JBoss Enterprise Application Platform 5

Cache に関するよくある質問とその回答 (FAQ)

JBoss Enterprise Application Platform 5 での使用向け

エディッション 5.1.2

Last Updated: 2017-10-26

JBoss Enterprise Application Platform 5 Cache に関するよくある質問とその回答 (FAQ)

JBoss Enterprise Application Platform 5 での使用向け
エディッション 5.1.2

Ben Wang

Bela Ban

Manik Surtani

Scott Marlow

Galder Zamarreño

編集者

Laura Bailey

法律上の通知

概要

本書は、JBoss Cache と JBoss Enterprise Application Platform 5 およびそのパッチリリースを使用した場合についてよくある質問とその回答をまとめたものです。

目次

第1章 一般的な情報

問： JBoss Cache とは何ですか？

答： JBoss Cache はレプリケートされたトランザクションキャッシュです。複数の JBoss Cache インスタンスを配布できるため、レプリケートされています（同じ JVM 内での配布か、同じマシンまたはネットワーク上の異なるマシン上に存在する複数の JVM 間での配布）。データはグループ全体でレプリケートされます。ユーザーが JTA 準拠のトランザクションマネージャを設定でき、キャッシュ動作をトランザクション化できるため、キャッシュはトランザクションキャッシュになり、処理中の JTA トランザクションに参加します。キャッシュはレプリケーションなしでも実行でき、これをローカルモードと呼びます。

JBoss Cache には、Core と POJO の 2 つのバージョンがあります。コアライブラリ (**`org.jboss.cache.Cache`** インターフェース) は、データをツリー構造で整理し、キャッシュ内にあるデータのすべてのロック、非活性化、除外、およびレプリケーションの特性を処理する基礎となるライブラリです。POJO ライブラリ (**`org.jboss.cache.pojo.PojoCache`** インターフェース) は、コアライブラリ上に構築され、キャッシュ内のオブジェクトのイントロスペクションを可能にし、JBoss AOP を使用して透過的な一貫性を提供します。JBoss Cache の POJO エディション (多くの場合、POJO Cache と呼ばれます) には、JBoss Cache ドキュメンテーション Web サイトで入手できる別のドキュメンテーションセット (『『POJO Cache User Guide』 (POJO Cache ユーザーガイド)』や FAQ など) が付属します。

問： JBoss Cache の開発者は誰ですか？

答： JBoss Cache には、アクティブな開発者とコントリビュータのコミュニティが存在します。このプロジェクトは Bela Ban 氏により創設され、現在 Manik Surtani 氏によって運営されています。POJO Cache サブシステムのリーダーは Jason Greene 氏であり、他の現在と過去のコントリビュータには、Ben Wang 氏、Harald Gliebe 氏、Brian Stansberry 氏、Vladimir Blagojevic 氏、Mircea Markus 氏、Jimmy Wilson 氏、Galder Zamarreño 氏、および Elias Ross 氏がいます。

問： 使用している JBoss Cache のバージョンを確認する方法を教えてください。

答： **`java -jar jbosscache-core.jar`** はバージョンの詳細を出力します。

問： アプリケーションと設定を JBoss Cache 1.x から 2.x にどのように移行できますか?

答： 詳細については、この wiki ページを参照してください。

問： 2.x から 3.x への場合はどうですか?

答： JBoss Cache 3.x は、2.x と API の互換性があります。ただし、廃止されたメソッドは JBoss Cache の今後のリリースで削除される可能性があるため、これらのメソッドをできる限り使用しないようコードをリファクタリングする必要があります。

JBoss Cache 3.x では、完全に新しい設定フォーマットが使用されます。ログにこれに関する警告が記録されますが、古い 2.x 設定ファイルは引き続き動作します。もう一度述べますが、設定ファイルを新しいフォーマットにできる限り移行することを推奨します。JBoss Cache 3.x ディストリビューションでは、設定ファイルを移行するためのスクリプトが提供されます (**`config2to3.sh`** と **`config2to3.bat`** を参照)。

JBoss Cache 3.x の一部の新しい機能を使用するには、新しい設定フォーマットを使用する必要があることに注意してください。

第2章 JBOSS CACHE: CORE

問： 同じ VM で複数の JBoss Cache インスタンスを実行できますか？

答： 実行できます。独自の設定（例：異なるキャッシュポリシーなど）を持つローカルキャッシュインスタンスを複数実行したい時など、JBoss Cache のインスタンスを複数実行したい場合があります。この例の場合、xml 形式の設定ファイルが複数必要となります。

