Red Hat JBoss Enterprise Application Platform 8.0

Red Hat JBoss Enterprise Application Platform 的性能调整

评估 Red Hat JBoss Enterprise Application Platform 性能以及配置更新以提高性能的说明。

Last Updated: 2024-02-10

Red Hat JBoss Enterprise Application Platform 8.0 Red Hat JBoss Enterprise Application Platform 的性能调整

评估 Red Hat JBoss Enterprise Application Platform 性能以及配置更新以提高性能的说明。

法律通告

摘要

本书是 Red Hat JBoss Enterprise Application Platform 性能调优指南。

目录

提供有关 JBOSS EAP 文档的反馈

要报告错误或改进文档，请登录到 Red Hat JIRA 帐户并提交问题。如果您没有 Red Hat Jira 帐户，则会提示您创建一个帐户。

流程

1. 单击以下链接 **以创建 ticket**。
2. 在 **Summary** 中输入问题的简短描述。
3. 在 **Description** 中提供问题或功能增强的详细描述。包括一个指向文档中问题的 URL。
4. 点 **Submit** 创建问题，并将问题路由到适当的文档团队。

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于这一努力的精力，这些更改将在即将发布的版本中逐渐实施。有关让我们的语言更加包含的更多详情，请参阅我们的CTO Chris Wright 信息。

第 1 章 性能调优简介

JBoss EAP 安装默认优化。但是，您的环境、应用和使用 JBoss EAP 子系统的配置可能会影响性能，这意味着可能需要其他配置。

本指南为常见的 JBoss EAP 用例提供了优化建议，以及监控性能和诊断性能问题的说明。

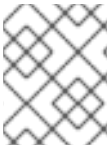
重要

在部署到生产之前，您应该测试并验证在暂存或测试环境中预计条件下的所有性能配置更改。

1.1. 关于本文档中 EAP_HOME 的使用

在本文档中，变量 ***EAP_HOME*** 用于表示 JBoss EAP 安装的路径。将此变量替换为 JBoss EAP 安装的实际路径。

- 如果您安装了 JBoss EAP 压缩文件，则安装目录是您提取压缩的存档的 **jboss-eap-8.0** 目录。
- 如果您使用安装程序安装 JBoss EAP， ***EAP_HOME*** 的默认路径为 **${user.home}/EAP-8.0.0** ：
  - 对于 Red Hat Enterprise Linux 和 Solaris： **/home/*USER_NAME*/EAP-8.0.0/**
  - 对于 Microsoft Windows： **C:\Users*USER_NAME*\EAP-8.0.0**
- 如果您使用 JBoss Tools 安装程序安装和配置 JBoss EAP 服务器， ***EAP_HOME*** 的默认路径为 **${user.home}/devstudio/runtimes/jboss-eap**:
  - 对于 Red Hat Enterprise Linux：**/home/*USER_NAME*/devstudio/runtimes/jboss-eap/**
  - 对于 Microsoft Windows：**C:\Users*USER_NAME*\devstudio\runtimes\jboss-eap** 或 **C:\Documents and Settings*USER_NAME*\devstudio\runtimes\jboss-eap**



注意

如果您在 JBoss 工具中将 **Target 运行时** 设置为 **8.0** 或更新的运行时版本，则您的项目与 Jakarta EE 8 规格兼容。

注意

EAP_HOME 不是环境变量。***JBOSS_HOME*** 是脚本中使用的环境变量。

第 2 章 监控性能

您可以使用任何可以检查计算机上运行的 JVM 的工具监控 JBoss EAP 性能。红帽建议您使用 JConsole，JBoss EAP 包含预配置的 wrapper 脚本或 Java Mission Control (JMC)。这些工具都提供 JVM 进程的基本监控，包括内存用量、线程使用率、加载类和其他 JVM 指标。

如果您在运行 JBoss EAP 的相同计算机上本地运行这些工具，则不需要配置。但是，如果您要运行这些工具之一来监控远程计算机上运行的 JBoss EAP，则需要一些配置才能接受远程 JMX 连接。

2.1. 为远程监控连接配置 JBOSS EAP

2.1.1. 为独立服务器配置远程监控连接

流程

1. 确保您已创建了管理用户。您可能希望创建单独的管理用户来监控您的 JBoss EAP 服务器。
2. 启动 JBoss EAP 时，将管理接口绑定到用于远程监控服务器的 IP 地址：

```
$ EAP_HOME/bin/standalone.sh -bmanagement=IP_ADDRESS
```

警告

这会将所有 JBoss EAP 管理接口（包括管理控制台和管理 CLI）公开给指定的网络。确保您只将管理接口绑定到专用网络。

3. 将以下 URI 与您的 JVM 监控工具中的管理用户名和密码一起使用，以连接到 JBoss EAP 服务器。以下 URI 使用默认管理端口(**9990**)。

```
service:jmx:remote+http://IP_ADDRESS:9990
```

2.1.2. 为受管域主机配置 JBoss EAP 远程监控连接

使用上述步骤来绑定受管域主机上的管理接口，将仅公开主机控制器 JVM 进行远程监控，而不是在该主机上运行的独立 JBoss EAP 服务器。

要将 JBoss EAP 配置为远程监控受管域主机上的各个服务器，请按照以下步骤操作。

流程

1. 在 **ApplicationRealm** 中创建一个新用户，您将用于连接到 JBoss EAP 服务器以进行远程监控。
2. 要将 **remoting** 子系统配置为使用 Elytron，请执行以下命令：

```
/profile=full/subsystem=jmx/remoting-connector=jmx:add(use-management-endpoint=false)
/socket-binding-group=full-sockets/socket-binding=remoting:add(port=4447)
/profile=full/subsystem=remoting/connector=remoting-connector:add(socket-
```

```
binding=remoting,sasl-authentication-factory=application-sasl-authentication)
```

3. 启动 JBoss EAP 受管域主机时，请将以下一个或多个接口绑定到您要用于监控的 IP 地址。
   - 如果要连接到受管域主机上运行的独立 JBoss EAP 服务器 JVM，请绑定公共接口：

     ```
     $ EAP_HOME/bin/domain.sh -b=IP_ADDRESS
     ```

   - 如果要连接到 JBoss EAP 主机控制器 JVM，也绑定管理界面：

     ```
     $ EAP_HOME/bin/domain.sh -bmanagement=IP_ADDRESS
     ```

     警告

     这会将所有 JBoss EAP 管理接口（包括管理控制台和管理 CLI）公开给指定的网络。确保您只将管理接口绑定到专用网络。

4. 在 JVM 监控工具中使用以下详情：
   - 要连接到受管域主机上运行的独立 JBoss EAP 服务器 JVM，请使用以下 URI 和之前创建的 **ApplicationRealm** 用户名和密码。

     ```
     service:jmx:remote+http://IP_ADDRESS:4447
     ```

     若要连接到单一主机上的不同 JBoss EAP 服务器，请将相应的服务器的端口偏移值添加到上述端口号。

   - 若要连接 JBoss EAP 主机控制器 JVM，可使用以下 URI 和管理用户名和密码：

     ```
     service:jmx:remote+http://IP_ADDRESS:9990
     ```

2.2. JCONSOLE

预配置的 JConsole 打包程序脚本与 JBoss EAP 绑定。使用此打包程序脚本可确保将所有必需的库添加到类路径中，同时还可从 JConsole 内访问 JBoss EAP 管理 CLI。

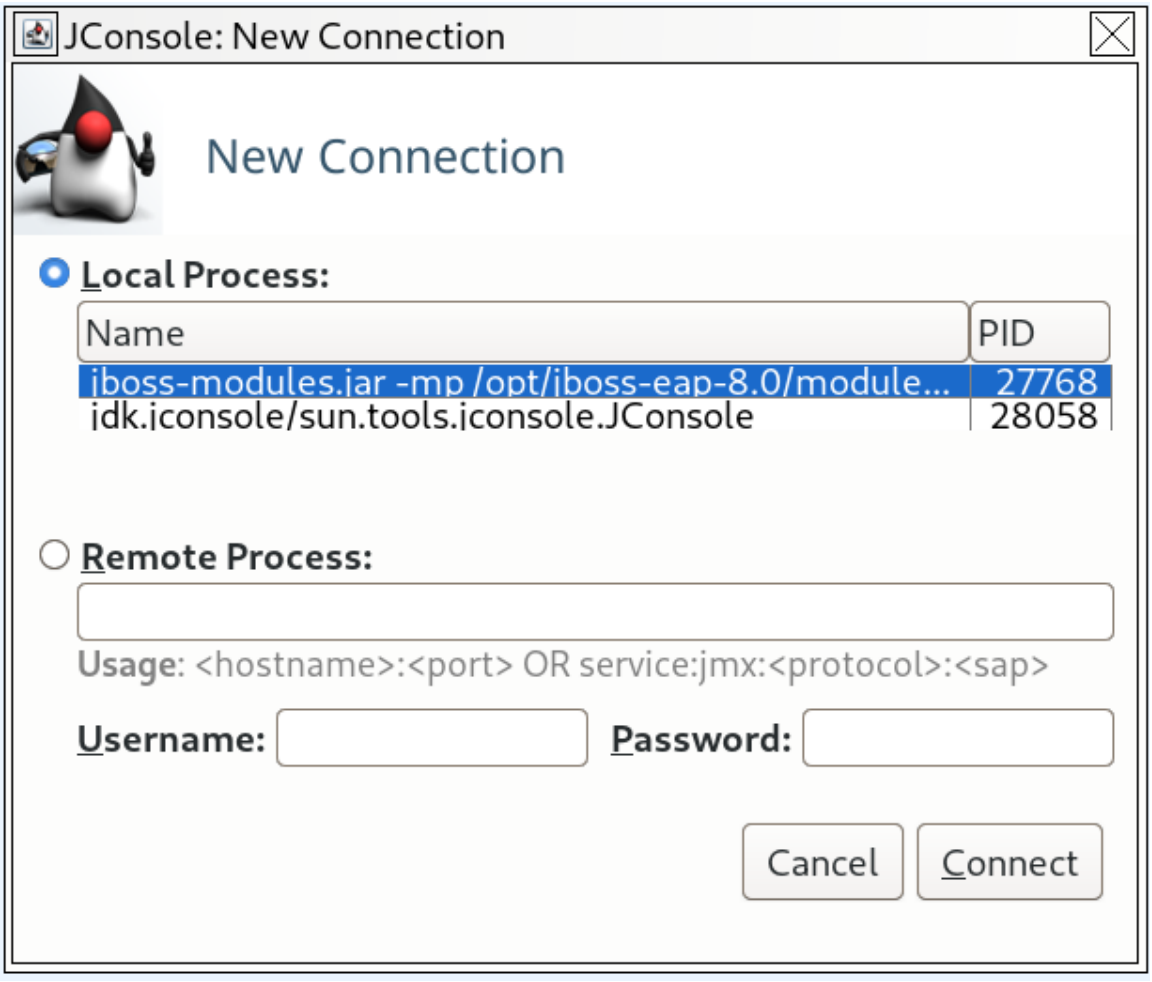
2.2.1. 使用 JConsole 连接到本地 JBoss EAP JVM

连接到在与 JConsole 相同的计算机上运行的 JBoss EAP JVM：

流程

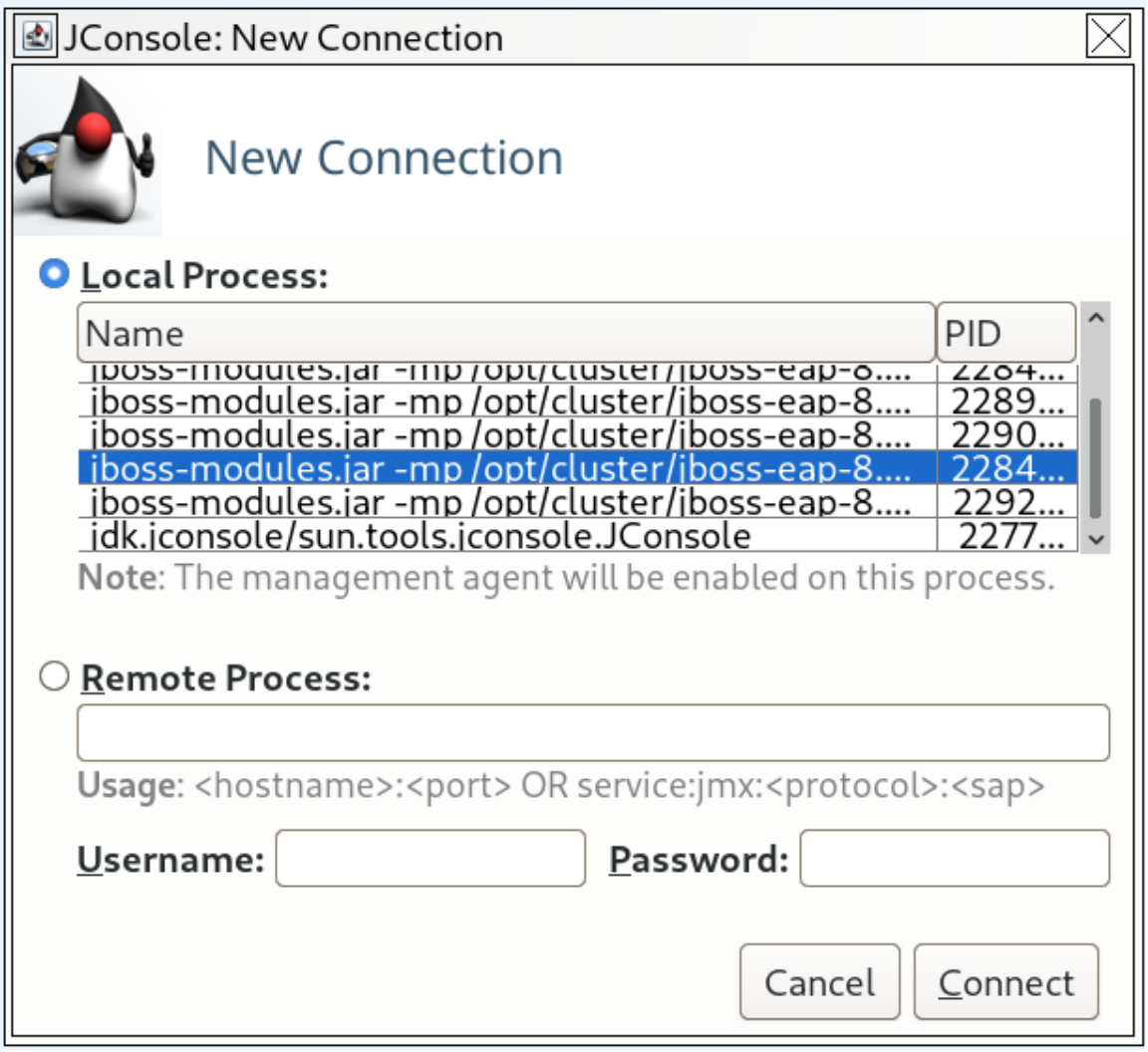
1. 在 ***EAP_HOME*/bin** 中运行 **jconsole** 脚本。
2. 在 **Local Process** 下，选择要监控的 JBoss EAP JVM 进程。
   - 对于独立的 JBoss EAP 服务器，有一个 JBoss EAP JVM 进程。

图 2.1. JConsole Local Standalone JBoss EAP Server JVM



- JBoss EAP 受管域主机有多个 JVM 进程，您可以连接：主机控制器 JVM 进程、进程控制器 JVM 进程，以及主机上每个 JBoss EAP 服务器的 JVM 进程。您可以通过查看 JVM 参数来确定已连接到的 JVM。

