMS を使用して実装されており、LOB をバッファー・プール内にもファイル・システム内にもキャッシュしません。

4.11.7 十分な使用可能ロック・リソースの確保

ロックは、**locklist** というデータベース・レベル・パラメーターによって制御される共通プールから割り当てられます。このパラメーターでは、この用途のために確保される 4000 ページの数を指定します。2 つ目のデータベース・レベル・パラメーターである **maxlocks** では、単一のアプリケーションが保持できるロック・プールの比率を制限します。アプリケーションが **maxlocks** で許可された比率を超えるロックを割り当てようとすると、または空きロック・プールが使い果たされると、DB2 はロック・エスカレーションを実行して、供給できるロックを補充します。ロック・エスカレーションによって、多数の行ロックが単一のテーブル・レベル・ロックに変換されます。

ロック・エスカレーションは、ロック・プールの過剰使用や枯渇という喫緊の問題に対処するものですが、データベースのデッドロックを引き起こす可能性があるため、通常の動作時は頻繁に実行されるべきではありません。場合によっては、アプリケーションの動作を変更して、多数の行をロックする大規模なトランザクションを小さめのトランザクションに分割することで、ロック・プールに対する負荷を軽減できます。より簡単な対処法として、まずデータベースをチューニングしてみてください。

バージョン 9 以降の DB2 では、**locklist** パラメーターと **maxlocks** パラメーターがデフォルトで自動的に調整されます。これらのパラメーターを手動でチューニングするには、db2diag.log を調べるか、システム・モニターを使用してデータベース・レベルのスナップショットを収集して、ロック・エスカレーションが実行されているかどうかを確認してください。初期症状がデータベースのデッドロックである場合は、次の手順に従って、これらのデッドロックがロック・エスカレーションによって生じたかどうかを検証してください。

1. 次のコマンドの出力で、ロック・エスカレーションのカウントを確認します。

   db2 get snapshot for database <*yourDatabaseName*>

2. 次のコマンドの出力を調べて、**locklist** と **maxlocks** の現行値を確認します。

   db2 get db config for <*yourDatabaseName*>

3. 必要に応じて、これらのパラメーターの値を変更します。例えば 例 4-3 に示すように、**locklist** は 100 に変更して、**maxlocks** は 20 に変更します。

例 4-3 locklist と maxlocks の値の設定

```
db2 update db config for <yourDatabaseName> using locklist 100 maxlocks 20
```

locklist のサイズ値を大きくするときは、必要になる追加メモリーの割り当てが与える影響を考慮してください。多くの場合、locklist は、バッファー・プール専用として割り当てられたメモリーよりも小さいですが、必要な合計メモリー量が仮想メモリー・ページングを引き起こすレベルであってはいけません。

maxlocks の比率値を大きくするときは、値を大きくすることで、少数のアプリケーションが空きロック・プールを枯渇させる可能性が生じるかどうかを考慮してください。このような枯渇は、ロック・エスカレーションの新たな原因となる可能性があります。あまり多くのロックを必要としない他のアプリケーションがロックを割り当てようとしたときには、空きロック・プールは使い果たされているからです。したがって、多くの場合は、まず locklist のサイズ値のみを大きくすることをお勧めします。

4.11.8 クラスター化アプリケーション用のカタログ・キャッシュのサイズ制限の設定

カタログ・キャッシュは、負荷の大きいアクティビティー（特に動的 SQL の実行計画の作成）の繰り返しを回避するために使用されます。したがって、このキャッシュを適切なサイズに設定することが重要です。

デフォルトでは、接続可能なアプリケーションごとに複数の 4 KB ページ分のメモリーが割り当てられます。接続可能なアプリケーションの数は、**MAXAPPLS** データベース・パラメーターによって指定されます。この 4 KB ページの数は、DB2 9 では 4、DB2 9.5 以降では 5 です。**MAXAPPLS** はデフォルトでは AUTOMATIC であるため、その値は実行時に接続されるアプリケーションのピーク数と一致するように調整されます。

クラスター化アプリケーション（BPM V7.5 の BPEL Process Choreographer 内に導入したアプリケーションなど）を実行している場合は、**MAXAPPLS** に 1000 より大きい値を指定できます。このような値を指定すると、デフォルト・チューニングの場合は、少なくとも 4000 ページがカタログ・キャッシュ用に割り当てられます。同じワークロードについては、500 ページで十分です。

db2 update db config for <yourDatabaseName> using catalogcache_sz 500

デフォルトの動作では、データベース接続がさまざまな形で使用されることが想定されています。クラスター化アプリケーションは通常、複数の接続にまたがって同じような形で使用されるため、小さめのパッケージ・キャッシュが有効になります。パッケージ・キャッシュのサイズを制限することで、メモリーが解放されて、その分を他のより有用な用途に使用できます。

CATALOGCACHE_SZ データベース・パラメーターを手動でチューニングするには、次の Web サイトに記載された参考情報を参照してください。

http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.admin.doc/doc/r0000338.htm

4.11.9 DB2 9.5 未満でのデータベース・ヒープの適切なサイズ設定

DB2 9.5 以降では、データベース・ヒープの **AUTOMATIC** チューニングがデフォルトでサポートされています。可能な場合は、この値を使用することをお勧めします。

DBHEAP データベース・パラメーターを手動でチューニングするには、次の Web サイトに記載された参考情報を参照してください。

http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.admin.doc/doc/r0000276.htm

4.11.10 DB2 9.7 未満でのログ・バッファーの適切なサイズ設定

バージョン 9.7 より前の DB2 では、**LOGBUFSZ** のデフォルト値はわずか 8 ページでした。次のコマンドを実行して、この値を 256（バージョン 9.7 のデフォルト値）に設定することをお勧めします。

db2 update db config for <yourDatabaseName> using logbufsz 256

4.11.11 DB2 9.7 以降での現行コミットの無効化の検討

DB2 9.7 では、照会のサブミット時に常にデータのコミット済み値を返す新しい照会セマンティクスがサポートされています。このサポートは、新規作成されたデータベースに対してデフォルトで有効になります。場合によっては、次のコマンドを実行してこの新しい動作を無効にして、元の DB2 照会セマンティクスに戻すとパフォーマンスが向上することがあります。

db2 update db config for <*yourDatabaseName*> using cur_commit disabled

4.11.12 Business Process Manager V7.5 での Business Process Execution Language ビジネス・プロセスに関する参考情報

次の Web サイトでは、DB2 データベースの初期設定の指定について説明しているとともに、BPEDB の SMS テーブル・スペースの作成例を紹介しています。この Web サイトでは、BPEDB の計画と細かいチューニングに関する有用なリンクも掲載しています。

http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r0mx/index.jsp?topic=/com.ibm.websphere.bpc.doc/doc/bpc/t5tuneint_spec_init_db_settings.html