問： JBoss Cache を第 2 レベルキャッシュとして Hibernate 内部で実行することはできますか？

答： Hibernate 3.0 リリース以降のリリースでは、JBoss Cache を第 2 キャッシュとして使用するよう設定できます。詳細については、Hibernate のドキュメンテーションおよび この wiki ページ をご覧ください。

MVCC 付き JBoss Cache 3.x が Hibernate 第 2 レベルキャッシュとして非常に効果的に動作します。

問： POJO Cache を Hibernate のキャッシュとして使用できますか？

答： Hibernate は Java オブジェクトの細かなフィールドを管理するため、Hibernate 内部で POJO Cache を第 2 レベルキャッシュとして使用する必要はありません。したがって、POJO Cache を使用する利点はなく、不必要なパフォーマンスの低下が発生します。。

問： JBoss Cache の設定方法を教えてください。

答： JBoss Cache は、設定 XML ファイルを使用して設定するか、**`org.jboss.cache.CacheFactory`** インスタンスに渡された **`org.jboss.cache.config.Configuration`** オブジェクトを使用してプログラムによって設定できます。

問： スキーマまたは DTD を使用して JBoss Cache 設定ファイルを検証できますか?

答： JBoss Cache 3.x 以降の場合はできます。XSD スキーマは jbosscache-core.jar ファイルで提供され、http://www.jboss.org/jbosscache/jbosscache-config-3.0.xsd でオンラインでも入手できます。ファイルを検証するためにこのスキーマを使用するよう IDE、テキストエディタ、または XML オーサリングツールを設定できます。

問： キャッシュモードの違いを教えてください。

答： JBossCache には、**`LOCAL`**、**`REPL_SYNC`**、**`REPL_ASYNC`**、**`INVALIDATION_SYNC`**、**`INVALIDATION_ASYNC`** の 5 つのキャッシュモードがあります。JBoss Cache を単一のインスタンスとして実行する場合は、レプリケートしないようにキャッシュモードを **`LOCAL`** に設定します。異なる JBoss Cache インスタンス間で同期レプリケーションを行う場合は、**`REPL_SYNC`** に設定します。非同期レプリケーションの場合は **`AYSNC_REPL`** を設定します。キャッシュしたデータをレプリケートせずに、特定アドレス下の陳腐化したデータをメモリーより削除すべきであることをクラスタの他のキャッシュに通知したい場合は、**`INVALIDATION_SYNC`** または **`INVALIDTAION_ASYNC`** を設定します。同時および非同期の動作は、インバリデーションおよびレプリケーションに適応されます。

REPL_ASYNC および **INVALIDATION_ASYNC** は非ブロッキングであることに注意してください。これは、他の JBoss Cache をミラーまたはバックアップして使用し、ミラーのメッセージ受信確認の待ち時間を発生させたくない場合に便利なことがあります。

問： JBoss Cache のレプリケーションメカニズムについて教えてください。

答： JBoss Cache は、ネットワーク通信のために JGroups を使用します。JBoss Cache 設定には、JGroups 設定セクションが存在します。

ユーザーは、同じクラスタ名 (`cluster name`) を共有することにより JBoss Cache インスタンスのクラスタを設定できます。また、`ClusterConfig` 属性で新しいインスタンスの開始時にキャッシュデータを生成するかは オプションとして指定できます。

すべてのインスタンスが同じレプリケーショングループに参加すると、レプリケーションの変更がすべての参加メンバーに伝搬されます。バディレプリケーション機能を使用しない限り、メンバーの一部によってレプリケーションを行えるようなサブパーティションのメカニズムは存在しません。この詳細については、『『JBoss Cache User Guide』 (JBoss Cache ユーザーガイド)』を参照してください。

問： 2 ノードクラスタを使用しています。ネットワークに障害が発生した場合、キャッシュの実行は維持されますか？

答： 両キャッシュの実行が維持されますが、レプリケーションモードによってはトランザクションや操作がすべて完全に行われない場合があります。**REPL_SYNC** の場合は操作に失敗しますが、**REPL_ASYNC** の場合は操作に成功します。成功しても、キャッシュは非同期の状態になります。