图 2.2. JConsole 本地受管域 JBoss EAP JVM



3. 点 **连接**。

2.2.2. 使用 JConsole 连接到远程 JBoss EAP JVM

您可以使用 **jconsole** 脚本使用 JConsole 连接到远程 JBoss EAP JVM。

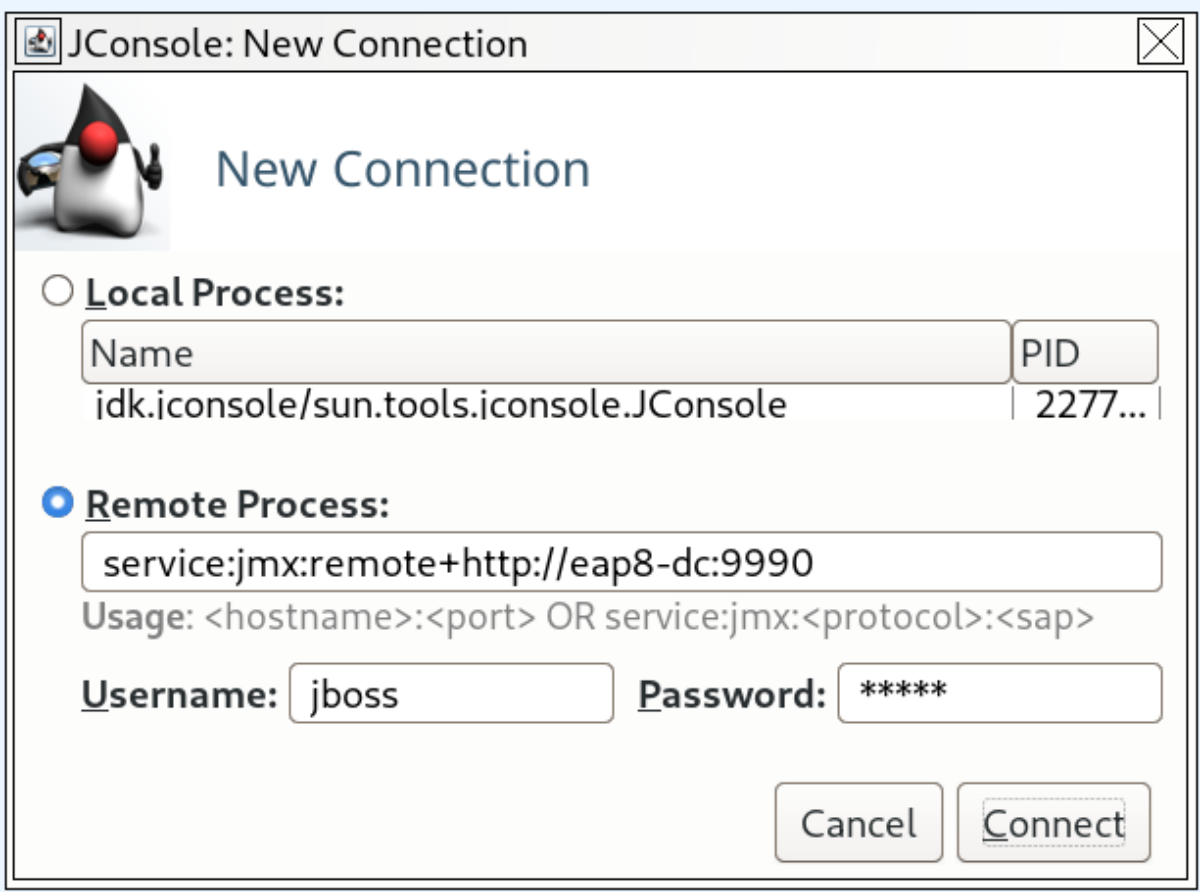
先决条件

- 配置 JBoss EAP 以进行远程监控连接。
- 将 JBoss EAP 的 ZIP 安装下载到您的本地计算机。

流程

1. 在 ***EAP_HOME*/bin** 中运行 **jconsole** 脚本。
2. 在 **Remote Process** 下，插入要监控的远程 JBoss EAP JVM 进程的 URI。

图 2.3. JConsole Remote JBoss EAP JVM



3. 确保您为监控连接提供用户名和密码。
4. 点 **连接**。

2.3. JAVA MISSION CONTROL

开发人员可以使用 Java Mission Control (JMC)来识别性能问题。JMC 是一个分析和诊断工具，它由以下组件组成：

- **Java 虚拟机(JVM)浏览器**，以查看正在运行的 Java 应用程序及其相关 JVM。
- 用于监控 JVM 的 **Java 管理(JMX)控制台**。
- **Java Flight Recorder (JFR)**，从正在运行的 Java 应用程序收集诊断数据。
- 用于堆转储分析的插件。

其他资源

- **有关将 JMC 连接到本地或远程 JBoss EAP JVM 的更多信息，请参阅如何在红帽客户门户网站上通过 EAP 远程连接 Java Mission Control?。**
- **有关下载和安装 JMC 的更多信息，**请参阅 Oracle *JDK Mission Control 用户指南中的安装 JDK Mission Control 和支持的插件* **部分。**

第 3 章 诊断性能问题

3.1. 启用垃圾回收日志记录

在尝试对 Java 性能问题进行故障排除时，检查垃圾回收日志非常有用，特别是与内存用量相关的内容。

除了编写日志文件的其他磁盘 I/O 活动外，启用垃圾回收日志记录不会显著影响服务器性能。

对于在 OpenJDK 或 Oracle JDK 上运行的独立 JBoss EAP 服务器，已默认启用垃圾回收日志记录。对于 JBoss EAP 受管域，可以为主机控制器、处理控制器或单独的 JBoss EAP 服务器启用垃圾回收日志记录。

1. 获取正确的 JVM 选项，用于为您的 JDK 启用垃圾回收日志记录。在下面的选项中替换您要创建日志的路径。

注意

红帽客户门户网站有一个 JVM 选项配置工具，可帮助您生成最佳 JVM 设置。

- 对于 OpenJDK 11 或 Oracle JDK 11：

```
-verbose:gc -Xloggc:<path_to_directory>/gc.log -XX:+PrintGCDetails -
XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=5 -
XX:GCLogFileSize=3M -XX:-TraceClassUnloading
```

- 对于版本 9 或更高版本的 OpenJDK、Oracle JDK 或支持 JEP 271 的任何 JDK：

```
-Xlog:gc*:file=<path_to_directory>/gc.log:time,uptimemillis:filecount=5,filesize=3M
```

其他资源

- 有关 JEP 271 的更多信息，请参阅 OpenJDK 网页上的 JEP 271: Unified GC Logging。

3.2. JAVA 堆转储

Java 堆转储是在特定时间点创建的 JVM 堆的快照。创建和分析堆转储可用于诊断和故障排除 Java 应用程序的问题。

根据您使用的 JDK，可以为 JBoss EAP 进程创建和分析 Java 堆转储的不同方法。本节介绍 Oracle JDK 和 OpenJDK 的常见方法。

3.2.1. 使用 OpenJDK 和 Oracle JDK 创建堆转储

3.2.1.1. 创建一个按需堆转储

您可以使用 **jcmd** 命令为在 OpenJDK 或 Oracle JDK 上运行的 JBoss EAP 创建按需堆转储。

流程

1. 确定您要从中创建堆转储的 JVM 进程 ID。
2. 使用以下命令创建堆转储：

```
$ jcmd JAVA_PID GC.heap_dump -all=true FILENAME.hprof
```

这会以 HPROF 格式创建一个堆转储文件，通常位于 ***EAP_HOME*** 或 ***EAP_HOME*/bin** 中。或者，您可以指定到另一个目录的文件路径。

3.2.1.2. 在 OutOfMemoryError 上自动创建一个堆转储

您可以使用 **-XX:+HeapDumpOnOutOfMemoryError** JVM 选项在引发 **OutOfMemoryError** 异常时自动创建堆转储。

这会以 HPROF 格式创建一个堆转储文件，通常位于 ***EAP_HOME*** 或 ***EAP_HOME*/bin** 中。或者，您可以使用 **-XX:HeapDumpPath=*/path*/**为堆转储设置自定义路径。如果您使用 **-XX:HeapDumpPath** 指定文件名，例如 **-XX:HeapDumpPath=*/path/filename.hprof*，**堆转储将相互覆盖。

3.2.2. 分析堆转储

3.2.2.1. Heap dump 分析工具

有很多工具可以分析堆转储文件并帮助识别问题。红帽支持建议使用 Eclipse Memory Analyzer 工具 (MAT)，它可以分析以 HPROF 或 PHD 格式格式化的堆转储。

其他资源

有关使用 Eclipse MAT 的详情，请参考 Eclipse MAT 文档。

3.2.2.2. 堆转储分析提示

有时，堆性能问题的原因很明显，但有些时候，您可能需要了解应用代码以及导致 **OutOfMemoryError** 等问题的具体情况。这有助于识别问题是否为内存泄漏，或者堆是否只是足够大。

识别内存用量问题的一些建议包括：

- 如果未找到单个对象来消耗太多内存，请尝试按类分组来查看很多小对象是否消耗大量内存。
- 检查内存的最大使用是否为线程。如果 **OutOfMemoryError**-triggered heap dump 比指定的 **Xmx** 最大堆大小小，则这的一个良好指示符是。
- 使内存泄漏更加可检测的技术是临时将正常的最大堆大小加倍。当发生 **OutOfMemoryError** 时，与内存泄漏相关的对象大小大约是堆大小的一半。

当确定了内存问题的来源时，您可以查看垃圾回收根中的路径，以查看对象处于活动状态。

3.3. JAVA FLIGHT RECORDER

3.3.1. 关于 Java Flight Recorder

Oracle *JDK Mission Control 用户指南*将 Java Flight Recorder (JFR)描述为"profiling and event collection framework"。开发人员可以将 JFR 与 JDK Mission Control (JMC)搭配使用，以收集有关 Java 虚拟机(JVM)和其他 Java 应用程序的数据。开发人员可以使用这些数据来识别和修复性能问题。

JFR 经过仔细设计，因此需要低程度的开销（消耗资源）。这意味着，JFR 分析可在某些生产环境中持续运行，且影响最小。开发人员可以使用 JFR 和 JMC 在事件后快速分析运行时信息。

注意

JFR 由 Java OpenJDK 8u262 或更高版本提供，作为 Java Diagnostic Command Tool 的一部分。

其他资源

- 有关 JFR 的更多信息，请参阅 Oracle *JDK Mission Control 用户指南中的Flight Recorder 部分*。

3.3.2. Java Flight Recorder 分析配置

开发人员可以修改性能分析配置，以自定义 Java Flight Recorder (JFR)的实例。JFR 提供两个不同的性能分析配置：

- 默认 ：提供稀疏信息抽样 ; 低性能分析详情
- 配置集 ：提供更全面的信息抽样；中文性能分析详细信息

开发人员可以修改任何配置文件，以启用额外的事件指标抽样。

3.3.3. 启用 Java Flight Recorder 配置文件捕获

开发人员可以使用 Java Flight Recorder (JFR)在裸机或 Red Hat OpenShift Container Platform 上对 JBoss EAP 安装进行性能分析。

要了解在 OpenShift 中使用 JFR 的信息，请参阅对 Cryostat：容器的 JDK Flight Recorder 简介。

3.3.3.1. 在裸机上启用 Java Flight Recorder 分析

开发人员可以使用命令行或 Java Mission Control (JMC)桌面应用程序启动 Java Flight Recorder (JFR)配置集。

3.3.3.2. 使用 Java 虚拟机配置标志为 JBoss EAP 配置 Java Flight Recorder 分析

您可以使用配置标志在 Java 虚拟机(JVM)上使用 JBoss EAP 配置 Java Flight Recorder (JFR)分析。

JVM 配置示例

```
-XX:StartFlightRecording=delay=15s,duration=60s,name=jboss-eap-profile,
filename=C:\TEMP\jboss-eap-profile.jfr,settings=default
```

通过 **StartFlightRecording=delay** 配置标志，您可以在启动性能分析会话前设置 JVM 引导后等待的时间 JFR。在前面的示例中，**StartFlightRecording=delay** 设置为 15 秒，这意味着性能分析将在 15 秒延迟后启动。

持续时间 配置标志允许您设置每个性能分析会话的时间长度。在前面的示例中，**持续时间** 设置为 60 秒。

name 配置标志允许您在内存配置集名称中设置。在本例中，内存配置文件名称设置为 **jboss-eap-profile**。

filename 配置标志允许您设置要保存文件的文件名和路径。在本例中，**filename** 设置为 **C:\TEMP\jboss-eap-profile.jfr**。

通过设置 配置标志，您可以选择性能分析配置。在本例中， 设置为默认。请注意，配置集配置的文件扩展已被排除。

分析会话完成后，将在 filename 选项定义的文件路径上创建文件。

3.3.3.3. 使用 Java 命令工具对正在运行的 JBoss EAP Java 虚拟机进行性能分析

您可以使用 Java Flight Recorder (JFR) JFR.start 命令配置正在运行的 JBoss EAP Java 虚拟机(JVM)以使用 Java 命令工具 jcmd 进行性能分析。

流程

- 使用以下命令之一：
  - 对于 Linux 操作系统：

    ```
    $ jcmd <PID> JFR.start duration=TIME filename=path/to/YOUR_PROFILE_NAME.jfr
    ```

    例如：

    用于 Linux 的 *JFR.start* 命令

    ```
    $ jcmd <PID> JFR.start duration=60s filename=/tmp/jboss-eap-profile.jfr
    ```

    JFR 分析会话启动后，您将收到以下确认消息：

    ```
    $ jcmd <PID> JFR.start duration=60s filename=/tmp/jboss-eap-profile.jfr
    <PID>:
    Started recording 1. The result will be written to:

    /tmp/jboss-eap-profile.jfr
    ```

  - 对于 Windows 操作系统：

    ```
    > jcmd.exe <PID> JFR.start duration=TIME filename=path/to/YOUR_PROFILE_NAME.jfr
    ```

    例如：

    用于 Windows 的 *JFR.start* 命令示例

    ```
    > jcmd.exe <PID> JFR.start duration=60s filename=C:\TEMP\jboss-eap-profile.jfr
    ```

    JFR 分析会话启动后，您将收到以下确认消息：

    ```
    > jcmd.exe <PID> JFR.start duration=60s filename=C:\TEMP\jboss-eap-profile.jfr
    <PID>:
    Started recording 1. The result will be written to:

    C:\TEMP\jboss-eap-profile.jfr
    ```

duration 选项允许您设置每个性能分析会话的时间长度。在前面的示例命令中，持续时间 设置为 60 秒。

filename 选项允许您设置要保存该文件的文件名和路径。在前面的示例命令中，在 Linux 示例中将 **filename** 设置为 **/tmp/jboss-eap-profile.jfr**，在 Windows 示例中为 **C:\TEMP\jboss-eap-profile.jfr**。

3.3.3.4. 使用 Java Mission Control 连接本地 Java 虚拟机

您可以使用 Java Mission Control (JMC)连接在与 JMC 实例相同的服务器上运行的本地 Java 虚拟机 (JVM)。

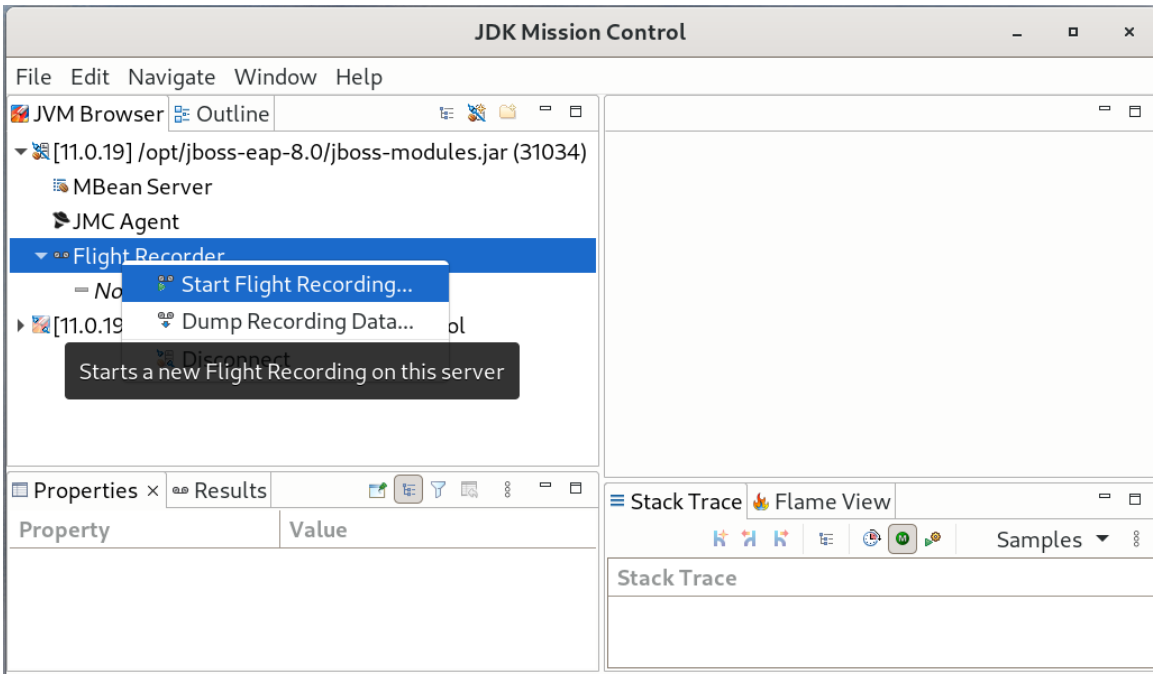
先决条件

- 配置了 JBoss EAP 库的 Java Mission Control。具体步骤请查看 如何远程连接 Java Mission Control?
- JBoss EAP 被配置为远程监控连接，并且在 **ApplicationRealm** 中创建用于监控的用户。