次の Web サイトでは、Business Process Choreographer 用の DB2 for Linux, UNIX, and Windows データベースを作成する方法を説明しています。この Web サイトでは、BPEDB の作成について詳述しているとともに、実稼働環境用のスクリプトの作成に関するヒントを記載しています。

http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r0mx/index.jsp?topic=/com.ibm.websphere.bpc.doc/doc/bpc/t2codbdb.html

4.11.13 Business Monitor V7.5 に関する参考情報

ここでは、Business Monitor V7.5 のパフォーマンスをチューニングして最大限に高める方法を説明します。

レジストリー変数の設定による並行性の向上

DB2 では、カーソル固定（CS: Cursor Stability）または読み取り固定（RS: Read Stability）の分離スキャンのための行ロックを延期できます。表または索引のスキャン時に行ロックが実行されたときは、いずれかのレコードが照会の述部を満たすことが判明するまで、この延期を継続できます。

並行性を向上させる手動として、いずれかの行が照会の条件を満たすことが確認されるまで行ロックを延期できる場合があります。通常は並行性を向上させるには、*DB2_SKIPDELETED* および *DB2_SKIPINSERTED* というレジストリー変数を設定します。前者を設定することで、コミットされていない削除をスキャンが無条件にスキップすることが許可され、後者を設定することで、コミットされていない挿入をスキャンが無条件にスキップすることが許可されます。例 4-4 には、MONITOR というデータベースに対してこれら 2 つのレジストリー変数を有効にする方法を示しています。

例 4-4　DB2_SKIPDELETED と DB2_SKIPINSERTED のレジストリー値の有効化

```
db2 connect to MONITOR

db2set DB2_SKIPDELETED=ON

db2set DB2_SKIPINSERTED=ON
```

トランザクション・ログの飽和の回避

DB2 ヘルス・モニター（HMON）は、データベース・マネージャー、データベース、表スペース、および表に関する情報を定期的に収集します。HMON は、データベース・システムのモニター・エレメント、オペレーティング・システム、および DB2 システムから取得されたデータに基づいてヘルス・インディケーターを計算します。HMON が <*MONMEBUS_SCHEMA*>.SIBOWNER 表の状況を調べて、トランザクションがトランザク

ション・ログ全体を占有している場合は、SQLCODE 964 というエラーが発生して、トランザクションは停止します。

このエラーを防止するには、次のコマンドを使用して HMON プロセスを無効にします。

db2 update dbm cfg using HEALTH_MON OFF

適切なロック・タイムアウトの設定

LOCKTIMEOUT パラメーターでは、アプリケーションがロックを取得するのを待つ秒数を指定します。このパラメーターは、アプリケーションのグローバル・デッドロックを回避するのに役立ちます。このパラメーターを 0 に設定した場合は、アプリケーションはロックを待ちません。この場合は、要求時にロックを取得できない場合は、アプリケーションは直ちに -911 という戻りコードを受け取ります。

このパラメーターを -1 に設定した場合は、ロック・タイムアウトの検知は無効になります。

初期値として 30 秒を設定した後で、必要に応じてチューニングすることをお勧めします。次の例では、MONITOR というデータベースに対してこのパラメーターを設定する方法を示しています。

例 4-5　LOCKTIMEOUT の初期値

```
db2 -v update db cfg for MONITOR using LOCKTIMEOUT 30
```

可能な場合にイベント XML を 32 K に制限

十分に小さいイベントは、着信イベント表内の標準 VARCHAR 列に永続化されます。大きいイベントは、代わりにバイナリー・ラージ・オブジェクト（BLOB）列に永続化されます。DB2 では、最大の VARCHAR 列は 32768 バイトです。BLOB の代わりに VARCHAR 列を使用すると、パフォーマンスが大幅に向上します。

マテリアライズ照会表の使用

大量の履歴データがある場合は（1 千万個以上のモニター・コンテキスト・インスタンスなど）、ディメンション・レポート内のダッシュボード・ナビゲーションの応答時間が大幅に長くなる可能性があります。IBM Business Monitor V7.5 のディメンション・レポート用には、Cognos Business Intelligence V10.1 が使用されています。BPM V7.5 で用意されているツールを DB2 で使用すると、既知のディメンション・メンバー値の指標値を事前計算するキューブ要約表を生成できます。このような値としては、例えばブロンズ顧客、シルバー顧客、およびゴールド顧客についての、「顧客ロイヤルティー・レベル」に基づいた「平均ローン額」という指標の値が挙げられます。DB2 では、これらの表はマテリアライズ照会表と呼ばれ、Business Monitor が提供する予約済みサービスによってこれらの表が定期的に更新されます。

この機能の使用について詳しくは、次のインフォメーション・センター・トピックを参照してください。

http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r5mx/index.jsp?topic=/com.ibm.wbpm.mon.admin.doc/data/enable_cubesumtable_refresh.html

4.12 Oracle 固有のデータベース・チューニング

DB2 の場合と同様に、Oracle データベースの包括的なチューニング・ガイドを提供することは、本書の範囲外です。ただし、いくつかの基本原則を知っておけば、BPM 製品と組み合わせて使用する場合の Oracle 環境のパフォーマンスを向上させるのに役立つ可能性があります。以下のセクションでは、これらの原則を説明して、詳細情報を紹介します。また、次のリファレンス情報も役に立ちます。

- Oracle Database 11g Release 1 資料（パフォーマンス・チューニング・ガイドを含む）
  http://www.oracle.com/pls/db111/homepage
- 「*Oracle Architecture and Tuning on AIX v2.20*」ホワイト・ペーパー
  http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100883

4.12.1 データベース統計情報の更新

Oracle は、デフォルトで有効になっている自動統計情報収集機能を備えています。スキーマ内のすべてのテーブルで統計情報を手動で更新する方法の 1 つは、dbms_stats ユーティリティーを使用することです。詳しくは、Oracle 製品資料を参照してください。

4.12.2 正しいバッファー・キャッシュ・サイズの設定

Oracle は、バッファー・キャッシュの自動メモリー管理機能を備えています。自動メモリー管理機能の構成について詳しくは、およびバッファー・キャッシュ・サイズの手動設定に関する指針については、次のリファレンス情報を参照してください。

- Oracle 10g R2 の場合
  http://download.oracle.com/docs/cd/B19306_01/server.102/b14211/memory.htm#i29118
- Oracle 11g R1 の場合
  http://download.oracle.com/docs/cd/B28359_01/server.111/b28274/memory.htm#i29118

4.12.3 正しいテーブル索引付けの管理

SQL アクセス・アドバイザーは、Enterprise Manager で提供されており、スキーマの変更（索引の変更を含む）について提案します。SQL アクセス・アドバイザーにアクセスするには、データベースのホーム・ページに移動して、ページ下部の「Related Links」セクションで**「Advisor Central」**というリンクを開きます。

4.12.4 適切なログ・ファイル・サイズの設定

DB2 とは異なり、Oracle では、ログの切り替え時に大きな負荷を伴うチェックポイント操作が実行されます。このチェックポイントでは、バッファー・キャッシュ内のすべてのダーティー・ページがディスクに書き出されます。したがって、ログ切り替えが頻繁に実行されることを防止するために、ログ・ファイルを十分な大きさにすることが重要です。この目的を達成するには、大量のログ・トラフィックを生成するアプリケーションは、大きめのログ・ファイルを必要とします。