問： リモート呼び出しやグループ通信に対応するため、JGroups の代わりに library X を接続することはできますか？

答： 現時点ではできません。パイプラインのコミュニケーションスイートと JBoss Cache の間に抽象層があるため、将来的には機能として導入される可能性があります。

問： キャッシュはクラスタ内にある他すべてのインスタンスに対してレプリケートする必要がありますか？クラスタが大きい場合、処理に時間がかかりませんか？

答： クラスタ内にあるすべてのノードに対してレプリケーションを行う必要がなくなりました。この機能はバディレプリケーションと呼ばれ、各ノードはクラスタ内で 1 つ以上の「バディ」を選択し、このバディに対してのみレプリケートすることができます。これにより、ノードが追加されてもメモリーやネットワークトラフィックは影響を受けないため、クラスターの拡張が容易になります。

バディレプリケーションの詳細や高拡張性の実現方法については、「ユーザーズガイド (Users' Guide)」をご覧ください。

問： バディレプリケーションを使用していますが、何らかの形式のセッションアフィニティは必要ですか?

答： セッションアフィニティは、使用される同じデータに対して同じキャッシュインスタンスに戻ることに関連します。これは厳密にはバディレプリケーションの要件ではありませんが、クラスタでのステータスの移動を最小限にすることが強く推奨されます。

問： 異なる設定プロパティが必要な場合（CacheModeや IsolationLevelなど）、適切な設定の org.jboss.cache.Cacheインスタンスを複数作成する必要がありますか？

答： はい、作成する必要があります。上記のプロパティはすべてキャッシュ毎のインスタンスです。したがって、個別の **org.jboss.cache.Cache** インスタンスが必要になります。

問： ネットワークの観点からこれはコストが高くなりませんか?つまり、各 org.jboss.cache.Cacheインスタンスに対して複数のソケットを作成する必要がありませんか?

答： はい、作成する必要があります。このような場合は、複数のキャッシュで単一の JGroups チャネルの共有を可能にする JGroups Multiplexer を使用してキャッシュを設定することが推奨されます。JGroups Multiplexer の設定方法については、『ユーザーズガイド (Users' Guide)』を参照してください。

より効率的な方法は、JGroups の共有トランスポートを使用することです。この方法の詳細については、JGroups ドキュメンテーションを参照してください。

問： ClusterName 設定要素は、JBoss AS クラスタの PartitionName と何らかの関係がありますか？

答： 両方とも JGroups のグループ名です。JGroups におけるチャネルの概念だけでなく、チャネルを異なるグループ名にパーティションすることができます。

問： JGroups ベースのコンポーネント (cluster-service.xml, cache [multiple instances]) を複数使用する場合に、マルチキャストアドレスが競合しないよう正しくコンポーネントを設定する方法を教えてください。

答： マルチキャストアドレス（およびポート）とグループ名の 2 つのパラメータを考慮しなければなりません。最低でも、コンポーネントを異なるグループ名で実行するようにしなければなりません。同じチャネルで実行するかは、通信パフォーマンスの重要度によって判断します。通信パフォーマンスが重要であれば、異なるチャネルで実行した方がよいでしょう。

問： 現在、JBoss Cache はキャッシュの永続ストレージをサポートしていますか？

答： サポートしています。JBoss Cache はキャッシュの永続化をサポートするキャッシュローダーインターフェイスを導入しています。キャッシュローダーに関する他の FAQ については以下を参照してください。

問： JBoss Cache はデータストアに対するキャッシュのパッシベーション／オーバーフローをサポートしていますか?

答： サポートしています。キャッシュパッシベーション／オーバーフローをサポートするために、JBoss Cache はキャッシュローダーを使用しています。この機能の設定方法や使用方法についてはドキュメンテーションを参照してください。

問： JBoss Cache はスレッドセーフですか？

答： スレッドセーフです。

問： JBoss Cache は現在 XA (2PC) トランザクションをサポートしていますか？

答： 現在サポートしていませんが、将来サポートされる予定です。内部実装は、異なるインスタンス間のトランザクションを調整するために 2PC に類似した手順を使用します。

問： JBoss Cache はどのトランザクションマネージャをサポートしますか?