流程

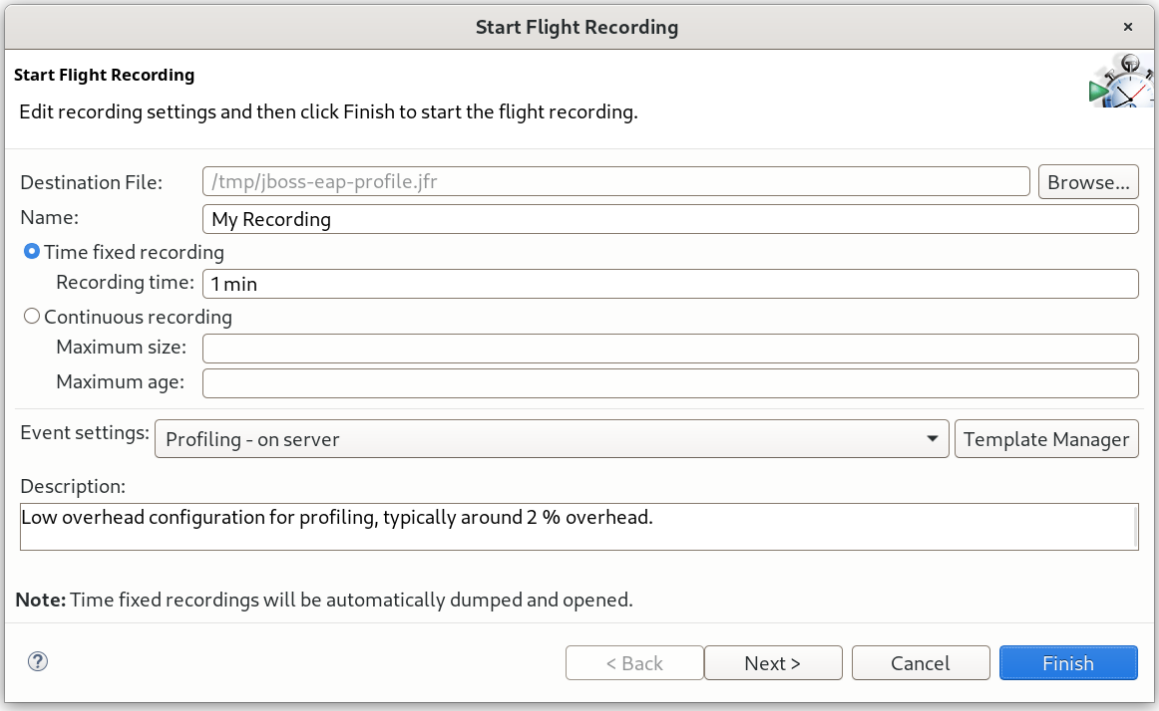
1. Open Java Mission Control。
2. 在 JVM 浏览器 窗格中，选择 JVM to profile。
3. 展开 JVM 的下拉菜单，以显示 Flight Recorder 项。
   a. 右键单击 Flight Recorder 打开子菜单，选择 Start Flight Recording...。

   图 3.1. JMC 中的 JVM 浏览器



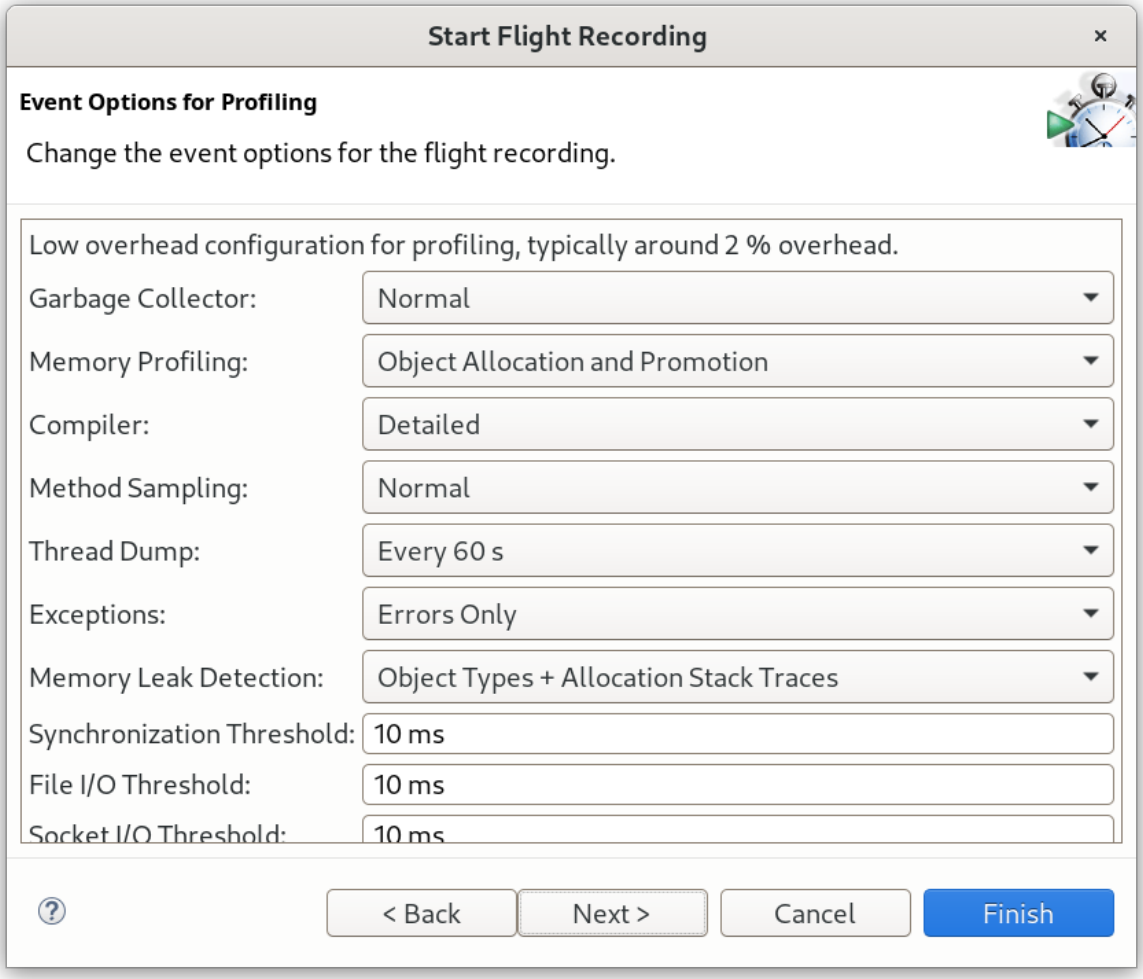
4. 在 Start Flight Recording窗口中，配置选项用于性能分析。

图 3.2. JVM 分析设置



5. 单击 Next 以了解详细的低级别设置。

图 3.3. JVM 分析高级设置



6. 单击 Finish 以开始性能分析。

3.3.3.5. 使用 Java Mission Control 连接远程 Java 虚拟机

您可以使用 Java Mission Control (JMC)连接到远程 Java 虚拟机(JVM)配置集。

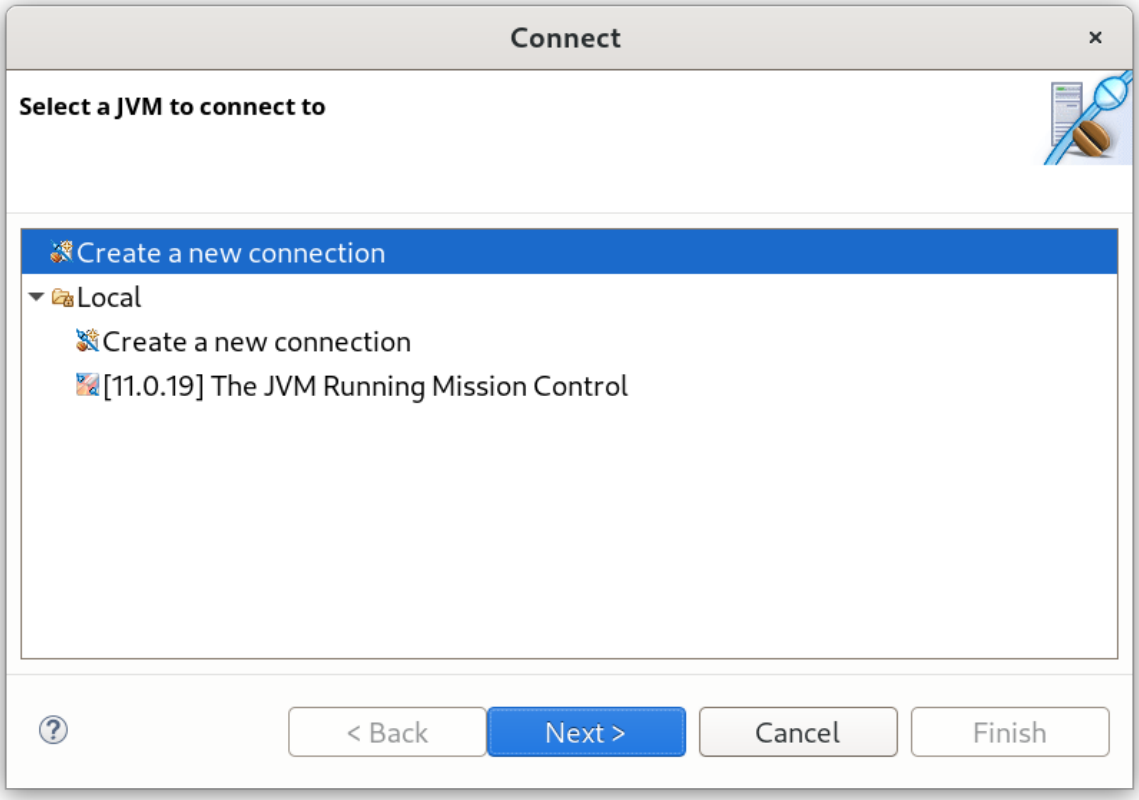
先决条件

- 使用 JBoss EAP 库配置 Java Mission Control。具体步骤请查看 如何远程连接 Java Mission Control?
- 配置 JBoss EAP 以进行远程监控连接，并在 ApplicationRealm 中创建用于监控的用户。

流程

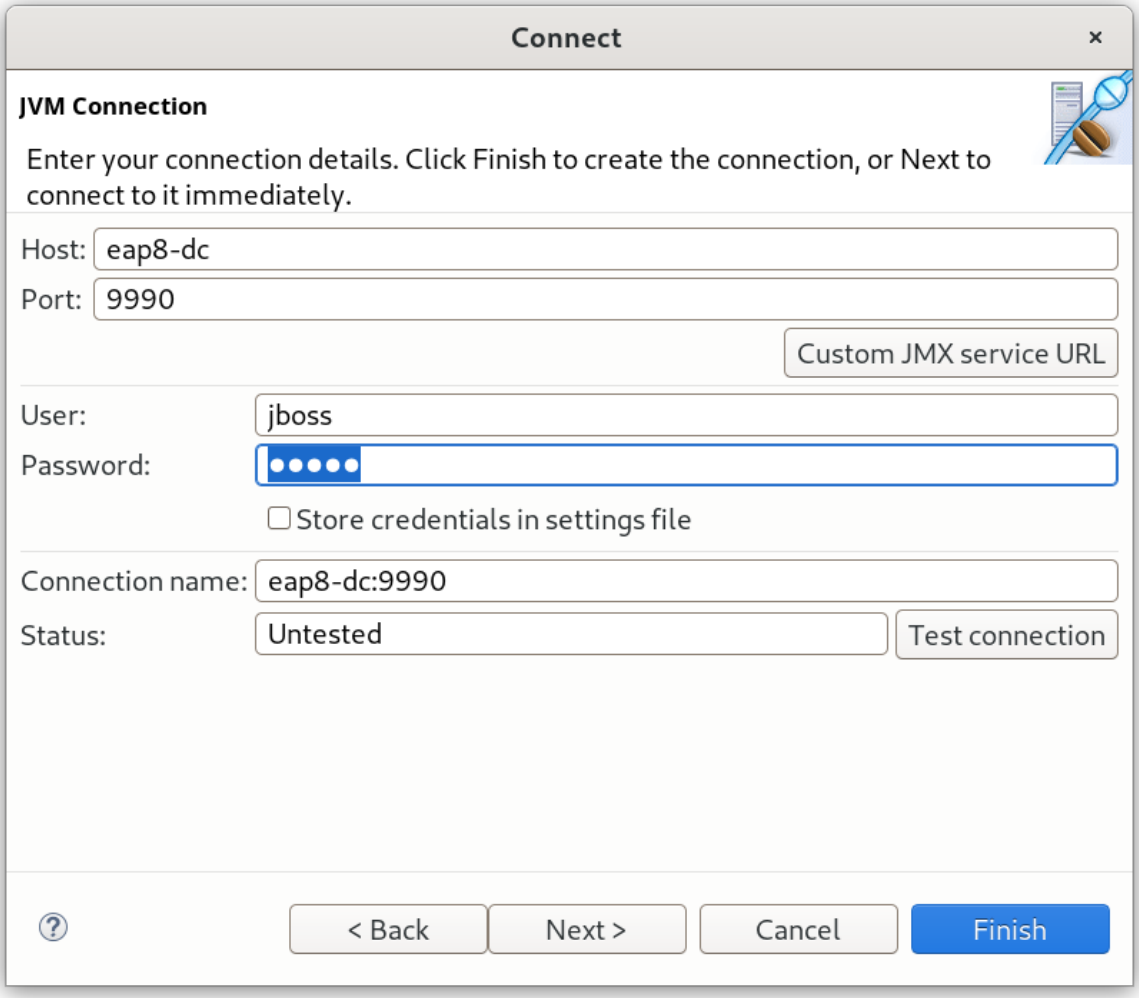
1. Open Java Mission Control。
2. 在 File 菜单中，选择 Connect。
3. 在 Connect 窗口中，选择 *Create a new connection*，然后单击 Next。

   图 3.4. JMC 中的连接窗口



4. 在 *JVM 连接*窗口中，完成要配置文件的远程 JBoss EAP JVM 的详细信息。

图 3.5. JMC 中的 JVM 连接详情



a. 在 Host 字段中，添加您的主机名或 IP 地址。

b. 在 Port 字段中，添加您的端口号。

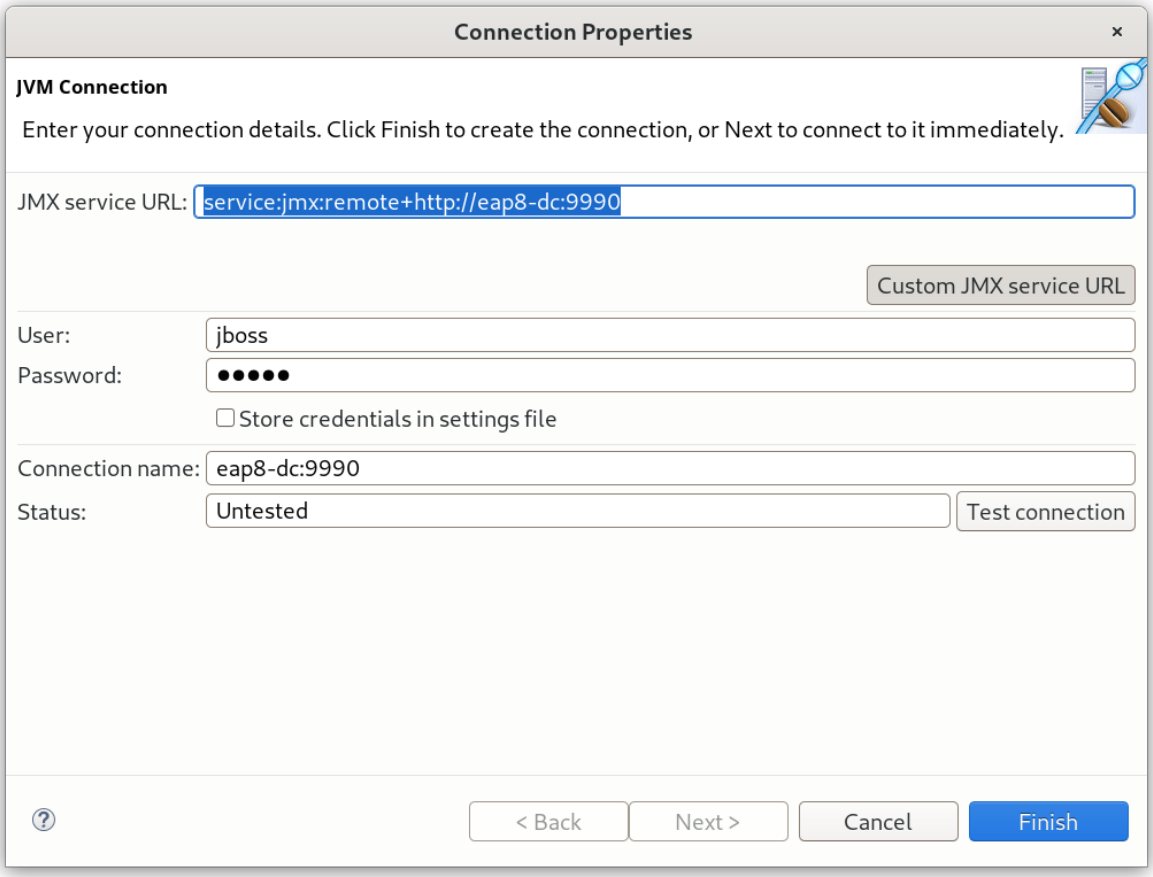
c. 在 User 字段中，添加您在 ApplicationRealm 中创建的用户。

d. 在 Password 字段中，添加在 ApplicationRealm 中创建的密码。

e. 可选 要将凭据存储在设置文件中，请点击在设置文件 中存储凭据 旁边的复选框。

5. 点 Custom JMX Service URL覆盖默认设置。

图 3.6. 用于 JVM 连接的 JMX 服务 URL

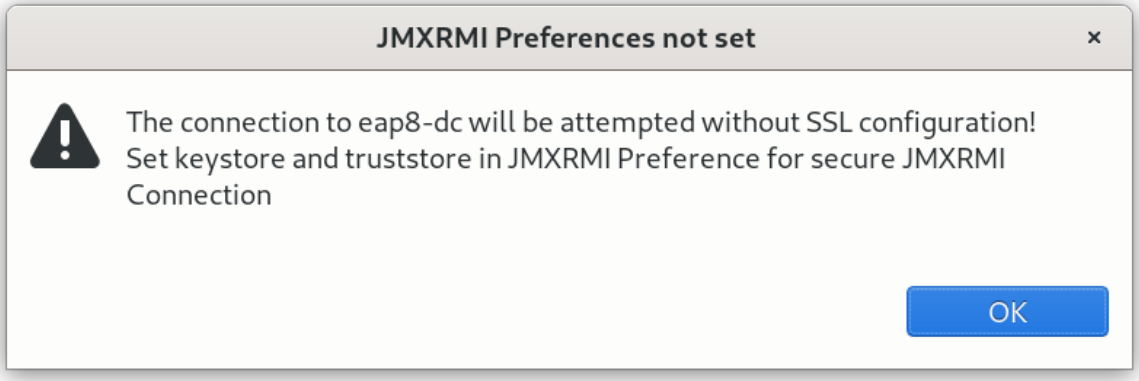


6. 更改 JMX 服务 URL 以定义 JBoss 远程协议。

```
service:jmx:remote+http://<host>:9990
```

7. 点 Test connection 验证您的设置。
8. 点 Finish 保存您的设置。
9. 这时将显示 *JMXRMI Preferences 未设置* 警告信息。