4.12.5 Business Process Manager V7.5 での Business Process Execution Language ビジネス・プロセスに関する参考情報

BPM V7.5 での BPEL ビジネス・プロセスのチューニングに関する参考情報が記載された Web 資料が提供されています。

次の Web サイトでは、Oracle データベースの初期設定を指定する方法を説明しています。

http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r0mx/index.jsp?topic=/com.ibm.websphere.bpc.doc/doc/bpc/t5tuneint_spec_init_db_oracle.html

次の Web サイトでは、Business Process Choreographer 用の Oracle データベースを作成する方法を説明しています。この Web サイトでは、BPEDB の作成について詳述しているとともに、実稼働環境用の有用な作成スクリプトに関するヒントを記載しています。

http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r0mx/index.jsp?topic=/com.ibm.websphere.bpc.doc/doc/bpc/t2codbdb.html

LOB に関するデフォルトの Oracle ポリシーは、対象オブジェクトのサイズがしきい値を超えていない場合は、データを行内に保管するというものです。場合によっては、ワークロードには、このしきい値をいつも超える LOB が付随することがあります。デフォルトでは、このような LOB へのアクセスはバッファー・キャッシュを経由せずに行われるため、推奨される直接ストレージ・パスや同時ストレージ・パスを使用している場合は、LOB の読み取り時にディスク入出力に伴う待ち時間が発生します。

4.13 高度な Java ヒープ・チューニング

BPM 製品セットは Java で記述されているため、JVM のパフォーマンスはこれらの製品のパフォーマンスを大きく左右します。JVM は、オーサリング・パフォーマンスとランタイム・パフォーマンスの両方を向上させるために使用される可能性のある複数のチューニング・パラメーターを外部化します。これらのパラメーターのうち最も重要なのは、ガーベッジ・コレクションと Java ヒープ・サイズの設定に関するものです。ここでは、これらのトピックについて詳しく説明します。

本書で扱っている製品は、ほとんどのプラットフォーム（AIX、Linux、Windows など）で IBM JVM を使用し、他の特定のシステム（Solaris や HP-UX など）では HotSpot JVM を使用します。ベンダー固有の JVM 実装の詳細と設定については、適宜説明しています。本書に登場するすべての BPM V7 製品は、Java 6 を使用します。Java 6 の特性は、BPM V6.1/V6.2.0 製品で使用されていた Java 5 と似ていますが、BPM V6.0.2.x 以前のバージョンで使用されていた Java 1.4.2 とは大きく異なります。簡潔にするために、ここでは Java 6 のチューニングのみを説明します。次の Web サイトでは、IBM Java 6 診断ガイドを提供しています。

http://publib.boulder.ibm.com/infocenter/javasdk/v6r0/index.jsp

このガイドでは、本書で説明しているよりもはるかに多くのチューニング・パラメーターを説明していますが、これらのうちほとんどのパラメーターは特定の状況用であり、汎用ではありません。IBM Java 6 ガーベッジ・コレクション・アルゴリズムについて詳しくは、メモリー管理のセクションを参照してください。

次のサイトでは、Sun HotSpot JVM に関する追加のリファレンス情報を提供しています。

- Solaris 用の HotSpot JVM オプションの要約

  http://java.sun.com/docs/hotspot/VMOptions.html
- Solaris HotSpot JVM に関してよくある質問

  http://java.sun.com/docs/hotspot/PerformanceFAQ.html#20
- Sun HotSpot JVM の追加のチューニング情報

  http://java.sun.com/docs/performance/

4.13.1 ガーベッジ・コレクションのモニター

ヒープを正しく設定するには、verbosegc のトレースを収集してヒープの使用状況を確認してください。詳細ガーベッジ・コレクション（verbosegc）のトレースでは、ガーベッジ・コレクションの動作内容と統計情報が、IBM JVM の標準エラー出力（stderr）と Sun HotSpot JVM の標準出力（stdout）に出力されます。verbosegc のトレースを有効にするに

は、-verbose:gc という Java ランタイム・オプションを使用します。verbosegc の出力は、HotSpot JVM の場合と IBM JVM の場合で異なります（例 4-6 と 例 4-7 を参照）。

例 4-6　IBM JVM の verbosegc トレース出力の例

```
<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>
```

例 4-7　Solaris HotSpot JVM の verbosegc トレース出力の例（新世代と旧世代）

```
[GC 325816K → 83372K(776768K), 0.2454258 secs]
[Full GC 267628K → 83769K <- live data (776768K), 1.8479984 secs]
```

次の追加オプションを指定することで、Sun HotSpot JVM の verbosegc 出力をさらに詳細化できます。

- **-XX:+PrintGCDetails**
- **-XX:+PrintGCTimeStamps**

テキスト・エディターを使用して verbosegc の出力を分析すると手間がかかることがあります。Java ヒープを効果的に分析するための視覚化ツールを Web 経由で入手できます。このようなツールとしては、IBM Pattern Modeling and Analysis Tool (PMAT) for Java Garbage Collector があります。このツールは、次の Web サイトにある IBM alphaWorks からダウンロードできます。

http://www.alphaworks.ibm.com/tech/pmat

PMAT は、IBM、Sun、HP などの主要 JVM ベンダーによって提供される JVM の verbosegc 出力形式をサポートしています。

4.13.2 ほとんどの構成に対応するヒープ・サイズの設定

ここでは、ほとんどの構成に適した Java ヒープ・サイズを設定するための指針を記載しています。同じシステム上で複数の JVM を同時に実行する必要がある構成を使用している場合は、91 ページの 4.13.3、「同一システムで複数の JVM を実行している場合のヒープ・サイズの設定」を参照してください。例えば、BPM サーバーと Integration Designer の両方を同じシステム上で実行している場合は、複数の JVM が必要になることがあります。ラージ・ビジネス・オブジェクトをサポートすることが目的の場合は、56 ページの 4.4.2、「ラージ・オブジェクトのチューニング」を参照してください。

ヒープ・サイズが小さすぎる場合は、メモリー不足エラーが発生します。ほとんどの実動アプリケーションにとっては、IBM JVM のデフォルト Java ヒープ・サイズは小さすぎるため、これらのサイズを大きくしてください。ヒープ・サイズが小さすぎる場合は、メモリー不足エラーが発生します。一般に、HotSpot JVM のデフォルトのヒープ・サイズと新世代領域（nursery）サイズも小さすぎるため、これらも大きくしてください。