答： JBoss Cache は、 JBoss Transactions など、JTA 準拠の TransactionManager をサポートします。

JBoss Cache にはダミーのトランザクションマネージャ (**`org.jboss.cache.transaction.DummyTransactionManager`**) が同梱されますが、このトランザクションマネージャを本番稼働用に使用することは推奨されません。このトランザクションマネージャはスレッドセーフではなく、内部テストを目的としてます。

問： キャッシュをトランザクション化するにはキャッシュをどのように設定したらよいですか？

答： JBoss AS に同梱されるデフォルトのトランザクションマネージャを使用するか、**`org.jboss.cache.transaction.TransactionManagerLookup`** インターフェイスを実装して **`javax.transaction.TransactionManager`** 実装のインスタンスを返します。設定プロパティである **`TransactionManagerLookupClass`** は、トランザクションマネージャへの参照を取得するためにキャッシュが使用するクラスを定義します。他のトランザクションマネージャをサポートするためにこのクラスを実装することは重要ではありません。属性が指定されると、キャッシュはトランザクションマネージャからトランザクションコンテキストをルックアップします。

JBoss Cache に同梱される **`org.jboss.cache.transaction.GenericTransactionManagerLookup`** クラスは最も人気があるトランザクションマネージャを検出し、そのマネージャにバインドできます。詳細については、**`GenericTransactionManagerLookup`** javadocs を参照してください。

問： キャッシュロッキングレベルはどのように制御しますか？

答： JBoss Cache では、トランザクション分離レベルを使用してキャッシュロッキングレベルを制御できます。この設定は、**`IsolationLevel`** 属性を使用して行います。トランザクション分離レベルは、データベース分離レベルである **`NONE`**、**`READ_UNCOMMITTED`**、**`READ_COMMITTED`**、**`REPEATABLE_READ`**、**`SERIALIZABLE`** に対応しています。楽観的ロッキングが使用された場合は、これらの隔離レベルは無視される点に注意してください。詳細については、『JBoss Cache User Guide』 (JBoss Cache ユーザーガイド) をご覧ください。

JBoss Cache 3.x 以降、MVCC ロッキングスキームを使用する場合は、**`READ_COMMITTED`** と **`REPEATABLE_READ`** だけがサポートされます。提供された他のすべての分離レベルは適切にアップグレードまたはダウングレードされます。

問： JBoss Cache は並行アクセスに対してどのようにデータをロックしますか？

答： JBoss Cache 2.x では、デフォルトで設定された分離レベルに基づいてペシミスティックロッキングを使用してデータノードをロックします。また、処理のオーバヘッドやパフォーマンスが若干悪化することと引き換えに並行処理の能力を向上できるオプティミスティックロッキングも提供します。JBoss Cache における並行処理やロッキングについての詳細はドキュメンテーションをご覧ください。

JBoss Cache 3.x では、楽観的および悲観的ロッキングが廃止され、楽観的および悲観的ロッキングよりも大幅に効率的な MVCC (Multi-Version Concurrency Control) が導入されました。MVCC 実装の詳細については、このブログのエントリーとこのwikiページを参照してください。

問： JBoss Cache でオプティミスティックロッキングまたは MVCC を有効にする方法を教えてください。

答： 詳細については、『『JBoss Cache User Guide』 (JBoss Cache ユーザーガイド)』の設定に関する項を参照してください。

問： トランザクションコンテキストなしでキャッシュロッキングレベルを使用できますか？

答： 使用できます。JBossCache は分離レベルのセマンティクスにより各ノードロッキング動作を制御します。そのため、トランザクションを使用しなくても、分離レベルを使用してロックレベルを指定することができます。トランザクション外部のノードロッキング動作は、**auto_commit** が有効なトランザクション下の場合と同様に考えることができます。

問： JBoss Cache は SELECT FOR UPDATE セマンティクスをサポートしますか？

答： はい、サポートします。ただし、これは、JTA トランザクション内で実行し、ノードロッキングスキームとして **MVCC** または **PESSIMISTIC** を使用している場合のみ可能です。

SELECT FOR UPDATE セマンティクスを実現するには、以下のことを行います。

```
// start transaction ...

cache.getInvocationContext().getOptionOverrides().setForceWriteLock(tr
ue);
Node n = cache.get("/a/b/c"); // this acquires a WRITE LOCK on this
node
    ...
    ...

// end transaction
```