图 3.7. JMXRMI 首选项警告信息



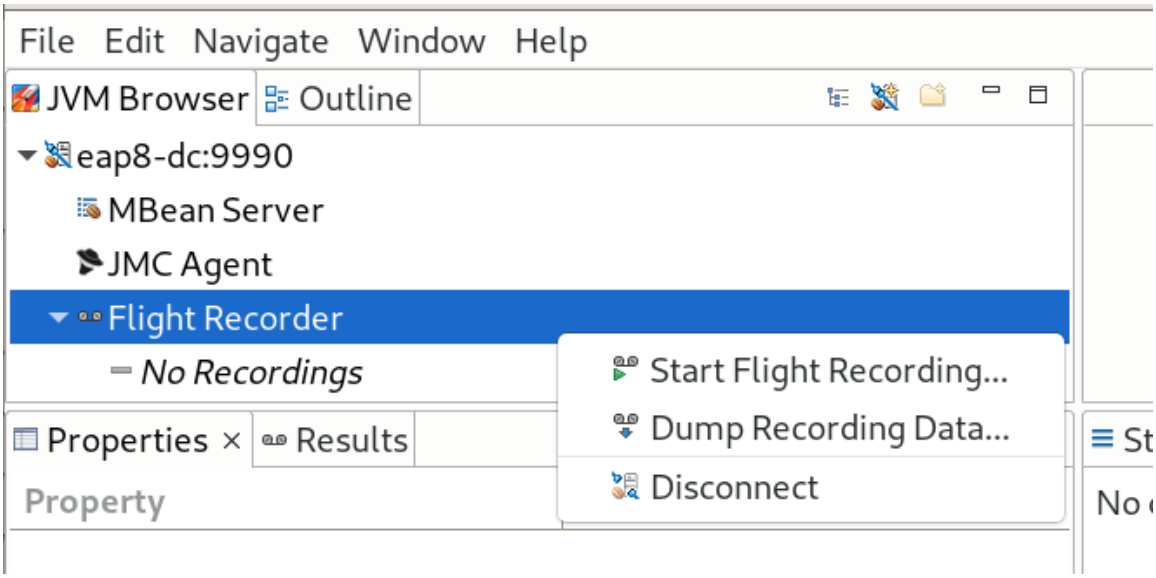
a. 单击 OK 以接受连接尝试。

10. 在 JVM 浏览器窗格中，选择 JVM to profile。

a. 展开 JVM 的下拉菜单，以显示 Flight Recorder 项。

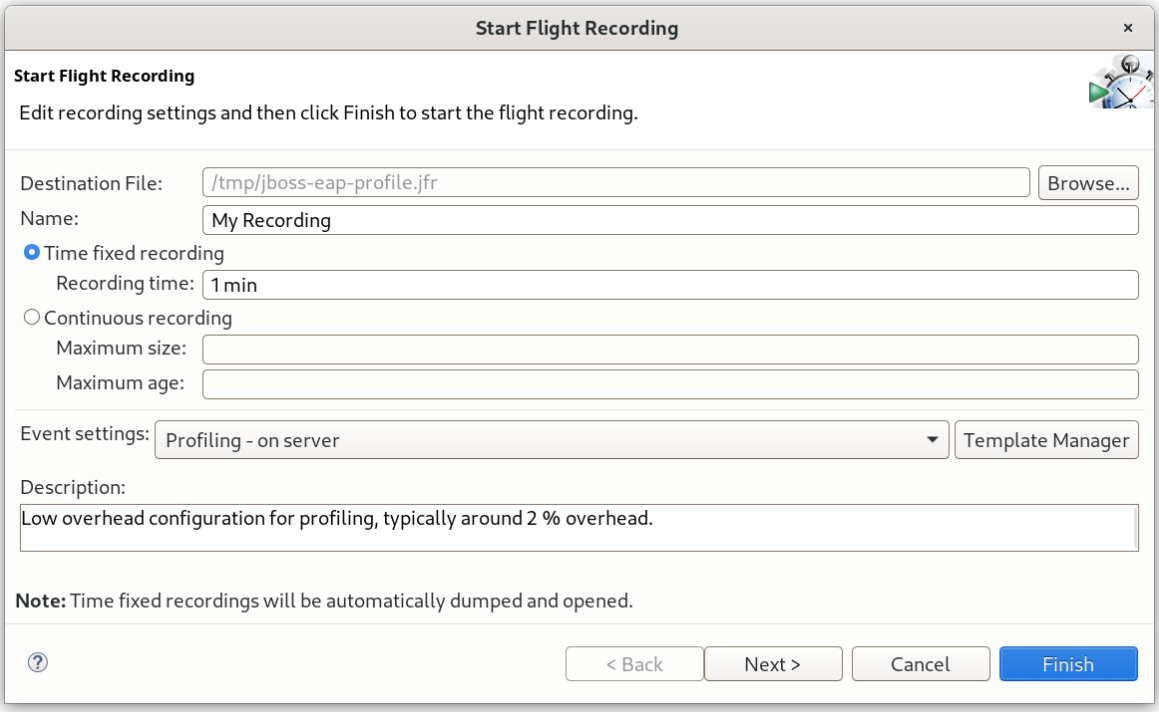
b. 右键单击 Flight Recorder 打开子菜单，然后选择 Start Flight Recording...。

图 3.8. 使用 JMC 配置集菜单连接远程 JVM



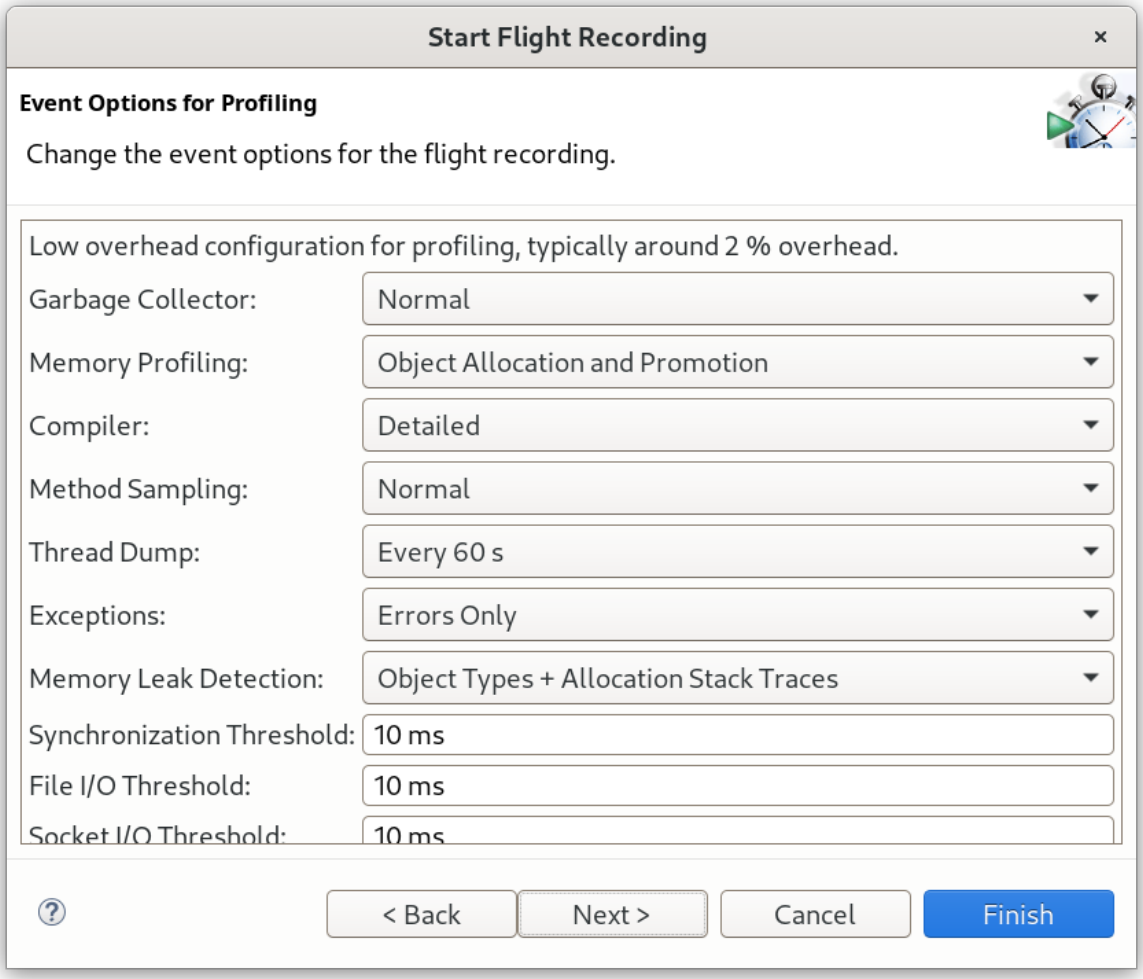
11. 在 Start Flight Recording窗口中，配置选项用于性能分析。

图 3.9. JVM 分析设置



12. 单击 Next 以了解详细的低级别设置。

图 3.10. JVM 分析高级设置



13. 单击 Finish 以开始性能分析。

3.4. 识别 JAVA 线程高 CPU 使用率

注意

对于在 Red Hat Enterprise Linux 或 Solaris 上使用 JBoss EAP 的客户，红帽客户门户网站上的 JVMPeg 实验室工具可帮助 收集和分析 Java 线程信息来识别高 CPU 使用率。按照 使用 JVMPeg 实验工具的说明，而不是使用以下步骤。

对于 OpenJDK 和 Oracle JDK 环境，可使用 jstack 工具提供 Java 线程诊断信息。

1. 识别使用高 CPU 百分比的 Java 进程的进程 ID。
在高使用过程中获取每个线程 CPU 数据也很有用。这可以通过在 Red Hat Enterprise Linux 系统上使用 top -H 命令来完成。

2. 使用 jstack 实用程序，创建 Java 进程的堆栈转储。例如，在 Linux 和 Solaris 上：

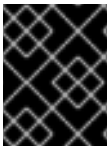
```
jstack -l JAVA_PROCESS_ID > high-cpu-tdump.out
```

您可能需要创建多个转储，以便查看一段时间内的任何更改或趋势。

3. 分析堆栈转储。您可以使用 Thread Dump Analyzer (TDA) 等工具。

3.5. 受管 EXECUTOR 服务以及受管调度的 EXECUTOR 服务的运行时统计信息

您可以通过查看管理 CLI 属性生成的运行时统计信息来监控受管 executor 服务的性能和管理的 executor 服务。您可以查看独立服务器的运行时统计信息，或查看映射到主机的单个服务器。



重要

domain.xml 配置不包括运行时统计管理 CLI 属性的资源，因此您无法使用管理 CLI 属性来查看受管域的运行时统计信息。

表 3.1. 显示管理 CLI 属性，用于监控受管 executor 服务以及受管调度的 executor 服务的性能。

属性	描述
active-thread-count	主动执行任务的大约线程数量。
completed-task-count	完成执行的任务大约总数。
hung-thread-count	挂起的 executor 线程数量。
max-thread-count	executor 线程的最大数量。
current-queue-size	executor 任务队列的当前大小。
task-count	已提交用于执行的任务大约总数。
thread-count	当前执行者线程数量。

查看在独立服务器上运行的受管 executor 服务的运行时统计信息示例。

```
[standalone@localhost:9990 /] /subsystem=ee/managed-executor-service=default:read-resource(include-runtime=true,recursive=true)
```

在独立服务器上运行的受管调度的 executor 服务的运行时统计信息示例。

```
[standalone@localhost:9990 /] /subsystem=ee/managed-scheduled-executor-service=default:read-resource(include-runtime=true,recursive=true)
```

查看映射到主机的服务器上运行的受管 executor 服务的运行时统计信息示例。

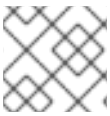
```
[domain@localhost:9990 /] /host=<host_name>/server=<server_name>/subsystem=ee/managed-executor-service=default:read-resource(include-runtime=true,recursive=true)
```

在映射到主机的服务器上运行的受管调度的 executor 服务的运行时统计信息示例。

```
[domain@localhost:9990 /] /host=<host_name>/server=<server_name>/subsystem=ee/managed-scheduled-executor-service=default:read-resource(include-runtime=true,recursive=true)
```

第 4 章 JVM 调整

为应用程序和 JBoss EAP 环境配置最佳 JVM 选项是调优性能的最基本的方法之一。本章论述了配置一些常规 JVM 选项。



注意

本章中列出的许多 JVM 选项可以使用红帽客户门户上的 JVM 选项配置工具 轻松生成。

4.1. 设置固定堆大小

您必须设置适当的堆大小以防止内存不足。

-Xms 选项设置初始堆大小，**-Xmx** 设置最大堆大小。建议您在生产环境中使用，将初始和最大堆大小选项设置为相同的大小，以便固定和预分配堆大小。

例如，以下选项设定 2048 MB 堆大小：

```
-Xms2048M -Xmx2048M
```

建议您在开发环境中的负载下测试应用程序，以确定最大内存用量。您的生产堆大小应至少比测试的最大值高 25%，从而为开销留出空间。

4.2. 配置垃圾收集器

虽然并行垃圾收集器（也称为吞吐量垃圾收集器）是服务器级机器的 Java 8 中的默认垃圾收集器，但红帽建议使用 G1 垃圾收集器，这是 Java 9 后的默认设置。在大多数情况下，G1 垃圾收集器通常比 CMS 和并行垃圾收集器更好地执行。

要启用 G1 收集器，请使用以下 JVM 选项：

```
-XX:+UseG1GC
```

4.2.1. 垃圾回收日志记录选项

独立 JBoss EAP 服务器默认启用垃圾回收日志记录。

4.3. 启用巨页

为 JBoss EAP JVM 启用大页面会导致内存锁定的页面，不能像常规内存一样被交换到磁盘。

特别是对于内存密集型应用程序，使用大页面的优点是无法将堆分页或交换到磁盘，因此始终可用。

使用大型页面的一个缺点是，在系统中运行的其他进程可能无法快速访问内存，这可能会导致这些进程出现过量分页。

与任何其他性能配置更改一样，建议您测试测试测试环境中更改的影响。

1. 您必须确保您的操作系统配置允许进程使用大型页面。
    - 对于 Red Hat Enterprise Linux 系统，您必须明确配置 HugeTLB 页面，以确保 JBoss EAP 进程可以访问大页。

- 对于 Windows Server 系统，运行 JBoss EAP 的用户必须分配有大页面权限：
  1. 选择 Control Panel → Administrative Tools → Local Security Policy。
  2. 选择 Local Policies → User Rights Assignment。
  3. 双击 内存中的锁定页面。
  4. 添加您要使用大页面的 Windows Server 用户和用户组。
  5. 重启机器。

2. 启用或禁用大页面支持：
   - 要显式启用 JBoss EAP JVM 的大型页面支持，请使用以下 JVM 选项：

     ```
     -XX:+UseLargePages
     ```

   - 要显式禁用 JBoss EAP JVM 的大型页面支持，请使用以下 JVM 选项：

     ```
     -XX:-UseLargePages
     ```

3. 启动 JBoss EAP 时，确保没有与保留内存相关的警告。
   - 在 Red Hat Enterprise Linux 中，错误可能类似如下：

     ```
     OpenJDK 64-Bit Server VM warning: Failed to reserve shared memory. (error = 1)
     ```

   - 在 Windows Server 中，错误可能类似如下：

     ```
     Java HotSpot(TM) 64-Bit Server VM warning: JVM cannot use large page memory
     because it does not have enough privilege to lock pages in memory.
     ```

