Red Hat build of OpenJDK 11

Red Hat build of OpenJDK 17 での Shenandoah ガベージコレクターの使用

Last Updated: 2024-05-10

Red Hat build of OpenJDK 11 Red Hat build of OpenJDK 17 での Shenandoah ガベージコレクターの使用

法律上の通知

概要

Red Hat build of OpenJDK は、Red Hat Enterprise Linux プラットフォーム上の Red Hat 製品です。Red Hat build of OpenJDK 17 での Shenandoah ガベッジコレクターの使用 ガイドでは、Shenandoah ガベッジコレクターの概要と、Red Hat build of OpenJDK 17 での設定方法を説明します。

目次

RED HAT BUILD OF OPENJDK ドキュメントへのフィードバック

エラーを報告したり、ドキュメントを改善したりするには、Red Hat Jira アカウントにログインし、課題を送信してください。Red Hat Jira アカウントをお持ちでない場合は、アカウントを作成するように求められます。

手順

1. 次のリンクをクリックして **チケットを作成します**。
2. **Summary** に課題の簡単な説明を入力します。
3. **Description** に課題や機能拡張の詳細な説明を入力します。問題があるドキュメントのセクションへの URL を含めてください。
4. **Submit** をクリックすると、課題が作成され、適切なドキュメントチームに転送されます。

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、Red Hat CTO である Chris Wright のメッセージ を参照してください。

第1章 SHENANDOAH ガベージコレクター

Shenandoah は、実行中の Java プログラムと同時にガベージコレクションを実行することで、GC の一時停止時間が短縮する低一時停止時間ガベージコレクター (GC) です。Red Hat build of OpenJDK 17 のデフォルトのガベッジコレクター (CMS) および G1 の同時マークは、ライブオブジェクトの同時マークを実行します。

Shenandoah は同時圧縮を追加します。Shenandoah は、実行中の Java スレッドと同時にオブジェクトを圧縮することで、GC 一時停止時間が短縮されます。Shenandoah による一時停止時間はヒープサイズから独立しています。つまり、ヒープが 200 MB または 200 GB の場合でも、一貫性のある一時停止時間が得られます。Shenandoah は、応答性と予測可能な短い一時停止を必要とするアプリケーションのアルゴリズムです。

関連情報

- Shenandoah ガベージコレクターの詳細は、Oracle OpenJDK ドキュメントの Shenandoah GC を参照してください。

第2章 SHENANDOAH ガベージコレクターを使用した JAVA アプリケーションの実行

Shenandoah ガベージコレクター (GC) を使用して Java アプリケーションを実行できます。

前提条件

- Red Hat build of OpenJDK をインストールしている。**RHEL での Red Hat build of OpenJDK 17 のインストールと使用** の Red Hat Enterprise Linux での Red Hat build of OpenJDK 17 のインストール を参照。

手順

- **-XX:+UseShenandoahGC** JVM オプションを使用して、Shenandoah GC で Java アプリケーションを実行します。

```
$ java <PATH_TO_YOUR_APPLICATION> -XX:+UseShenandoahGC
```

第3章 SHENANDOAH ガベージコレクターモード

Shenandoah は 3 つの異なるモードで実行できます。**-XX:ShenandoahGCMode=<name>** を指定して、特定のモードを選択します。以下のリストでは、各 Shenandoah モードを説明します。

normal/satb (製品、デフォルト)

このモードは、Snapshot-At-The-playning (SATB) マーキングで同時ガベージコレクター (GC) を実行します。このマーキングモードは、Red Hat build of OpenJDK 17 のデフォルトのガベッジコレクターである G1 と同様に機能します。

iu (実験的)

このモードは、Incremental Update (IU) マーキングで同時 GC を実行します。これにより、より強固なメモリーを回収できます。このマーキングモードは SATB モードをミラーリングします。これにより、特に弱い参照へのアクセスに関して、保持性が低くなります。

passive (診断)

このモードでは、Stop the World Event GCs を実行します。このモードは機能テストに使用されますが、GC バリアーでパフォーマンスの異常を分けたり、アプリケーションで実際のライブデータサイズを把握したりするのに便利です。

第4章 SHENANDOAH ガベッジコレクターの基本設定オプション

Shenandoah ガベージコレクター (GC) には、以下の基本的な設定オプションがあります。

-Xlog:gc

個別の GC タイミングを出力します。

-Xlog:gc+ergo

ヒューリスティックな決定を出力します。これにより、外れ値が明らかになることがあります。

-Xlog:gc+stats

実行の最後に Shenandoah 内部タイミングでサマリーテーブルを出力します。
これは、ロギングが有効な状態で実行することが最適です。このサマリー表は、GC パフォーマンスに関する重要な情報を通知します。ヒューリスティックログは、GC の外れ値を判断するのに便利です。

-XX:+AlwaysPreTouch

ヒープページをメモリーにコミットし、レイテンシーの問題を減らすのに役立ちます。

-Xms および -Xmx

-Xms = -Xmx でヒープをサイズ変更不可にすると、ヒープ管理が容易になります。**AlwaysPreTouch** では、**-Xms = -Xmx** は起動時にすべてのメモリーをコミットします。これにより、メモリーが最後に使用されたときに問題が発生するのを回避します。**-Xms** は、メモリーアンコミットの低境界も定義するため、**-Xms = -Xmx** はすべてコミットされたままになります。フットプリントを低く設定するために Shenandoah を設定する場合は、**-Xms** を低く設定することが推奨されます。コミット/コミット解除のオーバーヘッドとメモリーフットプリントのバランスを取るために、どの程度低く設定するかを決める必要があります。多くの場合、**-Xms** は自由裁量で低く設定できます。

-XX:+UseLargePages

hugetlbfs Linux サポートを有効にします。

-XX:+UseTransparentHugePages

Huge Page を透過的に有効にします。透過的な Huge Page では、**/sys/kernel/mm/transparent_hugepage/enabled** と **/sys/kernel/mm/transparent_hugepage/defrag** を **madvise** に設定することを推奨します。**AlwaysPreTouch** で実行すると、起動時に **defrag** ツールツールの負荷を負うことになります。

-XX:+UseNUMA

Shenandoah はまだ NUMA を明示的にサポートしていませんが、マルチソケットホストで NUMA インターリービングを有効にすることが推奨されます。**AlwaysPreTouch** と組み合わせることで、デフォルトの設定よりも優れたパフォーマンスが得られます。

-XX:-UseBiasedLocking

競合のない (バイアス) ロックスループットにはトレードオフがあり、JVM がそれらを有効または無効にする安全なポイントがあります。レイテンシー指向のワークロードの場合は、バイアスロックをオフにします。

-XX:+DisableExplicitGC

ユーザーコードから System.gc() を呼び出すと、Shenandoah に追加の GC サイクルの実行が強制されます。**-XX:+ExplicitGCInvokesConcurrent** がデフォルトで有効になるため、通常は問題はありません。つまり、STW Full GC ではなく、同時 GC サイクルが呼び出されることを意味します。

改訂日時: 2024-05-10