問： レプリケーション (REPL_SYNC/REPL_ASYNC) または インバリデーション (INVALIDATION_SYNC/INVALIDATION_ASYNC) の場合に、キャッシュがネットワーク上でメッセージをブロードキャストする頻度を教えてください。

答： アップデートがトランザクション下である場合、トランザクションがコミットしようとする時（内部では準備段階）のみブロードキャストされます。つまり、バッチアップデートになりますが、操作がトランザクションコンテキスト下でない場合、各アップデートによってレプリケーションがトリガされます。ネットワークレイテンシが問題である場合は、パフォーマンスに影響します。

問： 一括削除を行う方法を教えてください。

答： **cache.removeNode("/myroot")** を実行する場合は、"/myroot" 以下のエントリがすべて再帰的に削除されます。

問： JBoss Cache を監視し管理することはできますか？

答： はい、できます。JBoss AS または JDK 5 の **jconsole** ユーティリティに同梱されるような JMX コンソールを使用します。詳細については、『JBoss Cache User Guide』 (JBoss Cache ユーザーガイド) の**管理情報 (Management Information)** の章を参照してください。

問： JBoss Cache のオブジェクト名には：文字が使用されます。このため、MBean サーバーで問題が発生します。これについて何をしたらいいですか？

答： これは一部の MBean サーバーの場合と同様です。JBoss Cache は JMX でバインドするすべてのオブジェクトに対して接頭辞として **jboss.cache:service=JBossCache** を使用します。これを回避するために、別の接頭辞で渡す **-Djbosscache.jmx.prefix** JVM パラメータを使用します。

問： JBoss Cache 1.3.0 の JBoss Cache 管理属性を無効にできますか？

答： はい、できます。『『JBoss Cache User Guide』 (JBoss Cache ユーザーガイド』の設定に関する項を参照してください。

問： jboss-serialization.jar はどうなりましたか?

答： JBoss Cache 2.0.0 以降、更新された Java 5 VM で JBoss Serialization のほとんどの利点が利用可能になったため、JBoss Serialization の依存関係が削除されました。JBoss Cache 2.0.0 のベースラインは Java 5 であるため、これらの利点を別に提供する必要はありません。

問： JBoss Cache はパーティショニングをサポートしていますか？

答： 現時点ではサポートしていません。異なるデータセットを異なるキャッシュインスタンスに存在させながら 1 つのレプリケーショングループとして参加させるためにユーザーが設定するようなパーティショニングを JBoss Cache はサポートしていません。

問： JBoss Cache は、Java EE コンテナなどの内部のアプリケーションクラスローティングの概念に対応できますか？

答： アプリケーション特有のクラスローディングは Java EE コンテナ内部で広く使用されています。例えば、ユーザーライブラリの特定のバージョンをスコープするために、ウェブアプリケーションに新しいクラスローダーが必要になったとします。デフォルトでは、JBoss Cache はクラスローダーにとらわれません。これにより 2 つの問題が発生します。

> オブジェクトインスタンスは cache1 に保存され、cache2 へレプリケートされます。この結果、cache2 のインスタンスはシステムクラスローダーによって作成されます。cache2 のシステムクラスローダーが必要なクラスにアクセスできない場合、レプリケーションに失敗します。レプリケーションに失敗しなくても、cache2 のユーザースレッドがアプリケーションクラスローダーによって定義されたタイプを要求する場合、このユーザースレッドはオブジェクトにアクセスできません。
>
> オブジェクトインスタンスはスレッド 1 によって作成され、スレッド 2 によってアクセスされます（2 つの異なるクラスローダーがある場合）。JBoss Cache は関係する異なるクラスローダーについて概念がありません。その結果、**ClassCastException** を持つことになります。これは、オブジェクトを 1 つのアプリケーションスペースから別のアプリケーションスペースへ渡す場合の典型的な問題です。JBossCache はオブジェクトを渡す際に間接的なレベルのみを追加します。

最初に挙げたような問題を解決するために、JBoss Cache は **CacheMarshaller** を使用します。これにより、アプリケーションのコードが、キャッシュツリーの一部（そこにレプリケートされたオブジェクトを処理する）でクラスローダーを登録できるようになりました。詳細については、『JBoss Cache User Guide』 (JBoss Cache ユーザーガイド) の**CacheMarshaller**の項をご覧ください。