   如果确实看到警告，请验证您的操作系统配置和 JVM 选项是否已正确配置。

如需更多信息，请参阅 Oracle 文档有关 Java 对大页的支持。

4.4. 设置 ULIMITS

对于 Red Hat Enterprise Linux 和 Solaris 平台，您必须为 JBoss EAP JVM 进程配置适当的 ulimit 值。可以临时超过 "soft" ulimit，而 "hard" ulimit 是资源用法的严格 ceiling。根据您的环境和应用程序，适当的 ulimit 值会有所不同。

如果应用到 JBoss EAP 进程的限制太低，您会在启动 JBoss EAP 时看到类似如下的警告：

```
WARN  [org.jboss.as.warn.fd-limit] (main) WFLYSRV0071: The operating system has limited the
number of open files to 1024 for this process; a value of at least 4096 is recommended.
```

要查看您当前的 ulimit 值，请使用以下命令：

- 对于 soft ulimit 值：

  ```
  ulimit -Sa
  ```

- 对于 hard ulimit 值：

```
ulimit -Ha
```

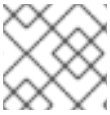
要为最大打开文件数设置 ulimit，请使用以下命令以及您要应用的编号：

- 要为最大打开文件数设置 soft ulimit ：

```
ulimit -Sn 4096
```

- 要为最大打开文件数设置 hard ulimit ：

```
ulimit -Hn 4096
```



注意

为了保证 ulimit 设置有效，建议在生产系统上将软和硬限制设置为相同的值。

其他资源

有关使用配置文件设置 ulimit 值的更多信息，请参阅如何在客户门户网站中设置 ulimit 值。

4.5. 主机控制器和进程控制器 JVM 调整

JBoss EAP 受管域主机对主机控制器和流程控制器具有单独的 JVM。

您可以调整主机控制器和流程控制器 JVM 设置，但即使在大型的受管域环境中，主机控制器和进程控制器的默认 JVM 配置也应可正常运行。

主机控制器和流程控制器 JVM 的默认配置已测试，受管域大小为 20 个 JBoss EAP 主机每个运行 10 个 JBoss EAP 服务器，总计有 200 个 JBoss EAP 服务器的域大小。

如果您在大型受管域时遇到问题，您可能需要监控环境中的主机控制器或进程控制器 JVM，以确定 JVM 选项的适当值，如堆大小。

第 5 章 JAKARTA ENTERPRISE BEANS 子系统调整

JBoss EAP 可以缓存 Jakarta Enterprise Beans 以节省初始化时间。这通过使用 bean 池来完成。

JBoss EAP 中可以调整两个不同的 bean 池： bean 实例池和 bean 线程池。

适当的 bean 池大小取决于您的环境和应用程序。建议您使用不同的 bean 池大小进行测试，并在模拟预期的真实条件的开发环境中执行压力测试。

5.1. BEAN 实例池

Bean 实例池用于无状态会话 Bean (SLSBs)和 Message Driven Beans (MDB)。默认情况下，SLSBs 使用实例池 **default-slsb-instance-pool**，MDB 使用实例池 **default-mdb-instance-pool**。

bean 实例池的大小限制了一次可以创建的特定企业 bean 的实例数量。如果特定企业 bean 的池已满，客户端将阻止并等待实例变为可用。如果客户端在池 的超时 属性中设置的时间内没有获得实例，则会抛出异常。

使用 derived-size 或 **max-pool -size** 来配置 bean 实例池的大小。**derived-size** 属性允许您使用以下值之一配置池大小：

- **from-worker-pools**，这表示从系统上配置的所有 worker 池的总线程大小派生出的最大池大小。
- **from-cpu-count**，这表示最大池大小派生自系统上可用处理器总数。请注意，这不一定是一个 1:1 映射，可能由其他因素增强。

如果 **derived-size** 未定义，则 **max-pool-size** 的值用于 bean 实例池的大小。

注意

derived **-size** 属性覆盖 **max-pool-size** 中指定的任何值。**derive-size** 必须未定义，才能使 **max-pool-size** 值生效。

您可以将企业 bean 配置为使用特定的实例池。这允许精细控制每个企业 bean 类型可用的实例。

5.1.1. 创建 bean 实例池

本节介绍如何使用管理 CLI 创建新的 bean 实例池。您还可以通过从 Configuration 选项卡导航到 Jakarta Enterprise Beans 子系统，然后选择 Bean Pool选项卡，来使用管理控制台配置 bean 实例池。

要创建新实例池，请使用以下命令之一：

- 创建具有最大池大小的 bean 实例组：

```
/subsystem=ejb3/strict-max-bean-instance-pool=POOL_NAME:add(derive-size=DERIVE_OPTION,timeout-unit=TIMEOUT_UNIT,timeout=TIMEOUT_VALUE)
```

以下示例创建一个名为 **my_derived_pool** 的 bean 实例池，最大从 CPU 数派生，超时为 2 分钟：

```
/subsystem=ejb3/strict-max-bean-instance-pool=my_derived_pool:add(derive-size=from-cpu-count,timeout-unit=MINUTES,timeout=2)
```

- 创建具有显式最大池大小的 bean 实例池：

```
/subsystem=ejb3/strict-max-bean-instance-pool=POOL_NAME:add(max-pool-
size=POOL_SIZE,timeout-unit=TIMEOUT_UNIT,timeout=TIMEOUT_VALUE)
```

以下示例创建一个名为 **my_pool** 的 bean 实例池，最大为 30 个实例，超时为 30 秒：

```
/subsystem=ejb3/strict-max-bean-instance-pool=my_pool:add(max-pool-size=30,timeout-
unit=SECONDS,timeout=30)
```

5.1.2. 指定 bean 应该使用的实例池

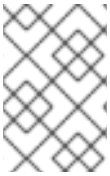
您可以通过利用 **@org.jboss.ejb3.annotation.Pool** 注释，或通过修改 bean 的**jboss-ejb3.xml** 部署描述符来设置特定的 Bean 将使用它的特定实例池。

5.1.3. 禁用默认的 bean 实例池

可以禁用默认的 bean 实例池，这会导致 enterprise bean 默认不使用任何实例池。相反，当线程需要调用企业 bean 的方法时，会创建一个新的企业 bean 实例。如果您不希望对所创建的企业 bean 实例数量有任何限制，这将非常有用。

要禁用默认的 bean 实例池，请使用以下管理 CLI 命令：

```
/subsystem=ejb3:undefine-attribute(name=default-slsb-instance-pool)
```

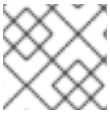


注意

If a bean is configured to use a particular bean instance pool, disabling the default instance pool does not affect the pool that the bean uses.

5.2. BEAN 线程池

默认情况下，名为 **default** 的 bean 线程池用于异步企业 bean 调用和企业 bean 计时器。



注意

从 JBoss EAP 7 开始，远程企业 bean 请求默认在 **io** 子系统中定义的 worker 中处理。

如果需要，您可以将每个企业 bean 服务配置为使用不同的 bean 线程池。如果您希望精细控制每个服务对 bean 线程池的访问，这将非常有用。

在确定适当的线程池大小时，请考虑您期望的并发请求数将一次处理。

5.2.1. 创建 bean 线程池

本节介绍如何使用管理 CLI 创建新的 bean 线程池。您还可以通过从 Configuration 选项卡导航到 Jakarta Enterprise Beans 子系统并在左侧菜单中选择 Container → Thread Pool，来使用管理控制台配置 bean 线程池。

要创建新线程池，请使用以下命令：

```
/subsystem=ejb3/thread-pool=POOL_NAME:add(max-threads=MAX_THREADS)
```

以下示例创建一个名为 my_thread_pool 的 bean 线程池，最大为 30 个线程：

```
/subsystem=ejb3/thread-pool=my_thread_pool:add(max-threads=30)
```

5.2.2. 配置企业 bean 服务以使用特定的 bean 线程池

企业 bean 异步调用服务和计时器服务都可以配置为使用特定的 bean 线程池。默认情况下，这两个服务都使用 默认的 bean 线程池。

本节介绍如何使用管理 CLI 配置上述企业 bean 服务以使用特定的 bean 线程池。您还可以通过从 Configuration 选项卡导航到 Enterprise Bean 子系统，选择 Services 选项卡并选择适当的服务，来使用管理控制台配置这些服务。

要将企业 bean 服务配置为使用特定的 bean 线程池，请使用以下命令：

```
/subsystem=ejb3/service=SERVICE_NAME:write-attribute(name=thread-pool-
name,value=THREAD_POOL_NAME)
```

使用您要配置的企业 bean 服务替换 SERVICE_NAME ：

- async 用于企业 bean 异步调用服务
- enterprise bean 计时器服务的 timer-service

以下示例将 enterprise bean async 服务设置为使用名为 my_thread_pool 的 bean 线程池：

```
/subsystem=ejb3/service=async:write-attribute(name=thread-pool-name,value=my_thread_pool)
```

5.3. BEAN 的运行时部署信息

您可以将运行时部署信息添加到 Bean 中，以进行性能监控。

有关可用运行时数据的详情，请查看 JBoss EAP 管理模型中的 ejb3 子系统。应用可以将运行时数据作为注解包含在 bean 代码或部署描述符中。应用程序可以使用这两个选项。

其他资源

- 有关可用运行时数据的更多信息，请参阅 JBoss EAP 管理模型中的 ejb3 子系统。

5.3.1. 用于从 Jakarta Enterprise Bean 检索运行时数据的命令行选项

Jakarta Enterprise Beans 中的运行时数据可从管理 CLI 获得，以便您可以评估 Jakarta Enterprise Beans 的性能。

获取所有 Bean 类型的运行时数据的命令使用以下模式：

```
/deployment=<deployment_name>/subsystem=ejb3/<bean_type>=<bean_name>:read-
resource(include-runtime)
```

将 <deployment_name> 替换为要检索运行时数据的部署 .jar 文件的名称。将 <bean_type> 替换为检索运行时数据的 bean 类型。以下选项对这个占位符有效：

- stateless-session-bean

- **stateful-session-bean**
- **singleton-bean**
- **message-driven-bean**

将 <bean_name > 替换为您要检索运行时数据的 bean 的名称。

系统会以 JavaScript 对象表示法(JSON)数据的形式提供 stdout 格式的结果。

在名为 ejb-management.jar的文件中部署了一个名为 ManagedSingletonBean 的单例 Bean 的运行时数据示例

```
/deployment=ejb-management.jar/subsystem=ejb3/singleton-bean=ManagedSingletonBean:read-resource(include-runtime)
```

单例 Bean 的输出运行时数据示例

```
{
    "outcome" => "success",
    "result" => {
        "async-methods" => ["void async(int, int)"],
        "business-local" => ["sample.ManagedSingletonBean"],
        "business-remote" => ["sample.BusinessInterface"],
        "component-class-name" => "sample.ManagedSingletonBean",
        "concurrency-management-type" => undefined,
        "declared-roles" => [
            "Role3",
            "Role2",
            "Role1"
        ],
        "depends-on" => undefined,
        "execution-time" => 156L,
        "init-on-startup" => false,
        "invocations" => 3L,
        "jndi-names" => [
            "java:module/ManagedSingletonBean!sample.ManagedSingletonBean",
            "java:global/ejb-management/ManagedSingletonBean!sample.ManagedSingletonBean",
            "java:app/ejb-management/ManagedSingletonBean!sample.ManagedSingletonBean",
            "java:app/ejb-management/ManagedSingletonBean!sample.BusinessInterface",
            "java:global/ejb-management/ManagedSingletonBean!sample.BusinessInterface",
            "java:module/ManagedSingletonBean!sample.BusinessInterface"
        ],
        "methods" => {"doIt" => {
            "execution-time" => 156L,
            "invocations" => 3L,
            "wait-time" => 0L
        }},
        "peak-concurrent-invocations" => 1L,
        "run-as-role" => "Role3",
        "security-domain" => "other",
        "timeout-method" => "public void sample.ManagedSingletonBean.timeout(javax.ejb.Timer)",
        "timers" => [{
            "time-remaining" => 4304279L,
            "next-timeout" => 1577768415000L,
```

```
            "calendar-timer" => true,
            "persistent" => false,
            "info" => "timer1",
            "schedule" => {
                "year" => "*",
                "month" => "*",
                "day-of-month" => "*",
                "day-of-week" => "*",
                "hour" => "0",
                "minute" => "0",
                "second" => "15",
                "timezone" => undefined,
                "start" => undefined,
                "end" => undefined
            }
        }],
        "transaction-type" => "CONTAINER",
        "wait-time" => 0L,
        "service" => {"timer-service" => undefined}
    }
}
```

在名为 ejb-management.jar的文件中部署的消息驱动的 bean 的运行时数据示例

```
/deployment=ejb-management.jar/subsystem=ejb3/message-driven-bean=NoTimerMDB:read-
resource(include-runtime)
```

message-driven bean 的输出示例

```
{
    "outcome" => "success",
    "result" => {
        "activation-config" => [
            ("destination" => "java:/queue/NoTimerMDB-queue"),
            ("destinationType" => "javax.jms.Queue"),
            ("acknowledgeMode" => "Auto-acknowledge")
        ],
        "component-class-name" => "sample.NoTimerMDB",
        "declared-roles" => [
            "Role3",
            "Role2",
            "Role1"
        ],
        "delivery-active" => true,
        "execution-time" => 0L,
        "invocations" => 0L,
```

```
        "message-destination-link" => "queue/NoTimerMDB-queue",
        "message-destination-type" => "javax.jms.Queue",
        "messaging-type" => "javax.jms.MessageListener",
        "methods" => {},
        "peak-concurrent-invocations" => 0L,
        "pool-available-count" => 16,
        "pool-create-count" => 0,
        "pool-current-size" => 0,
        "pool-max-size" => 16,
        "pool-name" => "mdb-strict-max-pool",
        "pool-remove-count" => 0,
        "run-as-role" => "Role3",
        "security-domain" => "other",
        "timeout-method" => undefined,
        "timers" => [],
        "transaction-type" => "CONTAINER",
        "wait-time" => 0L,
        "service" => undefined
    }
}
```

5.4. 可能需要指示企业 BEAN 子系统调整的例外

- **Stateless Jakarta Enterprise Beans 实例组不够大，或者超时太低**

```
javax.ejb.EJBException: JBAS014516: Failed to acquire a permit within 20 SECONDS
    at org.jboss.as.ejb3.pool.strictmax.StrictMaxPool.get(StrictMaxPool.java:109)
```

- **企业 bean 线程池不够大，或者企业 bean 需要比调用超时更长的时间**

```
java.util.concurrent.TimeoutException: No invocation response received in 300000
milliseconds
```

5.4.1. 有状态会话 Bean 的默认全局超时值

在 ejb3 子系统中，您可以使用 default-stateful-bean-session-timeout 属性为服务器实例上部署的所有有状态会话 Bean (SFSB)配置默认的全局超时值。

使用 default-stateful-bean-session-timeout 属性，您可以在 ejb3 子系统中使用以下管理 CLI 操作：

- 管理 CLI 中的 read-attribute 操作，以查看属性的当前全局超时值。

- 使用管理 CLI 配置属性的 write-attribute 操作。

属性行为因服务器模式而异。例如：

- 在单机服务器中运行时，配置的值将应用到应用服务器上部署的所有 SFSB。

- 在受管域中运行服务器时，服务器组内服务器实例上部署的所有 SFSB 都接收并发超时值。

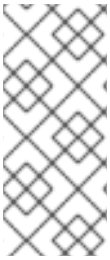
注意

当您更改属性的全局超时值时，更新的设置仅适用于新部署。您必须重新加载服务器，才能将新设置应用到当前部署。

默认情况下，属性值设为 -1， 这意味着部署的 SFSB 配置为永不超时。但是，您可以为属性配置以下类型的有效值：

- 当您将属性值设为 0 时，属性会立即标记被 ejb 容器移除的合格 SFSB。

- 当您设置大于 0 的属性值时，SFSB 会在 ejb 容器移除有资格的 SFSB 前保持闲置指定时间（以毫秒为单位）。



注意

您仍然可以使用预先存在的 @StatefulTimeout 注释或 stateful-timeout 元素（位于 ejb-jar.xml 部署描述符中）来配置 SFSB 的超时值。但是，设置此类配置会将默认的全局超时值覆盖 SFSB。

有两个方法用于验证您为属性设置的新值：

- 在管理 CLI 中使用 read-attribute 操作。
- 检查服务器配置文件的 ejb3 子系统部分。

其他资源

- 有关查看属性的当前全局超时值的更多信息，请参阅 *管理 CLI 指南中的 显示属性值*。
- 有关为属性更新当前全局超时值的更多信息，请参阅 *管理 CLI 指南中的 更新属性*。