最適なヒープ・サイズを設定するためのいくつかの手法があります。ここでは、IBM JVM を AIX 上で実行している場合にほとんどのアプリケーションで使用できる手法を説明します。この手法の基本要素は、他のシステムにも適用できます。初期ヒープ・サイズ（**-Xms** オプション）を標準的な値（例えば 64 ビット JVM では 768 MB、32 ビット JVM では 256 MB）に設定します。最大ヒープ・サイズ（**-Xmx** オプション）を標準的かつ大きい値（例えば 64 ビット JVM では 3072 MB、32 ビット JVM では 1024 MB）に設定します。最大ヒープ・サイズは、ヒープのページングが発生するような値に設定することは絶対に避けてください。ヒープは常に物理メモリー内に格納されている必要があります。JVM は、ヒープを拡大したり縮小したりすることで、GC 時間を適切な制限内に保とうとします。verbosegc の出力を使用して、GC アクティビティーがモニターされます。

世代別同時 GC（**-Xgcpolicy:gencon**）が使用されている場合は、新世代領域サイズを特定の値に設定することもできます。デフォルトでは、新世代領域サイズは、合計ヒープ・サイズの 4 分の 1 と 64 MB のうち小さい方です。パフォーマンスを向上させるには、新世代領域サイズをヒープ・サイズの 2 分の 1 以上に設定して、このサイズ値を 64 MB に制限しないでください。新世代領域サイズを設定するには、次の JVM オプションを使用します。

- **-Xmn<*size*>**
- **-Xmns<*initialSize*>**
- **-Xmnx<*maxSize*>**

同様の方法で、HotSpot ヒープのサイズを設定できます。最小と最大のヒープ・サイズを設定するだけでなく、新世代領域サイズをヒープ・サイズの 4 分の 1 から 2 分の 1 の間になるように増やしてください。新世代領域サイズをヒープ全体の 2 分の 1 より大きくすることは絶対に避けてください。

新世代領域サイズを設定するには、次のように **MaxNewSize** パラメーターと **NewSize** パラメーターを使用します。

- **-XX:MaxNewSize=128m**
- **-XX:NewSize=128m**

64 ビットの IBM JVM を使用している場合は、**-Xgc:preferredHeapBase** パラメーターを使用して、4 GB より下位のメモリー・アドレス（クラスやスレッドなどがこの領域に割り当てられます）の枯渇に起因するネイティブのメモリー不足問題を回避してください。**-Xgc:preferredHeapBase** オプションを使用すると、下位の 4 GB アドレス・スペースの外に Java ヒープを移動できます。この問題の解決策について詳しくは、次の IBM JVM インフォメーション・センターを参照してください。

http://publib.boulder.ibm.com/infocenter/javasdk/v6r0/topic/com.ibm.java.doc.diagnostics.60/diag/understanding/mm_compressed_references.html?resultof=%22%63%6f%6d%70%72%65%73%73%22%20

ヒープ・サイズを設定したら、verbosegc のトレースを使用して GC アクティビティーをモニターします。出力を分析した後で、分析結果に応じてヒープ設定を変更します。例えば、GC に費やされた時間の比率が高い場合に、ヒープが最大サイズにまで拡大している場合は、最大ヒープ・サイズを大きくすることでスループットが向上する可能性があります。目安として、合計時間の 10% 超が GC に費やされている場合は、一般に高い比率と見なされます。

Java ヒープの最大サイズを大きくしても、必ずしもこの種の問題が解決されるとは限りません。これはメモリーの過剰使用の問題である可能性があるからです。反対に、GC の中断時間が原因で応答時間が長すぎる場合は、ヒープ・サイズを小さくしてください。両方の問題が発生している場合は、アプリケーションのヒープ使用状況を分析する必要があります。

4.13.3 同一システムで複数の JVM を実行している場合のヒープ・サイズの設定

それぞれの実行中の Java プログラムには、1 つのヒープが割り当てられています。単一の物理システム上で複数の Java プログラムを実行する構成を使用している場合は、ヒープ・サイズを適切に設定することが特に重要です。ヒープ・サイズの設定は、アドレス指定可能なメモリーの合計量が制限されている 32 ビット・システムの場合も重要です。

このような構成の例としては、32 ビット JVM を使用して Integration Designer と BPM サーバーを同じ物理システム上に配置しているケースが挙げられます。これらの各アプリケーションは、それぞれが独自の Java ヒープを持つ別々の Java プログラムです。すべての仮想メモリー使用量（Java ヒープと他のすべての仮想メモリー割り当て量を含む）の合計が、アドレス指定可能な物理メモリーの量を超えている場合は、Java ヒープのページングが発生する可能性があります。このようなページングが発生すると、全体的なシステム・パフォーマンスが大幅に低下する可能性があります。システム全体のパフォーマンス低下の可能性を最小限に抑えるには、次の指針に従ってください。

まず、それぞれの実行中 JVM の verbosegc トレースを収集します。

- verbosegc のトレース出力に基づいて、初期ヒープ・サイズを小さめの値に設定します。例えば、verbosegc のトレース出力から、ヒープ・サイズが短時間で 256 MB まで拡大した後に、ゆっくりと 400 MB まで拡大して、このサイズで安定することがわかるとします。このサイズ変化に基づいて、初期ヒープ・サイズを 256 MB に設定します（**-Xms256m**）。
- 同様に verbosegc のトレース出力に基づいて、最大ヒープ・サイズを適切に設定します。最大ヒープ・サイズをあまり小さい値に設定しないように注意してください。この値が小さすぎると、メモリー不足エラーが発生します。最大ヒープ・サイズは、ピーク・スループットに対応できる十分な大きさである必要があります。先ほどと同じ例を基準にすると、適切な最大ヒープ・サイズとして 768 MB が考えられます（**-Xmx768m**）。正しい最大ヒープ・サイズを設定することで、Java ヒープは、必要に応じて現行サイズの 400 MB を超えて拡大するための領域が得られます。Java ヒープは必要な場合にしか拡大しないため（例えばピーク・アクティビティー期間中にスループット率が上昇する場合など）、最大ヒープ・サイズを現在必要なヒープ・サイズより大きい値に設定することは一般に望ましい方針です。
- ヒープ・サイズをあまり小さくしないように注意してください。ヒープ・サイズが小さすぎると、ガーベッジ・コレクションが頻繁に実行されて、スループットが低下する可能性があります。この場合も、verbosegc のトレースはガーベッジ・コレクションの頻度を確認するのに役立ちます。ヒープ・サイズは、ガーベッジ・コレクションの頻繁な実行を防止できる十分な大きさにする必要があります。その一方で、ヒープ・サイズの累計サイズは、ヒープのページングが発生するほど大きくなってはいけません。このバランス調整は、構成によって異なります。