次に挙げたような問題を解決するために、**Marshalledvalue** ラッパーでオブジェクトをラップする JBoss Cache の **UseLazyDeserialization** 設定オプションを使用できます。**MarshalledValue** は要求に応じてオブジェクトをシリアル化および非シリアル化し、常に適切なスレッドローカルコンテキストクラスローダーが使用されるようにします。

問： 現在、JBoss Cache は前処理イベントおよび後処理イベントの通知をサポートしていますか？

答： はい、サポートしています。ブール値が各通知コールバックに渡され、コールバックがイベント前であるか、イベント後であるかが識別されます。詳細については、**org.jboss.cache.notifications.annotations.CacheListener** アノテーションを参照してください。

問： キャッシュイベントをリッスンするようにカスタムリスナを実装する方法を教えてください。

答： これについては、『『JBoss Cache User Guide』 (JBoss Cache ユーザーガイド)』を参照してください。

問： 再デプロイされたキャッシュのデータにアクセスする際に ClassCastExceptions を回避するため、JBoss Cache で UseRegionBasedMarshalling 属性を使用できますか？

答： はい、使用できます。元々キャッシュマーシャリングは、ステート転送の際にキャッシュのオブジェクトを定義するクラスローダーへアクセスできないレプリケートされたキャッシュを回避するための方法でした。

デプロイする際、JBoss はトップレベルのデプロイメント成果物（例：EAR）のために新しいクラスローダーを作成します。また、アプリケーションサーバー内のクラスは、クラス名だけでなくクラスローダーによっても定義されます。例えば、デプロイメントの一部としてキャッシュをデプロイしない場合、アプリケーションをデプロイし、デプロイメントに属するクラスのインスタンスをキャッシュの内部に配置できますが、再デプロイを行って以前に配置したデータの get 操作を試行すると、ClassCastException が生じます。これは、クラス名が同じでもクラス定義が異なるためです。この場合、現在のクラスローダーはクラスが最初に配置されたクラスローダーとは異なります。

マーシャリングを有効にすると、キャッシュにあるデータのライフサイクルを制御することができます。また、アンデプロイする際にリージョンを非アクティブにし、デプロイの際に登録したクラスローダーの登録を抹消した場合、キャッシュにあるデータをローカルでエビクションします。よって、次にデプロイする際、データがキャッシュにないため、問題を回避することができます。データが存続する永続的なバッキングがある場合（例：CacheLoader の使用）や、JBoss Cache が永続化フレームワークの第 2 レベルキャッシュとして使用されている場合のみ、マーシャリングによる問題の回避が推奨されます。

この機能を実装するには、ユーザーズガイドの項「CacheMarshaller」に記載された例の指示に従ってください。**ServletContextListener** の代わりに、**start()** や **stop()** などのライフサイクルメソッドを格納する **MBean** にこのコードを追加することもできます。重要な点は、キャッシュが実行している間は動作できるよう、MBean をターゲットキャッシュに依存させることです。

第3章 エビクションポリシー

問： JBoss Cache はエビクションポリシーをサポートしていますか？

答： はい、サポートしています。JBoss Cache は現在 LRU、MRU、FIFO などの複数のエビクションポリシーをサポートします。また、ユーザーは独自のエビクションポリシーアルゴリズムをプラグインすることもできます。詳細については、『JBoss Cache User Guide』 (JBoss Cache ユーザーガイド) を参照してください。

問： JBoss Cache のエビクションポリシーはレプリケーションモードで動作しますか？

答： 場合によります。

エビクションポリシーはローカルモードでのみ動作します。そのため、ノードはローカルでのみエビクトされ、これにより一時的にキャッシュの内容が同期されないことがあります。しかし、エビクトされたノードのキャッシュの内容を取得しようとし、それが null だと分かった場合（例：**get** は null を返します）、他のデータソースから取得し、キャッシュのデータを再度投入するはずです。その間、ノードの内容は伝搬され、キャッシュの内容は同期されます。

しかし、キャッシュモードを **REPL_SYNC** または **REPL_ASYNC** に設定してエビクションポリシーを実行することもできます。ユースケースによっては、複数のキャッシュインスタンスを設定して独自のエビクションポリシーを持ったり（ローカルで適用されます）、エビクションポリシーを有効にして選択したインスタンスのみを持つこともできます。

また、キャッシュローダーオプションを利用すると、ローカルでエビクトされたノードをバックエンドストアへ永続化することもでき、後にそのノードをバックエンドストアより読み出すこともできます。