第 6 章 数据源和资源适配器调整

连接池是 JBoss EAP 用来优化使用数据源（如关系数据库或资源适配器）的环境的性能的主要工具。

在时间和系统资源方面，为数据源和资源适配器连接分配和处理资源非常昂贵。连接池通过创建一个可供应用程序可用的连接的 'pool' 来降低连接成本。

在配置连接池以获得最佳性能前，您必须监控负载下的数据源池统计或资源适配器统计，以确定您的环境的适当设置。

6.1. 监控池统计信息

6.1.1. 数据源统计

为数据源启用统计集合后，您可以查看数据源的运行时统计信息。

6.1.1.1. 启用数据源统计

默认情况下，*不启用* 数据源统计信息。您可以使用管理 CLI 或管理控制台启用数据源统计集合。

6.1.1.1.1. 使用管理 CLI 启用数据源统计信息

以下管理 CLI 命令启用 ExampleDS 数据源的统计信息集合。

注意

在受管域中，在此命令前带有 /profile=*PROFILE_NAME*。

```
/subsystem=datasources/data-source=ExampleDS:write-attribute(name=statistics-
enabled,value=true)
```

重新加载服务器以使更改生效。

6.1.1.1.2. 使用管理控制台启用数据源统计

使用以下步骤通过管理控制台为数据源启用统计信息集合。

流程

1. 导航到 **Configuration** → **Subsystems** → **Datasources & Drivers** → **Datasources**。
2. 选择数据源，再单击 **View**。
3. 单击 **Attributes** 选项卡下的 **Edit**。
4. 将 **Statistics Enabled** 字段设置为 **ON**，然后单击 **Save**。此时会出现一个弹出窗口，表示更改需要重新加载才能生效。
5. 重新加载服务器。
   - 对于单机服务器，单击弹出窗口中的 **Reload** 链接，以重新加载服务器。
   - 对于受管域，单击弹出窗口中的 **Topology** 链接。在 **Topology** 选项卡中，选择适当的服务器并选择 **Reload** 下拉菜单选项来重新加载服务器。

6.1.1.2. 查看数据源统计信息

您可以使用管理 CLI 或管理控制台查看数据源的运行时统计信息。

6.1.1.2.1. 使用管理 CLI 查看数据源统计信息

以下管理 CLI 命令检索 ExampleDS 数据源的核心 *池* 统计信息。

注意

在受管域中，在这些命令前面带有 **/host=*HOST_NAME*/server=*SERVER_NAME***。

```
/subsystem=datasources/data-source=ExampleDS/statistics=pool:read-resource(include-
```

```
runtime=true)
{
    "outcome" => "success",
    "result" => {
        "ActiveCount" => 1,
        "AvailableCount" => 20,
        "AverageBlockingTime" => 0L,
        "AverageCreationTime" => 122L,
        "AverageGetTime" => 128L,
        "AveragePoolTime" => 0L,
        "AverageUsageTime" => 0L,
        "BlockingFailureCount" => 0,
        "CreatedCount" => 1,
        "DestroyedCount" => 0,
        "IdleCount" => 1,
        ...
}
```

以下管理 CLI 命令检索 ExampleDS 数据源的 *JDBC* 统计信息。

```
/subsystem=datasources/data-source=ExampleDS/statistics=jdbc:read-resource(include-
runtime=true)
{
    "outcome" => "success",
    "result" => {
        "PreparedStatementCacheAccessCount" => 0L,
        "PreparedStatementCacheAddCount" => 0L,
        "PreparedStatementCacheCurrentSize" => 0,
        "PreparedStatementCacheDeleteCount" => 0L,
        "PreparedStatementCacheHitCount" => 0L,
        "PreparedStatementCacheMissCount" => 0L,
        "statistics-enabled" => true
    }
}
```

注意

由于统计数据是运行时信息，因此请务必指定 include-runtime=true 参数。

6.1.1.2.2. 使用管理控制台查看数据源统计信息

要从管理控制台查看数据源统计信息，请从 Runtime 选项卡中导航到 Datasources 子系统，请选择数据源，再单击 View。

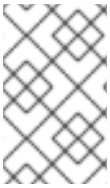
6.1.2. 资源适配器统计

您可以查看已部署资源适配器的核心运行时统计信息。如需了解所有可用 *统计* 的详细列表，请参阅资

源适配器统计附录。

6.1.2.1. 启用资源适配器统计

默认情况下，资源适配器统计 *不会被* 启用。以下管理 CLI 命令启用简单资源适配器 myRA.rar 的统计数据集合，其连接工厂在 JNDI 中绑定为 java:/eis/AcmeConnectionFactory:



注意

在受管域中，在命令中带有 /host=*HOST_NAME*/server=*SERVER_NAME*/。

```
/deployment=myRA.rar/subsystem=resource-adapters/statistics=statistics/connection-
definitions=java\:\/eis\/AcmeConnectionFactory:write-attribute(name=statistics-enabled,value=true)
```

6.1.2.2. 查看资源适配器统计

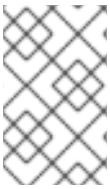
资源适配器统计信息可以从管理 CLI 检索。以下管理 CLI 命令返回资源适配器 myRA.rar 的统计信息，其连接工厂在 JNDI 中绑定为 java:/eis/AcmeConnectionFactory。

注意

在受管域中，在命令中带有 /host=*HOST_NAME*/server=*SERVER_NAME*/。

```
deployment=myRA.rar/subsystem=resource-adapters/statistics=statistics/connection-
definitions=java\:\/eis\/AcmeConnectionFactory:read-resource(include-runtime=true)
{
    "outcome" => "success",
    "result" => {
        "ActiveCount" => "1",
        "AvailableCount" => "20",
        "AverageBlockingTime" => "0",
        "AverageCreationTime" => "0",
        "CreatedCount" => "1",
        "DestroyedCount" => "0",
        "InUseCount" => "0",
        "MaxCreationTime" => "0",
        "MaxUsedCount" => "1",
        "MaxWaitCount" => "0",
        "MaxWaitTime" => "0",
        "TimedOut" => "0",
        "TotalBlockingTime" => "0",
        "TotalCreationTime" => "0"
    }
}
```

-



注意

由于统计数据是运行时信息，因此请务必指定 include-runtime=true 参数。

6.2. 池属性

本节详细介绍了可为最佳数据源或资源适配器性能配置的所选池属性的建议。

最小池大小

min-pool-size 属性定义连接池的最小大小。默认最小值是零连接。使用零 min-pool-size 时，会在第一个事务发生时创建连接并放入池中。

如果 min-pool-size 太小，它会在执行初始数据库命令时出现延迟增加，因为可能需要建立新的连接。如果 min-pool-size 太大，它会产生与数据源或资源适配器的连接。

在不活动期间，连接池将缩小，可能到 min-pool-size 值。

红帽建议将 min-pool-size 设置为允许应用程序按需吞吐量的连接数量。

最大池大小

max-pool-size 属性定义连接池的最大大小。这是一个重要的性能参数，因为它限制了活动连接的数量，因此还限制并发活动的数量。

如果 max-pool-size 太小，可能会导致请求被不必要阻止。如果 max-pool-size 太大，则可能会导致 JBoss EAP 环境、数据源或资源适配器使用比它可以处理的资源更多。

红帽建议将 max-pool-size 设置为至少 15% 大于负载下观察到的可接受 MaxUsedCount。这允许某些缓冲区大于预期条件。

预先填充

pool-prefill 属性指定 JBoss EAP 是否预先填充连接池，并在 JBoss EAP 启动时使用最小连接数。默认值为 false。

当 pool-prefill 设为 true 时，JBoss EAP 会在启动时使用更多资源，但初始事务的延迟会较低。

如果您优化了 min-pool-size，红帽建议将 pool-prefill 设置为 true。

严格最小值

pool-use-strict-min 属性指定 JBoss EAP 是否允许池中连接数量低于指定最小值。

如果 pool-use-strict-min 设为 true，JBoss EAP 将不允许连接数量临时低于指定的最小值。默认值为 false。

虽然指定了最少数量的池连接，但当 JBoss EAP 关闭连接时，如果连接空闲且已达到超时，则暂停可能会导致连接总数在创建新连接并添加到池中前临时低于这个连接。

超时

为连接池配置很多超时选项，但对于性能调优是 idle-timeout-minutes。

idle-timeout-minutes 属性指定在关闭前可以闲置的最大时间（以分钟为单位）。当闲置连接关闭时，池中的连接数将缩减到指定的最小值。

超时越长，会使用更多资源，但请求可能会更快地提供。超时越低，会使用较少的资源，但请求可能需要等待创建新连接。

6.3. 配置池属性

6.3.1. 配置数据源池属性

您可以使用管理 CLI 或管理控制台配置数据源池属性。

先决条件

- 安装 JDBC 驱动程序。

- 创建数据源。

流程

- 要使用管理控制台，请导航到 Configuration → Subsystems → Datasources & Drivers → Datasources，选择数据源，然后点击 View。池选项可在数据源 池选项卡下配置。超时选项可在数据源 超时选项卡下 配置。

- 要使用管理 CLI，请执行以下命令：

```
/subsystem=datasources/data-source=DATASOURCE_NAME/:write-attribute(name=ATTRIBUTE_NAME,value=ATTRIBUTE_VALUE)
```

例如，要将 ExampleDS 数据源 min-pool-size 属性设置为 5 个连接值，请使用以下命令：

```
/subsystem=datasources/data-source=ExampleDS/:write-attribute(name=min-pool-size,value=5)
```

6.3.2. 配置资源适配器池属性

您可以使用管理 CLI 或管理控制台配置资源适配器池属性。

先决条件

- 部署资源适配器并添加连接定义。

流程

- 要使用管理控制台，请导航到 Configuration → Subsystems → Resource Adapters，选择您的资源适配器，点 View，然后在左侧菜单中选择 Connection Definitions。池选项可在 池选项卡下配置。超时选项可在 Attributes 选项卡下配置。

- 要使用管理 CLI，请执行以下命令：

```
/subsystem=resource-adapters/resource-adapter=RESOURCE_ADAPTER_NAME/connection-definitions=CONNECTION_DEFINITION_NAME:write-attribute(name=ATTRIBUTE_NAME,value=ATTRIBUTE_VALUE)
```

例如，要将 my_RA 资源适配器 my_CD 连接定义 min-pool-size 属性设置为 5 个连接值，请使用以下命令：

```
/subsystem=resource-adapters/resource-adapter=my_RA/connection-
definitions=my_CD:write-attribute(name=min-pool-size,value=5)
```

第 7 章 消息传递子系统调整

为 messaging-activemq 子系统中的消息传递服务器启用统计集合后，您可以查看消息传递服务器上资源的运行时统计信息。

7.1. 启用消息传递统计信息

由于它可以对性能造成负面影响，因此 *默认情况下不启用* messaging-activemq 子系统的统计数据集合。您不需要启用队列统计信息来获取基本信息，如队列中的消息数量或添加到队列中的消息数量。这些统计数据可以使用队列属性，而无需将 启用统计数据 设置为 true。

您可以使用管理 CLI 或管理控制台启用额外的统计集合。

7.1.1. 使用管理 CLI 启用消息传递统计信息

以下管理 CLI 命令启用 默认 消息传递服务器的统计信息集合。

```
/subsystem=messaging-activemq/server=default:write-attribute(name=statistics-enabled,value=true)
```

池的连接工厂统计与其他消息传递服务器统计信息分开启用。使用以下命令，为池连接工厂启用统计信息。

```
/subsystem=messaging-activemq/server=default/pooled-connection-factory=activemq-ra:write-attribute(name=statistics-enabled,value=true)
```

重新加载服务器以使更改生效。

7.1.2. 使用管理控制台启用消息传递统计信息

使用以下步骤通过管理控制台为消息传递服务器启用统计信息集合。

流程

1. 导航到 **Configuration → Subsystems → Messaging (ActiveMQ) → Server**。

2. 选择服务器并点 View。

3. 点 Statistics 选项卡下的 Edit。

4. 将 Statistics Enabled 字段设置为 ON，然后单击 Save。

池的连接工厂统计与其他消息传递服务器统计信息分开启用。使用以下步骤为池连接工厂启用统计集合。

1. 导航到 Configuration → Subsystems → Messaging (ActiveMQ) → Server。

2. 选择服务器，选择 Connections，然后单击 View。

3. 选择 Pooled Connection Factory 选项卡。

4. 选择池连接工厂，然后点击 Attributes 选项卡下的 Edit。

5. 将 Statistics Enabled 字段设置为 ON，然后单击 Save。

6. 重新加载服务器以使更改生效。

7.2. 查看消息传递统计信息

您可以使用管理 CLI 或管理控制台查看消息传递服务器的运行时统计信息。

7.2.1. 使用管理 CLI 查看消息传递统计信息

您可以使用以下管理 CLI 命令查看消息传递统计信息：确保包含 include-runtime=true 参数，因为统计是运行时信息。

- 查看队列的统计信息。

```
/subsystem=messaging-activemq/server=default/jms-queue=DLQ:read-resource(include-
runtime=true)
{
    "outcome" => "success",
    "result" => {
        "consumer-count" => 0,
        "dead-letter-address" => "jms.queue.DLQ",
        "delivering-count" => 0,
        "durable" => true,
        ...
    }
}
```

- 查看主题的统计信息。

```
/subsystem=messaging-activemq/server=default/jms-topic=testTopic:read-resource(include-
runtime=true)
{
    "outcome" => "success",
    "result" => {
        "delivering-count" => 0,
        "durable-message-count" => 0,
        "durable-subscription-count" => 0,
        ...
    }
}
```

- 查看池连接工厂的统计信息。

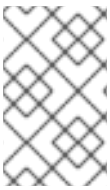
```
/subsystem=messaging-activemq/server=default/pooled-connection-factory=activemq-
ra/statistics=pool:read-resource(include-runtime=true)
{
    "outcome" => "success",
    "result" => {
        "ActiveCount" => 1,
        "AvailableCount" => 20,
        "AverageBlockingTime" => 0L,
        "AverageCreationTime" => 13L,
        "AverageGetTime" => 14L,
        ...
    }
}
```

注意

池的连接工厂统计与其他消息传递服务器统计信息分开启用。

7.2.2. 使用管理控制台查看消息传递统计信息

若要从管理控制台查看消息传递统计信息，可从 Runtime 选项卡导航到 Messaging (ActiveMQ) 子系统，然后选择服务器。选择一个目的地来查看其统计信息。



注意

Prepared Transactions 页面是您可以查看、提交和回滚准备的事务。

7.3. 配置消息计数器

您可以为消息传递服务器配置以下消息计数器属性。

- message-counter-max-day-history ：消息计数器历史记录的天数。
- message-counter-sample-period ：队列被抽样的频率（以毫秒为单位）。用于配置这些选项的管理 CLI 命令使用以下语法：务必将 *'STATISTICS_NAME'* 和 *'STATISTICS_VALUE'* 替换为您要配置的统计名称和值。

```
/subsystem=messaging-activemq/server=default::write-
attribute(name=STATISTICS_NAME,value=STATISTICS_VALUE)
```

例如，使用以下命令将 message-counter-max-day-history 设置为五天，并将 message-counter-sample-period 设置为 2 秒。

```
/subsystem=messaging-activemq/server=default:write-attribute(name=message-counter-max-day-
history,value=5)
/subsystem=messaging-activemq/server=default:write-attribute(name=message-counter-sample-
period,value=2000)
```

7.4. 查看队列的消息计数器和历史记录

您可以使用以下管理 CLI 操作来查看队列的消息计数器和消息计数器历史记录。

- **list-message-counter-as-json**
- **list-message-counter-as-html**
- **list-message-counter-history-as-json**
- **list-message-counter-history-as-html**

使用显示这些值的管理 CLI 命令使用以下语法：务必将 *'QUEUE_NAME'* 和 *'OPERATION_NAME'* 替换为您要使用的队列名称和操作。

```
/subsystem=messaging-activemq/server=default/jms-queue=QUEUE_NAME:OPERATION_NAME
```

例如，使用以下命令以 JSON 格式查看 TestQueue 队列的消息计数器：

```
/subsystem=messaging-activemq/server=default/jms-queue=TestQueue:list-message-counter-as-
json
{
    "outcome" => "success",
    "result" => "
{\"destinationName\":\"TestQueue\",\"destinationSubscription\":null,\"destinationDurable\":true,\"count\":
0,\"countDelta\":0,\"messageCount\":0,\"messageCountDelta\":0,\"lastAddTimestamp\":\"12/31/69
7:00:00 PM\",\"updateTimestamp\":\"2/20/18 2:24:05 PM\"}"
}
```

7.5. 为队列重置消息计数器

您可以使用 reset-message-counter 管理 CLI 操作重置队列的消息计数器。

```
/subsystem=messaging-activemq/server=default/jms-queue=TestQueue:reset-message-counter
{
    "outcome" => "success",
    "result" => undefined
}
```

7.6. 使用管理控制台的运行时操作

使用管理控制台，您可以：

- 执行强制故障转移到另一个消息传递服务器
- 重置消息传递服务器的所有消息计数器
- 重置消息传递服务器的所有消息计数器历史记录
- 查看与消息传递服务器相关的信息
- 关闭消息传递服务器的连接
- 回滚事务
- 提交事务

7.6.1. 执行强制故障转移到另一个消息传递服务器

您可以使用管理控制台执行强制故障转移到另一个消息传递服务器。

流程

1. 使用以下任一方法访问管理控制台并导航到 Server ：
   - Runtime→ Browse By → Hosts → Host → Server
   - Runtime→ Browse By → Server Groups → Server Group → Server
2. 点 Messaging ActiveMQ → Server