4.13.4 メモリー不足エラーが発生する場合のヒープ・サイズの縮小または拡大

java.lang.OutOfMemory 例外は JVM によってさまざまな状況で使用されるため、この例外の原因を特定することが難しい場合があります。これらの考えられるエラー原因を見分けるための決定的なメカニズムは存在しませんが、最初のステップとして、verbosegc を使用してトレースを収集することをお勧めします。ヒープ内のメモリー不足が問題である場合は、トレースの出力でこの不足状況を簡単に確認できます。verbosegc の出力について詳しくは、88 ページの 4.13.1、「ガーベッジ・コレクションのモニター」を参照してください。一般にこの例外の前には、多くのガーベッジ・コレクションが実行されて、空きヒープ・スペースがほとんどなくなるという現象が発生します。このような空きヒープ・スペースの不足が問題である場合は、ヒープ・サイズを大きくしてください。

java.lang.OutofMemory 例外が発生したときに十分な空きメモリーが存在している場合は、次のステップとして、verbosegc の出力でファイナライザーのカウントを確認してください（この情報が得られるのは IBM JVM の場合のみです）。このカウント値が大きい場合は、ヒープ外のリソースがヒープ内のオブジェクトによって保持されて、ファイナライザーによってクリーニングされてるという検知しにくい状況が発生しています。ヒープ・サイズを

小さくすると、ファイナライザーの実行頻度が低下して、この状況を緩和できます。また、お使いのアプリケーションを調べて、ファイナライザーの実行を回避したり最小限に抑えたりできるかどうかを確認してください。

メモリー不足エラーは、Java ヒープの使用量とは無関係の問題（特定のシステム・リソースの枯渇など）によって発生する可能性もあります。このような問題の例としては、ファイル・ハンドル数が不十分なことやスレッド・スタック・サイズが小さすぎることなどが挙げられます。

場合によっては、構成をチューニングすることでネイティブ・ヒープの枯渇を回避できます。スレッドのスタック・サイズを小さくしてみてください（**-Xss** パラメーター）。スタック・サイズが不十分な場合は、深くネストされたメソッドによってスレッド・スタック・オーバーフローが発生する可能性があります。

また、64 ビットの IBM JVM を使用している場合は、**-Xgc:preferredHeapBase** パラメーターを使用して、4 GB より下位のメモリー・アドレス（クラスやスレッドなどがこの領域に割り当てられます）の枯渇に起因するネイティブのメモリー不足問題を回避してください。**-Xgc: preferredHeapBase** オプションを使用すると、下位の 4 GB アドレス・スペースの外に Java ヒープを移動できます。この問題の解決策について詳しくは、次の IBM JVM インフォメーション・センターを参照してください。

http://publib.boulder.ibm.com/infocenter/javasdk/v6r0/topic/com.ibm.java.doc.diagnostics.60/diag/understanding/mm_compressed_references.html?resultof=%22%63%6f%6d%70%72%65%73%73%22%20

ミドルウェア製品については、プロセス内バージョンの JDBC ドライバーを使用している場合は、ネイティブ・メモリー要件に大きな影響を与える可能性のあるプロセス外ドライバーを見つけることができます。例えば、タイプ 4 の JDBC ドライバー（DB2 Net ドライバーまたは Oracle の Thin ドライバー）を使用したり、IBM MQSeries をバインディング・モードからクライアント・モードに切り替えたりできます。詳しくは、DB2、Oracle、および MQSeries の資料を参照してください。

4.14 WebSphere InterChange Server のマイグレーションされたワークロードのチューニング

以下のチューニング・ヒントは、Integration Designer で WebSphere InterChange Server のマイグレーション・ウィザードを使用してマイグレーションされたワークロードのみに当てはまります。これらのヒントに加えて、本書で詳述している他の BPM サーバー関連のチューニング・ヒントも参照してください。

- 古い WBI アダプターやカスタム・アダプターと通信するために使用される JMS ベースのメッセージング用として、可能な場合は非永続キューを使用してください。
- 古い WBI アダプターやカスタム・アダプターと通信するために使用される JMS ベースのメッセージング用として、可能な場合は WebSphere MQ ベースのキューを使用してください。デフォルトでは、これらのアダプターは WebSphere MQ の API を使用して、MQ リンクを介してサービス統合バス宛先に接続します。MQ リンクは、MQ ベースのクライアントが送受信するメッセージを変換するプロトコル変換層です。WebSphere MQ ベースのキューに切り替えることで、MQ リンク変換に伴う負荷が発生しなくなり、パフォーマンスが向上します。
- 冗長ワークロードについてサーバー・ログを無効にしてください。一部のワークロードではトランザクションごとにログ・エントリーが生成されるため、絶え間なくディスク書き込みが行われて、全体的なスループットが低下します。サーバー・ログを無効にすると、このようなワークロードに起因するスループットの低下を緩和できる場合があります。

5

初期構成設定

本章では、いくつかの関連パラメーターの初期設定を提案します。これらの値がすべてのケースで最適であるというわけではありませんが、これらの値は内部パフォーマンス評価で高い効果を示しています。これらの値は少なくとも、多くの概念実証および顧客展開における開始点として有用です。48 ページの 4.1、「パフォーマンス・チューニング手法」で説明しているように、チューニングは反復プロセスです。その手順に従って、これらの値をご使用の環境に適するように調整してください。

5.1 Business Process Manager サーバー設定

ここでは、Business Process Manager（BPM）V7.5 サーバーの IBM 内部パフォーマンス評価に基づいた設定を示します。これらの設定は、47 ページの第 4 章、「パフォーマンスのチューニングと構成」で示しているチューニング手法およびガイドラインを使用して導き出されました。これらの設定を本製品使用の有用な開始点と考えてください。ここに示されていない設定については、製品インストーラーによって指定されるデフォルト設定を開始点として使用し、48 ページの 4.1、「パフォーマンス・チューニング手法」に示されているチューニング手法に従ってください。

ここでは以下の 3 つの設定について説明します。

- 別のサーバーに実動データベースを備えた Business Process Execution Language（BPEL）ビジネス・プロセスの 3 層セットアップ
- 別のサーバーに実動データベースを備えた Business Processing Modeling Notation（BPMN）ビジネス・プロセスの 3 層セットアップ
- サーバーに配列された実動データベースを備えた BPEL ビジネス・プロセスの 2 層（クライアント / サーバー）セットアップ

5.1.1 Web サービスおよびリモート DB2 システムを備えた 3 層構成

WebSphere Application Server を介して、内部パフォーマンス作業で 3 層構成を使用し、自動車保険請求処理をモデリングする BPEL ビジネス・プロセスのパフォーマンスを評価しました。この構成は、DB2 が BPM サーバーとは別のシステムにある多くの実稼働環境の 1 つの例です。通信には Web サービス・バインディングを使用しました。このビジネス・プロセスには以下の 2 つの操作モードがあります。

- BPEL microflow（直線的なプロセス）。人的介入を必要としない請求を処理します。
- BPEL microflow に macroflow（長期実行プロセス）を追加したパターン。macroflow は、レビューまたは承認が必要な場合（請求額が一定の限度を超えた場合など）に開始されます。

この構成では、3 つのシステムが使用されました。

- 要求ドライバー
- BPM V7.5 サーバー
- DB2 データベース・サーバー

BPM サーバーと DB2 データベース・サーバーには、スループットを最大化するために大規模なチューニングが必要でした。一部のチューニングは、オペレーティング・システム（AIX や Windows など）およびプロセッサー・コア数に合わせて変更されました。一般的なチューニングについて説明した後に、これらの変更を表形式で示します。