問： JBoss Cache は リージョン（Region） をサポートしていますか？

答： サポートしています。JBoss Cache にはリージョンの概念があり、リージョンでエビクションポリシーパラメータを設定することができます (例：**maxNodes**あるいは**timeToIdleSeconds**)。

JBoss Cache におけるリージョンとは、完全修飾名（**org.jboss.cache.Fqn**）などのツリー階層の一部を意味します。例えば、**/org/jboss** と **/org/foocom** を個別の 2 つのリージョンとして定義できます。しかし、現在リージョンはプログラムで設定することができる点に注意してください（例：すべて xml ファイルで設定されなければならない）。

問： エビクションポリシーを有効にしたのに「メモリー不足」(OOM) 例外が発生するのはなぜですか？

答： キャッシュのアクセス速度がタイマーに対応するエビクションポリシーの速度を越えると OOM が発生します。エビクションポリシーハンドラーは、**wakeUpInterval** ミリ秒 (3.x よりも前の場合は **wakeUpIntervalSeconds** 秒) 毎にウェイクアップし、エビクションイベントキューを処理します。そのため、キューのサイズが満杯になるとバックログが発生するため、エビクションタイマーが処理に追いつかないと OOM が発生します。この問題に対応し、VM ヒープサイズを増加させるには、**wakeUpInterval** の値を小さくし、タイマースレッドがキューをより頻繁に処理するようにします。

第4章 キャッシュローダー

問： キャッシュローダーとは何ですか？

答： キャッシュローダーは、JBoss Cache を (永続) データストアへ接続します。データがキャッシュにない場合、ストアからデータを読み出すためにキャッシュローダーが JBoss Cache により呼び出されます。また、キャッシュのデータに変更があった場合、変更をストアに保存するために Cache Loader が呼び出されます。

エビクションポリシーと共に、キャッシュローダーを持つ JBoss Cache を利用すると、大きなバックエンドデータストアのバインドされたキャッシュを維持することができます。頻繁に使用されるデータはデータストアからキャッシュへ読み出されます。頻繁にアクセスされるデータのアクセス速度を向上するため、使用頻度が最も低いデータはエビクトされます。設定はすべて XML で行われ、プログラマはローディングやエビクションを考慮する必要はありません。

現在、JBoss Cache には下記を含む複数のキャッシュローダー実装が同梱されています。

`org.jboss.cache.loader.FileCacheLoader`: この実装はファイルシステムを使用してデータを保存し、読み出します。JBoss Cache のノードは、ディレクトリやサブノード、サブディレクトリなどにマップされます。ノードの属性は、ディレクトリ内のデータファイルにマップされます。

`org.jboss.cache.loader.jdbm.JdbmCacheLoader`: この実装は JDBM (オープンソースファイルベースのトランザクション永続化エンジン) に基づきます。

`org.jboss.cache.loader.bdbje.BdbjeCacheLoader`: これは、速度が速く、効率的なトランザクションデータベースである Oracle の Berkeley DB Java Edition データベースに基づいた実装です。ストア全体に対して 1 つのファイルを使用します。Berkeley DB キャッシュローダーを JBoss Cache で使用し、製品として同梱したい場合は、Oracle から商用ライセンスを取得する必要があります。

`org.jboss.cache.loader.JDBCCacheLoader`: この実装は、リレーショナルデータベースを永続ストレージとして使用します。

詳細については、『『JBoss Cache User Guide』 (JBoss Cache ユーザーガイド)』のキャッシュローダーに関する章をご覧ください。

問： FileCacheLoader を本番稼働用に推奨しますか?

答： いいえ、推奨しません。FileCacheLoader には、本番稼働環境での使用を制限するいくつかの重大な考慮事項が存在します。このような環境で使用する場合は、慎重にこれらの考慮事項を十分に理解する必要があります。

FileCacheLoader がツリー構造をディスク (ディレクトリおよびファイル) で表す方法が原因で、深いツリーに対してトラバーサルが非効率です。

適切なファイルロックが実装されず、データが破損することがあるため、NFS や Windows 共有などの共有ファイルシステムでの使用は回避してください。

NONE の分離レベルを使用する場合は、複数のスレッドが同じファイルに書き込みを行おうとするため、書き込みによりデータの破損が発生することがあります。

ファイルシステムは本質的にトランザクション対応ではないため、トランザクションコンテキストでキャッシュを使用する場合に、ファイルへの書き込みエラー (コミットフェーズ中に発生) を回復することはできません。