3. 单击 View 旁边的箭头按钮，然后单击 Force Failover。

4. 在 Force Failover 窗口中，单击 Yes。

7.6.2. 为消息传递服务器重置所有消息计数器

您可以使用管理控制台重置消息传递服务器的所有消息计数器。

流程

1. 使用以下任一方法访问管理控制台并导航到 Server ：

   - Runtime→ Browse By → Hosts → Host → Server
   - Runtime→ Browse By → Server Groups → Server Group → Server

2. 点 Messaging ActiveMQ → Server

3. 单击 View 旁边的箭头按钮，然后单击 重置。

4. 在 Reset 窗口中，点 Reset all message 计数器 旁的切换按钮来启用该功能。

   按钮现在以蓝色背景显示 ON。

5. 单击 Reset。

7.6.3. 为消息传递服务器重置消息计数器历史记录

您可以使用管理控制台重置消息传递服务器的消息计数器历史记录。

流程

1. 使用以下任一方法访问管理控制台并导航到 Server ：

   - Runtime→ Browse By → Hosts → Host → Server
   - Runtime→ Browse By → Server Groups → Server Group → Server

2. 点 Messaging ActiveMQ → Server

3. 单击 View 旁边的箭头按钮，然后单击 重置。

4. 在 Reset 窗口中，点 Reset all message 计数器历史记录 旁边的切换按钮来启用该功能。

   按钮现在以蓝色背景显示 ON。

5. 单击 Reset。

7.6.4. 查看与消息传递服务器相关的信息

使用管理控制台，您可以查看与消息传递服务器相关的信息列表：

- 连接
- 消费者
- 制作者
- 连接器
- 角色

- Transactions

查看与消息传递服务器相关的信息：

流程

1. 使用以下任一方法访问管理控制台并导航到 Server ：
   - Runtime→ Browse By → Hosts → Host → Server
   - Runtime→ Browse By → Server Groups → Server Group → Server
2. 单击 Messaging ActiveMQ → Server，然后单击 View。
3. 单击导航窗格上的相应项目，以查看右侧窗格中的项目列表。

7.6.5. 关闭消息传递服务器的连接

您可以通过提供 IP 地址、 ActiveMQ 地址匹配或用户名来关闭连接。

关闭消息传递服务器的连接：

流程

1. 使用以下任一方法访问管理控制台并导航到 Server ：
   - Runtime→ Browse By → Hosts → Host → Server
   - Runtime→ Browse By → Server Groups → Server Group → Server

2. 单击 **Messaging ActiveMQ → Server，然后单击 View。**

3. **在导航窗格中，**单击 **Connections。**

4. **在 Close 窗口中，点击您要关闭的连接的适当**标签页。

5. **根据您的**选择，输入 **IP 地址、ActiveMQ 地址匹配或用户名，然后**单击 **Close。**

7.6.6. 消息传递服务器的回滚事务

您可以使用管理控制台为**消息**传递**服务器回滚事务。**

流程

1. **使用以下任一方法**访问**管理控制台并导航到 Server ：**

   - **Runtime→ Browse By → Hosts → Host → Server**
   - **Runtime→ Browse By → Server Groups → Server Group → Server**

2. 单击 **Messaging ActiveMQ → Server，然后单击 View。**

3. **在导航窗格中，**单击 **Transactions。**

4. 选择**您要回滚的事务，然后点 Rollback。**

7.6.7. 为消息传递服务器提交事务

您可以使用管理控制台为**消息**传递**服务器提交事务。**

流程

1. 使用以下任一方法访问管理控制台并导航到 Server：

   - Runtime→ Browse By → Hosts → Host → Server
   - Runtime→ Browse By → Server Groups → Server Group → Server

2. 单击 Messaging ActiveMQ → Server，然后单击 View。

3. 在导航窗格中，单击 Transactions。

4. 选择您要提交的事务，然后单击 Commit。

7.7. 调整 JAKARTA 消息传递

如果使用 Jakarta Messaging API，请查看以下信息，以了解有关如何提高性能的提示。

- 禁用消息 ID。

  如果您不需要消息 ID，请在 MessageProducer 类上使用 setDisableMessageID （） 方法禁用它们。将值设为 true 可消除创建唯一 ID 的开销，并减少消息的大小。

- 禁用消息时间戳。

  如果您不需要消息时间戳，请在 MessageProducer 类上使用 setDisableMessageTimeStamp （） 方法禁用它们。将值设为 true 可消除创建时间戳的开销，并减少消息的大小。

- 避免使用 ObjectMessage。

  ObjectMessage 用于发送包含序列化对象（即消息的正文或有效负载）的消息，作为字节数流通过线发送。Java 序列化形式甚至小对象非常大，并占用线上大量空间。与自定义划分技术相比，它也会较慢。只有在您无法使用其他消息类型之一时，才使用 ObjectMessage，例如，如果您不知道有效负载的类型，直到运行时为止。

- 避免 AUTO_ACKNOWLEDGE.

  选择使用消费者的确认模式会影响性能，因为通过发送通过网络发送的确认消息会增加开销和流量。AUTO_ACKNOWLEDGE 会导致这个开销，因为它需要为客户端上收到的每个消息从服务器发送一个确认。如果您可以使用 DUPS_OK_ACKNOWLEDGE，它以延迟方式确认消息 CLIENT_ACKNOWLEDGE，这意味着客户端代码将调用一个方法来确认消息，或使用一个确认或提交的一个确认或提交。

- 避免持久消息。

  默认情况下，Jakarta Messaging 消息是 durable。如果您不需要持久消息，请将它们设置为 非持久消息。持久化消息会导致大量开销，因为它们被保留到存储中。

- 使用 TRANSACTED_SESSION 模式在单个事务中发送和接收消息。

  通过在单一事务中批处理消息，JBoss EAP 中集成的 ActiveMQ Artemis 服务器只需要一个网络往返提交，而不是在每次发送或接收时。

7.8. 调优持久性

- 将消息日志放在其自己的物理卷上。

  仅附加日志的一个优点是磁盘头移动被最小化。如果磁盘共享，则此优点将会丢失。当多个进程（如事务协调器、数据库和其他日志）时，从同一磁盘读取和写入，性能会受到影响，因为磁盘头必须在不同文件之间跳过。如果您使用分页或大型消息，请确保它们也放在单独的卷中。

- 调优 journal-min-files 值。

  将 journal-min-files 参数设置为适合您平均可持续增长率的文件数量。如果您经常看到在日志数据目录上创建的新文件，这意味着大量数据会被保留，您需要增加最少的文件数。这允许日志重复使用，而不是创建新的数据文件。

- 优化日志文件大小。

  日志文件大小必须与磁盘上 cylinder 的容量一致。默认值 10MB 在大多数系统上应该足够了。

- 使用 AIO 日志类型。

  对于 Linux 操作系统，请将您的日志类型保留为 AIO。AIO 比 Java NIO 更好地扩展。

- 调整 journal-buffer-timeout 值。

  增加 journal-buffer-timeout 值会导致延迟增加吞吐量。

- 调整 journal-max-io 值。

  如果您使用 AIO，可以通过增加 journal-max-io 参数值来提高性能。如果您使用 NIO，请不要更改这个值。

7.9. 其他调整选项

本节介绍 JBoss EAP 消息传递中可以调整的其他位置。

- 使用异步发送确认。

  如果您需要发送非事务处理的消息，且不需要保证在调用 send （） 返回时已达到服务器，请不要将它们设置为发送阻止。反之，使用异步发送确认项在单独的流中返回您的发送确认。但是，如果服务器崩溃，一些信息可能会丢失。

- 使用 pre-acknowledge 模式。

  使用 pre-acknowledge 模式时，消息会在发送到客户端之前被确认。这可减少有线路上的确认流量量。但是，如果客户端崩溃，如果客户端重新连接，则不会更新消息。

- 禁用安全性。

  通过将 security-enabled 属性设置为 false，在禁用安全性时，性能会很小。

- 禁用持久性。

您可以通过将 persistence-enabled 设置为 false 来完全关闭消息持久性。

- 同步事务不正常。

将 journal-sync-transactional 设置为 false 可提供更好的事务持久性性能，但牺牲在失败时丢失事务的可能性。

- 同步非事务处理。

将 journal-sync-non-transactional 设置为 false 可提供更好的非事务处理性能，以牺牲失败时丢失的持久消息。

- 发送消息非阻塞。

为了避免在发送的每个消息中等待网络往返，如果您使用 Jakarta Messaging 和 JNDI，将 block-on-durable-send 和 block-on-non-durable-send 设置为 false，或通过调用 setBlockOnDurableSend （） 和 setBlockOnNonDurableSend （） 方法直接在 ServerLocator 中设置。

- 优化 consumer-window-size。

如果您有非常快速的用户，您可以提高 consumer-window-size 来有效地禁用消费者流控制。

- 使用核心 API 而不是 Jakarta Messaging API。

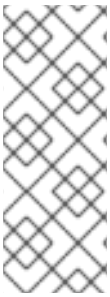
Jakarta 消息传递操作必须在服务器处理它们之前转换为核心操作，从而比使用核心 API 的性能较低。在使用核心 API 时，尽量使用采用 SimpleString 的方法。与 java.lang.String 不同，在将简单字符串写入有线之前不需要复制，因此如果您在调用之间重复使用 SimpleString 实例，您可以避免一些不必要的复制。请注意，核心 API 无法移植到其他代理。

7.10. 避免反模式

- 尽可能重复使用连接、会话、使用者和制作者。

最常见的消息传递反模式是为每个发送或消耗的消息创建新连接、会话和制作者。这些对象需

要创建时间，并涉及多个网络往返，因此使用资源不佳。始终重复使用它们。



注意

Spring Messaging 模板等一些流行的库使用这些反模式。如果您使用 Spring Messaging 模板，则性能可能不佳。Spring Messaging Template 只能在缓存 Jakarta 消息传递会话的应用程序服务器上安全使用，例如使用 Jakarta Connectors，然后仅发送消息。即使在应用服务器中，它也无法安全地用于同步使用的消息。

- 避免出现 fat 消息。

  详细格式，如 XML 在线上占用大量空间，因此性能会受到影响。如果可以，请避免消息正文中的 XML。

- 不要为每个请求创建临时队列。

  这种常见的反模式涉及临时队列请求响应模式。使用临时队列请求响应模式时，消息将发送到目标，并使用本地临时队列的地址设置回复标头。当接收者收到该消息时，会处理该消息，然后发回响应到回复标头中指定的地址。此模式出现的常见错误是在每个发送的消息上创建一个新的临时队列，从而大大降低性能。相反，应将临时队列重新用于许多请求。

- 除非是必需的，否则不要使用消息驱动的 Bean。

  使用 MDBs 使用消息比使用简单的 Jakarta 消息传递消息使用者要慢。

第 8 章 LOGGING 子系统调整

您可以通过禁用日志记录到控制台、配置适当的日志记录级别并指定存储日志文件的最佳位置，进一步提高 JBoss EAP 日志子系统在生产环境中的性能。

8.1. 禁用日志记录到控制台

禁用控制台日志记录可以提高 JBoss EAP 性能。虽然在开发和测试环境中输出日志到控制台会很有用，但对于生产环境，在大多数情况下不需要这样做。JBoss EAP 根日志记录器包括所有默认单机服务器配置文件的控制台日志处理程序，但 standalone-full-ha 除外。默认受管域配置文件不包含控制台处理程序。

若要从根日志记录器移除默认控制台处理程序，可使用以下管理 CLI 命令：

```
/subsystem=logging/root-logger=ROOT:remove-handler(name=CONSOLE)
```

8.2. 配置日志记录级别

为了获得最佳性能，您必须为生产环境相应地配置日志级别。例如，虽然 INFO 或 DEBUG 级别可能适合开发或测试环境，但在大多数情况下，您应该将生产环境日志级别设置为更高，如 WARN 或 ERROR。

8.3. 配置日志文件的位置

您应该将日志文件的存储位置视为潜在的性能问题。如果您将日志保存到具有 I/O 吞吐量的文件系统或磁盘配置中，则可能会影响整个平台的性能。

为防止日志记录影响 JBoss EAP 性能，建议您将日志位置设置为具有大量空间的高性能专用磁盘。

第 9 章 UNDERTOW 子系统调整

与 JBoss EAP 6 中之前的 Web 子系统相比，JBoss EAP 7 中引入的非阻塞 I/O (NIO) undertow 子系统显著提高了性能。为您的环境调整 undertow 子系统的机会包括：

- 缓冲缓存配置
- 字节缓冲池配置
- Jakarta Server Pages 配置选项
- 侦听器配置选项
- 会话属性 marshalling

9.1. 缓冲缓存配置

缓冲区缓存存储由 undertow 子系统处理的静态文件。这包括镜像、静态 HTML、CSS 和 JavaScript 文件。您可以为每个 Undertow servlet 容器指定默认缓冲区缓存。为您的 servlet 容器拥有优化的缓冲区缓存可以提高 Undertow 性能来提供静态文件。

缓冲区缓存中的缓冲区在区域内分配，并且是固定大小。每个缓冲区缓存有三个可配置的属性：

buffer-size

单个缓冲区的大小，以字节为单位。默认值为 1024 字节。将缓冲区大小设置为完全存储您的最大静态文件。

buffer-per-region

每个区域的缓冲区数量。默认值为 1024。

max-regions

最大区域数量，用于设置分配给缓冲区缓存的最大内存量。默认值为 10 个区域。

您可以通过乘以缓冲区大小、每个区域的缓冲区数量以及最大区域数来计算缓冲区缓存使用的最大内存量。例如，默认缓冲区缓存为 1024 字节，每个区域 * 10 个区域 = 10 MB。

根据您的静态文件的大小配置缓冲区缓存，以及测试开发环境中预期负载的结果。在确定对性能的影响时，请考虑缓冲区缓存性能的平衡与所使用的内存的好处。

9.2. 字节缓冲池配置

Undertow 字节缓冲区池用于分配池 NIO ByteBuffer 实例。所有监听器都有一个字节缓冲区池，您可以为每个监听程序使用不同的缓冲区池和 worker。可以在不同的服务器实例之间共享字节缓冲区池。

严重影响性能的主要字节缓冲区池属性是 buffer-size。默认值是基于系统的 RAM 资源计算的，在大多数情形中都足够了。如果您要手动配置此属性，则大多数服务器的理想大小为 16 KB。

9.3. JAKARTA SERVER PAGES 配置选项

Undertow servlet 容器的以下 Jakarta Server Pages 配置选项为如何将 Jakarta Server Pages 编译成 Java 字节码提供优化：

generate-strings-as-char-arrays

如果您的 Jakarta 服务器页面包含许多 String 常数，请通过将 String 常量转换为 char 数组来优化 scriptlet。

optimize-scriptlets

如果您的 Jakarta Server Pages 包含多个 String 串联，则此选项通过删除每个 Jakarta Server Pages 请求的 String 串联来优化 scriptlets。

trim-spaces

如果您的 Jakarta 服务器页面包含大量空格，启用这个选项会从 HTTP 请求中修剪空格，并减少 HTTP 请求有效负载。

9.3.1. 使用管理控制台启用 Jakarta Server Pages 选项

要使用管理控制台启用 Undertow Jakarta Server Pages 配置选项，请完成以下步骤：

流程

1. 导航到 Configuration → Subsystems → Web (Undertow) → Servlet Container。
2. 选择您要配置的 servlet 容器，然后单击 View。
3. 选择 Jakarta Server Pages 并点 Edit。
4. 对于您要启用的每个选项，请将字段设置为 ON，然后单击 Save。

9.3.2. 使用管理 CLI 启用 Jakarta Server Pages 选项

要使用管理 CLI 启用 Undertow Jakarta Server Pages 配置选项，请完成以下步骤：

流程

- 使用以下命令：

```
/subsystem=undertow/servlet-container=__SERVLET_CONTAINER__/setting=jsp/:write-attribute(name=__OPTION_NAME__,value=true)
```

例如，要为 默认 servlet 容器启用 generate-strings-as-char-arrays，请使用以下命令：

```
/subsystem=undertow/servlet-container=default/setting=jsp/:write-attribute(name=generate-strings-as-char-arrays,value=true)
```

9.4. 侦听器配置选项

根据您的应用程序和环境，您可以配置特定于某些类型的流量的多个监听程序，如特定端口上的流量，然后为每个监听程序配置选项。

以下是可在 HTTP、HTTPS 和 AJP 侦听器上配置的性能相关选项。

max-connections

侦听器可以处理的并发连接的最大数量。默认情况下，此属性未定义，这会导致连接无限。

您可以使用此选项设置侦听器可以处理的连接数量，这可能对大写资源使用量很有用。在配置这个值时，您应该考虑您的工作负载和流量类型。另请参阅以下 no-request-timeout。

no-request-timeout

连接在关闭前闲置的时间长度（以毫秒为单位）。默认值为 60000 毫秒(1 分钟)。

在您的环境中调优此选项以获得最佳连接效率有助于提高网络性能。如果闲置连接被预先关闭，则重新建立连接时会存在开销。如果闲置连接过长，它们不会必要地使用资源。

max-header-size

HTTP 请求标头的最大大小，以字节为单位。默认值为 1048576 (1024KB)。

限制标头大小有助于防止某些类型拒绝服务攻击。

buffer-pool

指定 io 子系统中用于侦听器的缓冲池。默认情况下，所有监听程序都使用默认缓冲区池。

您可以使用这个选项将每个监听程序配置为使用唯一的缓冲区池，或者有多个监听程序使用相同的缓冲池。

worker

undertow 子系统依赖于 io 子系统来提供 XNIO 工作程序。这个选项指定监听器使用的 XNIO worker。默认情况下，侦听器使用 io 子系统中的 默认 worker。