ここに示すすべてのトポロジーについて、以下の処理を実行して、BPM および DB2 をチューニングし、スループットを最大化することをお勧めします。

- 実動テンプレートを使用します。
- 共通データベースをローカル DB2 タイプ 4 として定義します。
- bpeconfig.jacl で BPEL ビジネス・プロセスのサポートを確立します。**「データ・ソース」「BPEDataSourceDb2」**をクリックして、WebSphere Application Server データ・ソースのプロパティーのステートメント・キャッシュを 300 に指定します。
- PMI を無効にします。
- **HTTP maxPersistentRequests** を -1 に設定します。

GC ポリシーを -Xgcpolicy:gencon に設定します（nursery（新世代領域）設定 **-Xmn** については表 5-1 および表 5-2 を参照）。

- SIB System、BPCDB、および SIB BPEDB には、リモート DB2 データベース（接続タイプ 4）を使用します。

表 5-1 は、BPM V7.5 サーバーを AIX にデプロイする場合にデフォルト値から変更する BPM V7.5 サーバー関連の設定を示しています。

表 5-1 AIX の場合の 3 層アプリケーション・クラスター設定

設定	**値**
Java ヒープ・サイズ（MB）	1536
Java nursery（新世代領域）サイズ（MB）**-Xmn**	768
デフォルトのスレッド・プール最大サイズ	100
BPEDB データ・ソース → 接続プール最大サイズ	300
BPEDB データ・ソース → WebSphere Application Server データ・ソースのプロパティー →ステートメント・キャッシュ・サイズ	300
BPC メッセージング・エンジン・データ・ソース → 接続プール最大サイズ	50
SCA SYSTEM メッセージング・エンジン・データ・ソース → 接続プール最大サイズ	50
BPM 共通データ・ソース → 接続プール最大サイズ	500
J2C アクティベーション・スペック → SOABenchBPELMod2_AS → カスタム・プロパティー → maxConcurrency、maxBatchSize	50、
リソース → 非同期 Bean → 作業マネージャー → BPENavigationWorkManager →作業要求キュー・サイズ、最大スレッド数、増加可能	400、 50、 不可
アプリケーション・クラスター → Business Flow Manager → メッセージ・プール・サイズ、トランザクション間キャッシュ・サイズ	5000、 400
WebContainer スレッド・プール最小サイズ、最大サイズ	100、100
com.ibm.websphere.webservices.http.maxConnection	50

表 5-2 は、BPM V7.5 サーバーを Intel システム上の Windows および Linux にデプロイする場合にデフォルト値から変更する BPM V7.5 サーバー関連の設定を示しています。

表 5-2 Intel システム上の Windows および Linux の場合の 3 層 Web サービスおよびリモート DB2 チューニングの変更内容

チューニングの変更内容	**microflow: コア数**			**macroflow: コア数**	
	1	**2**	**4**	**1**	**4**
Java ヒープ・サイズ（MB）	1280	1280	1280	1280	1280
Java nursery（新世代領域）サイズ（MB）-Xmn	640	640	640	768	768
Web コンテナーのスレッド・プール最大サイズ	100	150	150	100	300
デフォルトのスレッド・プール最大サイズ	100	200	200	100	200
BPE データベース接続プール最大サイズ	150	250	250	150	350
BPC メッセージング・エンジン・データベース接続プール最大サイズ	30	30	30	30	150
SYSTEM メッセージング・エンジン・データベース接続プール最大サイズ	30	40	40	30	100
共通データベース接続プール最大サイズ	80	80	80	80	100

チューニングの変更内容	microflow: コア数			macroflow: コア数	
	1	2	4	1	4
J2C アクティベーション・スペック → SOABenchBPELMod2_AS → カスタム・プロパティー → maxConcurrency	40	40	40	160	160
BPEInternalActivationSpec バッチ・サイズ				10	10
SOABenchBPELMod2_AS バッチ・サイズ				32	32
Java カスタム・プロパティー com.ibm.websphere.webservices.http.maxConnection	100	200	200	200	200
アプリケーション・サーバー → server1 → Business Flow Manager → allowPerformanceOptimizations				Yes	Yes
アプリケーション・サーバー → server1 → Business Flow Manager → interTransactionCache.size				400	400
アプリケーション・サーバー → server1 → Business Flow Manager → workManagerNavigation.messagePoolSize				4000	4000
リソース → 非同期 Bean → 作業マネージャー → BPENavigationWorkManager → スレッド最小数、スレッド最大数、要求キュー・サイズ				30, 30, 30	30, 30, 30

DB2 データベース・サーバーには、BPM V7.5 サーバーで使用するように定義された複数のデータベースが備わっています。RAID 配列にわたってデータベース・ログおよびテーブル・スペースを展開し、ディスク使用を分散してください。以下のように、SCA.SYSTEM.<セル名 >.BUS データベースおよび BPEDB をチューニングします。

- **db2 update db cfg for sysdb using logbufsz 512 logfilsiz 8000 logprimary 20 logsecond 20 auto_runstats off**
- **db2 alter bufferpool ibmdefaultbp size 30000**

以下の DB2 コマンドおよび生成スクリプトを使用して、BPE データベースを作成およびチューニングします。

- **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**（BPM V7.5 サーバー生成スクリプト）
- **db2 -tf createSchema.sql**（BPM V7.5 サーバー生成スクリプト）
- **db2 alter bufferpool ibmdefaultbp size 132000**
- **db2 alter bufferpool bpebp8k size 132000**

5.1.2 Business Processing Modeling Notation プロセスによる、ヒューマン・サービスを使用した 3 層構成

内部パフォーマンス作業で 3 層構成を使用し、自動車保険請求処理をモデリングする BPMN ビジネス・プロセスのパフォーマンスを評価しました。照会タスク、請求タスク、完了タスク、およびコミット・タスクが含まれるコール・センター・シナリオによる請求の処理にヒューマン・サービスが使用されました。

この構成は、DB2 が BPM サーバーとは別のシステムにある多くの実稼働環境の 1 つの例です。通信には Web サービス・オープン SCA バインディングが使用されました。この構成では、3 つのシステムが使用されました。

- 要求ドライバー
- BPM V7.5 サーバー
- DB2 データベース・サーバー

3 層構成における BPM V7.5 サーバー

3 層構成で BPM V7.5 を使用する場合、以下の設定をお勧めします。

- 64 ビット JVM の場合、Java コマンド行パラメーター **-Xgencon**、**-Xms1800M**、**-Xmx1800M**、**-Xmn800M** を追加することによって、Java メモリー管理設定を変更します。
- WebContainer ThreadPool のサイズを最小 200、最大 400 に増加します。
- デフォルトのスレッド・プールの最大サイズを 40 に増加します。
- 選択した BPM V7.5 コンポーネント（Web サービスなど）のロギングを無効にします。