一般的に、同時性、トランザクション性、またはストレス度が高い環境での FileCacheLoader の使用は推奨されません。FileCacheLoader の使用はテストに限定されます。

問： キャッシュローダーへの書き込みを非同期にできますか？

答： はい、できます。**async** 属性を true に設定します。詳細については、JBoss Cache ユーザーズガイドをご覧ください。デフォルトの設定では、キャッシュローダーの書き込みは同期され、さらにブロックされます。

問： 独自のキャッシュローダーを記述することはできますか？

答： はい、できます。キャッシュローダーは **`org.jboss.cache.loader.CacheLoader`** を実装、または **`org.jboss.cache.loader.AbstractCacheLoader`** を拡張するクラスです。設定は XML ファイルで行います (『『JBoss Cache User Guide』 (JBoss Cache ユーザーガイド)』を参照)。

問： キャッシュローダーは永続ストアを使用しなければなりませんか？

答： 使用する必要はありません。例えば、キャッシュローダーは webdav 対応のウェブサーバーからデータを読み出しすることができます（可能な場合は保存もできます）。また、ウェブからコンテンツを取得するキャッシングプロキシサーバーも利用できます。この場合、CacheLoader の実装は「ロード」機能のみ実装し、「保存」機能は実装しないことがあります。

問： 複数のキャッシュローダーを使用することはできますか？

答： はい、できます。CacheLoaderConfiguration XML 要素（『JBoss Cache User Guide』 (JBoss Cache ユーザーガイド) のキャッシュローダーに関する章を参照）を使用して、複数のキャッシュローダーを定義できます。この結果、有効な null 要素でないデータが見つかるまで、キャッシュは設定された順番ですべてのキャッシュローダーを確認します。書き込みは、すべてのキャッシュローダーに対して行われます（ignoreModifications 要素が true に設定されてるキャッシュローダーがある場合を除く）。

問： JBoss Cache 1.x.x でフォーマットされたデータを含む JDBCacheLoader または FileCacheLoader ベースキャッシュストアを JBoss Cache 2.0 フォーマットに移行できますか?

答： はい、できます。『『JBoss Cache User Guide』 (JBoss Cache ユーザーガイド)』に含まれる節「キャッシュローダー (Cache Loaders)」内の節「キャッシュローダーの変換 (Transforming Cache Loaders)」を参照してください。

問： TCPCacheServer を再起動しても TCPDelegatingCacheLoader は保持されますか?

答： JBoss Cache 2.1.0 以降の場合は、保持されます。TCP 接続を再確立するための再試行回数と待機時間を設定および調整する方法の詳細については、『『JBoss Cache User Guide』 (JBoss Cache ユーザーガイド)』を参照してください。

JBoss Cache 2.1.0 よりも前のバージョンの場合は、TCPCacheServer を再起動すると、キャッシュを使用するアプリケーションも再起動されることになります。

第5章 トラブルシューティング

問： JBoss Cache がうまく動作しないのですが、トラブルシューティングの情報はどこで入手できますか？

答： トラブルシューティングに関する項については、次の wiki リンクを参照してください。

付録A 更新履歴

改訂 5.1.2-100.402 **Fri Oct 25 2013** **Rüdiger Landmann**
Rebuild with Publican 4.0.0

改訂 5.1.2-100.2 **2012-07-18** **Anthony Towns**
Rebuild for Publican 3.0

改訂 5.1.2-100 **Thu Dec 8 2011** **Jared Morgan**
JBoss Enterprise Application Platform 5.1.2 GAに対する変更を追加。本ガイド文書の変更に関する情報は、『リリースノート 5.1.2』を参照してください。

改訂 5.1.1-100 **Mon Jul 18 2011** **Jared Morgan**
JBoss Enterprise Application Platform 5.1.1 GAに対する変更を追加。本ガイド文書の変更に関する情報は、『リリースノート 5.1.1』を参照してください。

改訂 5.1.0-100 **Wed Sep 15 2010** **Laura Bailey**
新しいバージョン要件に応じてバージョン番号が変更されました。
JBoss Enterprise Application Platform 5.1.0.GA 用に改訂。