将每个监听程序配置为使用特定的 worker 可能会有帮助，以便您可以将不同的 worker 资源分配给某些类型的网络流量。

9.4.1. 使用管理控制台配置监听程序选项

要使用管理控制台配置监听程序选项，请完成以下步骤：

流程

1. 导航到 Configuration → Subsystems → Web (Undertow) → Server。

2. 选择您要配置的服务器，然后点 View。

3. 在左侧菜单中选择 Listener，然后选择要配置的监听程序类型，如 HTTP Listener，然后在表中选择监听程序。

4. 点 Edit，修改您要配置的选项，然后点 Save。

9.4.2. 使用管理 CLI 配置监听程序选项

要使用管理 CLI 配置监听程序选项，请完成以下步骤：

流程

- 使用以下命令：

```
/subsystem=undertow/server=SERVER_NAME/LISTENER_TYPE=LISTENER_NAME:write-attribute(name=OPTION_NAME,value=OPTION_VALUE)
```

例如，要将 default - server Undertow 服务器中的默认 HTTP 侦听器的 max- connections 设置为 100000，请使用以下命令：

```
/subsystem=undertow/server=default-server/http-listener=default:write-attribute(name=max-connections,value=100000)
```

第 10 章 IO 子系统调整

io 子系统定义供其他 JBoss EAP 子系统使用的 XNIO 工作程序和缓冲区池，如 Undertow 和 Remoting。

10.1. 配置 WORKER

您可以创建多个单独的 worker，每个 worker 都有自己的性能配置，以及处理不同的 I/O 任务。例如，您可以创建一个 worker 来处理 HTTP I/O，另一个 worker 来处理 Jakarta Enterprise Beans I/O，然后单独配置每个 worker 的属性来满足特定的负载要求。

严重影响性能的 worker 属性包括 io-threads，它设置 worker 可以使用的 I/O 线程 总数，以及设置可用于特定任务的最大线程数。这两个属性的默认值根据服务器的 CPU 数量计算。

10.1.1. 监控 worker 统计

您可以使用管理 CLI 查看 worker 的运行时统计信息。这会公开 worker 统计信息，如连接计数、线程计数和队列大小。

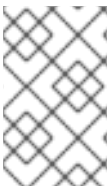
以下命令显示 默认 worker 的运行时统计信息：

```
/subsystem=io/worker=default:read-resource(include-runtime=true,recursive=true)
```

注意

由 core-pool-size 统计跟踪的核心线程数量目前始终设置为与最大线程数相同的值，由 max-pool-size 统计跟踪。

10.2. 配置缓冲池



注意

IO 缓冲池已弃用，但它们仍设置为当前版本中的默认设置。

io 子系统中的缓冲池是一个池化的 NIO 缓冲实例，用于 I/O 操作。与 worker 一样，您可以创建单独的缓冲区池，这些池可以专用于处理特定的 I/O 任务。

严重影响性能的主要缓冲区池属性是 buffer-size。默认值是基于系统的 RAM 资源计算的，在大多数情形中都足够了。如果您要手动配置此属性，则大多数服务器的理想大小为 16KB。

第 11 章 JGROUPS 子系统调整

为了获得最佳性能，建议您在支持它的环境中对 JGroups 使用 UDP 多播。

注意

TCP 的开销更大，通常被视为比 UDP 慢，因为它处理错误检查、数据包排序和拥塞控制本身。JGroups 处理 UDP 的这些功能，而 TCP 则保证它们本身。在将 JGroups 用于不可靠或高拥塞网络或多播不可用时，TCP 是一个不错的选择。

本章假设您已选择了 JGroups 堆栈传输协议(UDP 或 TCP)以及 JGroups 集群通信将使用的通信协议。

11.1. 监控 JGROUPS 统计信息

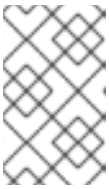
您可以使用管理 CLI 或 JMX 启用 jgroups 子系统的统计信息，以监控 JBoss EAP 集群。

注意

启用统计信息会对性能造成负面影响。仅在需要时启用统计信息。

流程

1. 使用以下命令，启用 JGroups 通道的统计信息。



注意

在受管域中，在这些命令之前使用 */profile=PROFILE_NAME*。

```
/subsystem=jgroups/channel=CHANNEL_NAME:write-attribute(name=statistics-
enabled,value=true)
```

例如，使用以下命令为默认 ee 频道启用统计信息。

```
/subsystem=jgroups/channel=ee:write-attribute(name=statistics-enabled,value=true)
```

2. 重新加载 JBoss EAP 服务器。

```
reload
```

3. 现在，您可以使用管理 CLI 或使用 JVM 监控工具通过 JMX 来查看 JGroups 统计信息：

- 要使用管理 CLI，请在您要查看统计信息的 JGroups 频道或协议上使用 :read-resource (include-runtime=true) 命令。

注意

在受管域中，在这些命令前面带有 */host=HOST_NAME/server=SERVER_NAME*。

例如：

- 要查看 ee 频道的统计信息，请使用以下命令：

```
/subsystem=jgroups/channel=ee:read-resource(include-runtime=true)
```

- 要查看 ee 频道中 FD_ALL 协议的统计信息，请使用以下命令：

```
/subsystem=jgroups/channel=ee/protocol=FD_ALL:read-resource(include-runtime=true)
```

- 要使用 JVM 监控工具连接 JBoss EAP，请参阅监控性能 章节。您可以通过 JMX 连接查看 JGroups MBeans 的统计信息。

11.2. 网络和巨型帧

在可能的情况下，建议 JGroups 流量的网络接口应该是专用虚拟局域网(VLAN)的一部分。这样，您可以将集群通信与其他 JBoss EAP 网络流量分离，以更轻松地控制集群网络性能、吞吐量和安全性。

要提高集群性能的另一个网络配置是启用巨型帧。如果您的网络环境支持，通过增加最大传输单元(MTU)来启用巨型帧有助于提高网络性能，特别是在高吞吐量环境中。

要使用巨型帧，您的网络中的所有 NIC 和交换机都必须支持它。

其他资源

有关为 Red Hat Enterprise Linux 启用巨型帧的说明，请参阅红帽客户门户网站。

11.3. 消息绑定

JGroups 中的消息通过组合多个小消息到更大的捆绑包来提高网络性能。相反，消息不会通过网络发送到集群节点，而是会排队，直到达到最大捆绑包大小或没有更多消息发送。排队的消息被编译到更大的消息捆绑包中，然后发送。

这种绑定减少了通信开销，特别是在 TCP 环境中，对于网络通信开销较高。

11.3.1. 配置消息捆绑

JGroups 消息 bundling 使用 max_bundle_size 属性进行配置。默认的 max_bundle_size 是 64KB。

调整捆绑包大小的性能改进取决于您的环境，以及在捆绑被编译时，网络流量是否平衡到可能的通讯延迟。

流程

- 使用以下管理 CLI 命令配置 max_bundle_size。

```
/subsystem=jgroups/stack=STACK_NAME/transport=TRANSPORT_TYPE/property=max_bundle_size:add(value=BUNDLE_SIZE)
```

例如，将默认 udp 堆栈的 max_bundle_size 设置为 60K ：

```
/subsystem=jgroups/stack=udp/transport=UDP/property=max_bundle_size:add(value=60K)
```

11.4. JGROUPS 线程池

jgroups 子系统使用自己的线程池来处理集群通信。JGroups 包含用于 默认、内部、oob 和 计时器 功

能的线程池，您可以单独配置。每个 JGroups 线程池包括 keepalive-time、max-threads、min-threads 和 queue-length 的可配置属性。

每个线程池属性的适当值取决于您的环境，但在某些情况下，默认值应该会挂起。

11.5. JGROUPS 发送和接收缓冲区

jgroups 子系统为 UDP 和 TCP 堆栈具有可配置的发送和接收缓冲区。

JGroups 缓冲区的适当值取决于您的环境，但在某些情况下，默认值应该会挂起。建议您在开发环境中的负载下测试集群，以调整缓冲区大小的适当值。

注意

您的操作系统可能会限制可用的缓冲区大小，JBoss EAP 可能无法使用其配置的缓冲区值。

第 12 章 事务子系统调整

如果您的环境使用 XA 分布式事务，您可以调整事务管理器的日志存储以提高性能。

默认事务日志存储使用一个简单的文件存储。对于此类日志存储的 XA 事务可能效率低下，因为它为每个事务日志创建一个系统文件。特别是对于 XA 事务而言，日志存储更高效，因为它使用由一个文件组成的日志来所有事务。

为获得更高的 XA 事务性能，建议您使用日志存储。对于 Red Hat Enterprise Linux 系统，您还可以在日志存储中启用异步 I/O (AIO)以进一步提高性能。

注意

对于 Red Hat Enterprise Linux 系统，如果您在日志存储中启用异步 I/O (AIO)，请确保安装了 libaio 软件包。

12.1. 使用管理控制台启用日志存储

您可以使用管理控制台启用日志存储。

流程

1. 导航到 Configuration → Subsystems → Transaction →，然后点击 View。
2. 在 Configuration 选项卡中，单击 Edit。
3. 将 Use Journal Store 字段设置为 ON。
4. 可选 ： 对于 Red Hat Enterprise Linux 系统，将 Journal Store Enable Async IO 字段设置为 ON。
5. 点击 Save。

12.2. 使用管理 CLI 启用日志存储

您可以使用管理 CLI 启用日志存储。

流程

1. **要使用管理 CLI 启用日志存储，请使用以下命令：**

```
/subsystem=transactions:write-attribute(name=use-journal-store,value=true)
```

2. **可选 ： 对于 Red Hat Enterprise Linux 系统，使用以下命令启用日志存储异步 I/O：**

```
/subsystem=transactions:write-attribute(name=journal-store-enable-async-io, value=true)
```

附录 A. 参考资料

A.1. 数据源统计

表 A.1. 核心池统计数据

Name	描述
ActiveCount	活跃连接的数量。每个连接都由一个应用程序使用，或者在池中可用。
AvailableCount	池中可用连接的数量。
AverageBlockingTime	阻止在池中获取专用锁定的平均时间。这个值以毫秒为单位。
AverageCreationTime	创建连接的平均时间。这个值以毫秒为单位。
AverageGetTime	获取连接的平均时间。
AveragePoolTime	连接在池中花费的平均时间。
AverageUsageTime	使用连接的平均时间。
BlockingFailureCount	尝试获取连接的失败次数。
CreatedCount	创建的连接数。
DestroyedCount	销毁的连接数量。
IdleCount	当前闲置的连接数量。
InUseCount	当前正在使用的连接数量。
MaxCreationTime	创建连接所需的最长时间。这个值以毫秒为单位。
MaxGetTime	获取连接的最长时间。
MaxPoolTime	池中连接的最大时间。
MaxUsageTime	使用连接的最大时间。
MaxUsedCount	使用的最大连接数。
MaxWaitCount	同时等待连接的最大请求数。
MaxWaitTime	等待池中专用锁定所花费的最长时间。

Name	描述
timedout	超时连接数量。
TotalBlockingTime	等待池中专用锁定的总时间。这个值以毫秒为单位。
TotalCreationTime	创建连接的总时间。这个值以毫秒为单位。
TotalGetTime	获取连接的总时间。
TotalPoolTime	池中连接所花费的总时间。
TotalUsageTime	使用连接的总时间。
WaitCount	必须等待获取连接的请求数。
XACommitAverageTime	XAResource 提交调用的平均时间。
XACommitCount	XAResource 提交调用的数量。
XACommitMaxTime	XAResource 提交调用的最长时间。
XACommitTotalTime	所有 XAResource 提交调用的总时间。
XAEndAverageTime	XAResource 结束调用的平均时间。
XAEndCount	XAResource 端到端调用的数量。
XAEndMaxTime	XAResource 端到端调用的最长时间。
XAEndTotalTime	所有 XAResource 结束调用的总时间。
XAForgetAverageTime	XAResource forget 调用的平均时间。
XAForgetCount	XAResource forget 调用的数量。
XAForgetMaxTime	XAResource forget 调用的最长时间。
XAForgetTotalTime	所有 XAResource forget 调用的总时间。
XAPrepareAverageTime	XAResource 准备调用的平均时间。
XAPrepareCount	XAResource 准备调用的数量。
XAPrepareMaxTime	XAResource 准备调用的最长时间。
XAPrepareTotalTime	所有 XAResource 准备调用的总时间。

Name	描述
XARecoverAverageTime	XAResource 恢复调用的平均时间。
XARecoverCount	XAResource 恢复调用的数量。
XARecoverMaxTime	XAResource 恢复调用的最长时间。
XARecoverTotalTime	所有 XAResource 恢复调用的总时间。
XARollbackAverageTime	XAResource 回滚调用的平均时间。
XARollbackCount	XAResource 回滚调用的数量。
XARollbackMaxTime	XAResource 回滚调用的最长时间。
XARollbackTotalTime	所有 XAResource 回滚调用的总时间。
XAStartAverageTime	XAResource 开始调用的平均时间。
XAStartCount	XAResource 开始调用的数量。
XAStartMaxTime	XAResource 开始调用的最长时间。
XAStartTotalTime	所有 XAResource 开始调用的总时间。

表 A.2. JDBC 统计

Name	描述
PreparedStatementCacheAccessCount	访问声明缓存的次数。
PreparedStatementCacheAddCount	添加到声明缓存的声明数。
PreparedStatementCacheCurrentSize	当前在声明缓存中缓存的准备和可调用语句的数量。
PreparedStatementCacheDeleteCount	从缓存中丢弃的声明数。
PreparedStatementCacheHitCount	从缓存中使用该语句的次数。
PreparedStatementCacheMissCount	声明对缓存中声明无法满足的次数。

A.2. 资源适配器统计

表 A.3. 资源适配器统计

Name	描述
ActiveCount	活跃连接的数量。每个连接都由一个应用程序使用，或者在池中可用
AvailableCount	池中可用连接的数量。
AverageBlockingTime	阻止在池中获取专用锁定的平均时间。该值以毫秒为单位。
AverageCreationTime	创建连接的平均时间。该值以毫秒为单位。
CreatedCount	创建的连接数。
DestroyedCount	销毁的连接数量。
InUseCount	当前正在使用的连接数量。
MaxCreationTime	创建连接所需的最长时间。该值以毫秒为单位。
MaxUsedCount	使用的最大连接数。
MaxWaitCount	同时等待连接的最大请求数。
MaxWaitTime	等待池中专用锁定所花费的最长时间。
timedout	超时连接数量。
TotalBlockingTime	等待池中专用锁定的总时间。该值以毫秒为单位。
TotalCreationTime	创建连接的总时间。该值以毫秒为单位。
WaitCount	必须等待连接的请求数。

A.3. IO 子系统属性

注意

这些表中的属性名称会在管理模型中出现时列出，例如使用管理 CLI 时。请参阅位于 ***EAP_HOME*/docs/schema/wildfly-io_2_0.xsd** 的架构定义文件，以查看 XML 中出现的元素，因为管理模型可能会有所不同。

表 A.4. worker 属性

属性	Default（默认）	描述
io-threads		为 worker 创建的 I/O 线程数量。如果没有指定，则线程数量被设置为 CPU hieradata 2 的数量。
stack-size	0	用于 worker 线程的堆栈大小（以字节为单位）。
task-keepalive	60000	保持非核心任务线程的毫秒数。
task-core-threads	2	核心任务线程池的线程数量。
task-max-threads		worker 任务线程池的最大线程数量。如果没有指定，则最大线程数设置为 CPU Thycotic 16 的数量，如果设置，则使用 **MaxFileDescriptorCount** JMX 属性。

表 A.5. buffer-pool 属性

属性	Default（默认）	描述
buffer-size		每个缓冲区片段的大小（以字节为单位）。如果没有指定，则根据系统的可用 RAM 设置大小： • 512 字节，少于 64 MB RAM • 64 MB - 128 MB RAM 为 1024 字节(1 KB) • 16384 字节(16 KB)超过 128 MB RAM
buffers-per-slice		将更大的缓冲区划分为多少个片段或部分。这比分配许多独立缓冲区的内存效率更高。如果没有指定，根据系统的可用 RAM 设置分片数量：* 10 代表小于 128 MB RAM，20 代表 128 MB RAM，超过 128 MB RAM
direct-buffers		缓冲区池是否使用直接缓冲区，在很多情况下，在 NIO 中会更快。请注意，有些平台不支持直接缓冲区。

注意

从 JBoss EAP 7.2 开始，IO 缓冲区池已弃用。虽然它们仍然设置为当前版本中的默认设置，它们将在以后的版本中被 Undertow 字节缓冲区池替代。

更新于 2024-02-08