3 層構成における DB2 データベース・サーバー

3 層構成で DB2 データベース・サーバーを使用する場合、以下の設定が必要です。

- ログ・ファイルとコンテナーを別個の（RAID）ディスクに分けます。
- twproc データベースのログ・ファイル・サイズを大きくします（16384 ページに設定）。
- 以下のコマンドを実行して twproc データベースのファイル・システム・キャッシングを有効にします。
  db2 alter tablespace userspace1 file system caching
- 以下の技術情報に従って、SIBOWNER テーブルを自動 runstats 実行の対象から除外します。

  http://www-01.ibm.com/support/docview.wss?uid=swg21452323
- データベース統計情報が最新であることを確認します。

  データベースが高いスループット率で実行されている場合は、ピーク・スループットへの影響を避けるために、データベースの自動保守タスクをいくつかまたはすべて無効にすることを検討してください。ただし、これらの機能を無効にした場合は、必ず **runstats** を定期的に実行してデータベース統計情報を更新してください。
- 以下の技術情報に従って、新しい索引を作成します。

  http://www-01.ibm.com/support/docview.wss?uid=swg21474536

  DB2 を使用している場合は、次のコマンドを使用して索引を作成します（他のデータベースにも同様のコマンドが用意されています）。
  - **db2 "CREATE INDEX IDX_SOAB_BPD_INSTANCE ON LSW_BPD_INSTANCE (SNAPSHOT_ID ASC, BPD_INSTANCE_ID ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS”**
  - **db2 "CREATE INDEX IDX_SOAB_BPD_INSTANCE_VAR ON LSW_BPD_INSTANCE_VARIABLES (BPD_INSTANCE_ID ASC, VARIABLE_TYPE ASC, ALIAS ASC) ALLOW REVERSE SCANS COLLECT SAMPLED DETAILED STATISTICS"**

5.1.3 Java Message Service ファイル・ストアを使用した 2 層構成

2 層構成を使用して、一般的な住宅ローン申請処理をモデリングする長期実行ビジネス・プロセスのパフォーマンスを評価しました。この構成は、使用可能なハードウェアが制限されている場合の概念実証で一般的ですが、同じ物理システム上にある BPM V7.5 サーバーと DB2 で使用されます。ただし、この構成は、代表的な実動構成ではありません。Java Message Service（JMS）バインディングが通信に使用されます。

この構成では、BPE が DB2 データベースを使用し、メッセージング・エンジンはファイル・ストアを使用するよう構成されます。ファイル・ストア・オプションを選択するには、プロファイル管理ツールを始動し、**「拡張プロファイル作成」**をクリックして、「データベースの構成」ウィンドウで**「メッセージング・エンジンでこのファイル・ストアを使用する」**をクリックします。

BPE データベースのチューニング・パラメーターの設定は、最初は DB2 Configuration Advisor を使用して導き出されました。そのうちのいくつかの主要パラメーター設定を以下のように変更します。

- **MAXAPPLS** は、可能なすべての JDBC 接続プール・スレッドからの接続に対応できるように拡張します。
- 各データベースのデフォルトのバッファー・プール・サイズ（IBMDEFAULTBP の 4 KB ページの数）は各プール・サイズが 256 MB になるように設定します。

表 5-3 は、この構成に推奨されるパラメーター設定を示しています。

表 5-3 JMS ファイル・ストアを使用した 2 層構成のパラメーター設定

パラメーター名	Business Process Choreographer データベース（BPEDB）の設定
APP_CTL_HEAP_SZ	144
APPGROUP_MEM_SZ	13001
CATALOGCACHE_SZ	521
CHNGPGS_THRESH	55
DBHEAP	600
LOCKLIST	500
LOCKTIMEOUT	30
LOGBUFSZ	245
LOGFILSIZ	1024
LOGPRIMARY	11
LOGSECOND	10
MAXAPPLS	90
MAXLOCKS	57
MINCOMMIT	1
NUM_IOCLEANERS	6
NUM_IOSERVERS	10
PCKCACHESZ	915
SOFTMAX	440
SORTHEAP	228
STMTHEAP	2048
DFT_DEGREE	1
DFT_PREFETCH_SZ	32
UTIL_HEAP_SZ	11663

パラメーター名	Business Process Choreographer データベース（BPEDB）の設定
IMBDEFAULTBP	65536

これらのデータベース・レベルのパラメーター設定に加えて、管理コンソールを使用してその他のいくつかのパラメーターも変更する必要があります。これらの設定は、主に同時実行（スレッドの設定）に影響を与えます。

- 未完了のトランザクションが多いと、必要なスレッド数も多くなるため、予期される同時実行の数は、スレッド・プールのサイズに影響を与えます。可能な解決方法として、デフォルトの 50 スレッドを超えるようにデフォルトのスレッド・プール・サイズの増加調整が必要な場合があります。
- アクティベーション・スペックとして、最大同時実行スレッド数を 50 に設定します。
- Business Process Choreographer データベース（BPEDB）の場合は、データベース接続プール・サイズを 60 に、ステートメント・キャッシュ・サイズを 300 に増加します。
- JMS 接続プールの最大接続数のプロパティーを 40 に設定します。
- DB2 JDBC Universal Driver タイプ 2 ドライバーを使用してローカル・データベースに接続します。タイプ 2 ドライバーでは、BPM V7.5 サーバーおよび DB2 が同じ物理システムに配置されている場合にパフォーマンスが高くなります。
- BPM V7.5 サーバー JVM のヒープ・サイズを 1024 MB の固定サイズに設定します。さらに、gencon ガーベッジ・コレクション・ポリシーを使用します。

5.2 WebSphere ESB の設定

ここでは、WebSphere ESB の選択した内部パフォーマンス評価で使用した設定について説明します。これらの設定は、47 ページの第 4 章、「パフォーマンスのチューニングと構成」で示しているチューニング手法およびガイドラインを使用して導き出されました。これらの設定を WebSphere ESB 使用の開始点と考えてください。ここに示されていない設定については、製品インストーラーによって指定されるデフォルト設定を開始点として使用してください。チューニング手法の説明については、48 ページの 4.1、「パフォーマンス・チューニング手法」を参照してください。

5.2.1 WebSphere ESB の一般的な設定

WebSphere ESB の設定は、バインディングの選択肢にかかわらず、WebSphere ESB ソリューションのチューニングの開始点として適しています。

- トレースを無効にします。
- セキュリティーを確保する必要がある場合は、Java2 セキュリティーではなくアプリケーション・セキュリティーを使用して、プロセッサー使用量を軽減します。
- Java ヒープ・サイズを Windows の場合も AIX の場合も 1280 MB に設定します。
- gencon ガーベッジ・コレクション・ポリシーを有効にし（**-Xgcpolicy:gencon**）、nursery（新世代領域）ヒープ・サイズを 1024 MB に設定します。

5.2.2 Web サービス用の WebSphere ESB の設定

Web サービス用の WebSphere ESB の設定は、以下のとおりです。

- PMI モニタリングを無効にします。
- WebContainer スレッド・プール・サイズの値を最大 50、最小 10 に設定します。
- WebContainer スレッド・プールの非アクティブ・タイムアウトの値を 3500 に設定します。

5.2.3 Java Message Service 用の WebSphere ESB の設定

WebSphere MQ および JMS 用の WebSphere ESB の設定は、以下のとおりです。

- アクティベーション・スペック

  同時実行エンドポイント最大数を 50 に設定します。

- キュー接続ファクトリー

  接続プール最大サイズを 51 に設定します。

- DiscardableDataBufferSize

  10 MB に設定し、CachedDataBufferSize を 40 MB に設定します。

5.2.4 Java Message Service 永続用の DB2 の設定

以下の値を設定します。これらの設定ではメッセージの保持にデータベースが使用されるため、これらの設定は JMS 永続構成にのみ関連します。

- データベースの表スペースとログを高速ディスク・サブシステムに配置します。
- ログを表スペースとは別のデバイスに配置します。
- バッファー・プール・サイズを適切に設定します。
- 最小および最大接続数を 30 に設定します。
- ステートメント・キャッシュ・サイズを 40 に設定します。
- DB2 ログ用に未加工パーティションをセットアップします。

ワークロードの説明で特に断りのない限り、製品インストーラーによって指定されるデフォルトの設定を使用してください。

関連資料

ここに示している資料は、特に、この文書で取り上げているトピックをより詳細に考察する場合に適しています。

IBM Redbooks

この資料のご注文については、102 ページの『Redbooks の入手方法』を参照してください。次の参照文献は、ソフトコピーのみで提供されている可能性があります。

IBM Business Process Manager V7.5 Production Topologies、SG24-7976:

http://www.redbooks.ibm.com/abstracts/sg247976.html

オンライン・リソース

以下に、詳細情報源としての関連 Web サイトを示します。

- WebSphere Application Server Performance

  http://www.ibm.com/software/webservers/appserv/was/performance.html

- WebSphere Application Server インフォメーション・センター（チューニング・ガイドを含む）

  http://www-306.ibm.com/software/webservers/appserv/was/library/?S_CMP=rnav

- DB2 バージョン 9 のベスト・プラクティス

  http://www.ibm.com/developerworks/data/bestpractices/?&S_TACT=105AGX11&S_CM

- DB2 バージョン 9.7 インフォメーション・センター

  http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp

- IBM SDK および Runtime Environment Java Technology Edition バージョン 6 の診断ガイド

  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 ドキュメント・ライブラリ

  http://docs.oracle.com/cd/E16338_01/index.htm

- IBM Integration Designer V7.5 インフォメーション・センター

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r5mx/topic/com.ibm.wbpm.main.do.homepage-bpm.html

- IBM Business Process Management V7.5 インフォメーション・センター

  http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r5mx/topic/com.ibm.wbpm.main.do.homepage-bpm.html

- Performance tuning resources for WebSphere Process Server and IBM BPM solutions

  http://www.ibm.com/developerworks/websphere/library/techarticles/1111_herrmann/1111_herrmann.html
- Understanding and Tuning the Event Manager

  http://www-01.ibm.com/support/docview.wss?uid=swg21439613
- Best practices for DB2 for Linux, UNIX, and Windows

  http://www.ibm.com/developerworks/data/bestpractices/db2luw/
- DB2 バージョン 9.5 インフォメーション・センター

  http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp
- Extending a J2CA adapter for use with WebSphere Process Server and WebSphere Enterprise Service Bus

  http://www-128.ibm.com/developerworks/library/ws-soa-j2caadapter/index.html?ca=drs-
- WebSphere Enterprise Service Bus のサポート・ホーム・ページのリンク

  http://www-306.ibm.com/software/integration/wsesb/support/
- Oracle Architecture and Tuning on AIX V2.2.0

  http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100883
- Oracle 10g リリース 2 ドキュメント・ライブラリ（パフォーマンス・チューニング・ガイドを含む）

  http://otndnld.oracle.co.jp/document/products/as10g/101202/doc_cd1/index.htm

Redbooks の入手方法

以下の Web サイトから、Redbooks、Redpaper、技術情報、ドラフト資料、およびその他の資料の検索、表示、またはダウンロード、およびハードコピーによる Redbooks 資料のご注文ができます。

ibm.com/jp/support/redbooks

IBM からのサポート

IBM サポートおよびダウンロード

ibm.com/support

IBM グローバル・サービス

ibm.com/services

IBM Business Process Manager V7.5 パフォーマンス・チューニングおよびベスト・プラクティス

チューニングに役立つヒントを学ぶ

最新のベスト・プラクティスを入手する

設定例を確認する

この IBM Redpaper 資料には、IBM Business Process Manager（BPM）V7.5（全エディション）および IBM Business Monitor V7.5 のパフォーマンス・チューニングのヒントとベスト・プラクティスが記載されています。これらの製品は、主要な一連のサービス指向アーキテクチャー（SOA）およびビジネス・プロセス・マネジメントのテクノロジーに基づいた開発およびランタイムの統合環境を表します。このようなテクノロジーとして、Service Component Architecture（SCA）、Service Data Object（SDO）、Web サービス用 Business Process Execution Language（BPEL）、および Business Processing Modeling Notation（BPMN）があります。

BPM および Business Monitor は双方とも IBM WebSphere Application Server インフラストラクチャーのコア機能が基盤となっています。このため、BPM ソリューションでは、WebSphere Application Server およびそれに対応するプラットフォーム Java 仮想マシン（JVM）に関するチューニング、構成、およびベスト・プラクティスの情報が役に立ちます。

本書は、IBM 内（開発、サービス、テクニカル・セールスなど）と顧客の両方にわたる、さまざまなグループを対象としています。BPM および Business Monitor を組み込むソリューションの実装を検討している顧客、またはそのような実装の初期段階にある顧客にとって、本書は必ず有益な参照資料になります。本書は、アプリケーションの開発およびデプロイメントの際のベスト・プラクティスとして、またセットアップ、チューニング、および構成に関する情報の参照資料として役に立ちます。

本書は、各製品のパフォーマンスに影響を及ぼす問題の多くを紹介しており、構成とパフォーマンスの設定において合理的な第 1 選択を行うためのガイドとして役立ちます。同様に、これらの製品を使用するソリューションを既に実装している顧客は、本書の情報を使用して、統合ソリューションの全体的なパフォーマンスを向上させる方法について見識を得ることができます。

SG88-4065-00
(英文原典 : REDP-4784-00)

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

実際の経験に基づいたテクニカル情報の作成

IBM Redbooks は、IBM International Technical Support Organization が開発しています。世界中の IBM の専門家、顧客、およびパートナーが、現実的なシナリオに基づいて時宜を得たテクニカル情報を作成しています。提供される特定の推奨事項は、IT ソリューションを環境により効果的に実装するための助けとなります。

詳しくは、以下を参照してください。
ibm.com/jp/support/redbooks