Red Hat Satellite 6.14

调优 Red Hat Satellite 的性能

优化 Satellite 服务器和 Capsule 的性能

Last Updated: 2024-08-09

Red Hat Satellite 6.14 调优 Red Hat Satellite 的性能

优化 Satellite 服务器和 Capsule 的性能

Red Hat Satellite Documentation Team
satellite-doc-list@redhat.com

法律通告

摘要

本指南旨在涵盖一组调优和提示，可用于扩展 Red Hat Satellite 环境。

目录

向红帽文档提供反馈

我们感谢您对我们文档的反馈。让我们了解如何改进它。

使用 Red Hat JIRA 中的 **Create Issue** 表单提供您的反馈。JIRA 问题在 Red Hat Satellite Jira 项目中创建，您可以在其中跟踪其进度。

先决条件

- 确保您已注册了 红帽帐户。

流程

1. 单击以下链接： **创建问题**。如果 Jira 显示登录错误，则登录并在您重定向到表单后继续。
2. 完成 **Summary** 和 **Description** 字段。在 **Description** 字段中，包含文档 URL、章节号以及问题的详细描述。不要修改表单中的任何其他字段。
3. 点 **Create**。

第 1 章 性能调优简介

本文档提供了针对性能和可扩展性调整 Red Hat Satellite 的指南。虽然已经考虑到了很多关注，使内容适用于涵盖大量用例，但如果有些用例没有被覆盖，请随时联系红帽以获得对未记录的用例的支持。

第 2 章 性能调优快速入门

您可以使用 Satellite 中包含的调优配置文件中的内置配置集（使用安装例程的调优标记），根据预期的受管主机数量和硬件分配来调优 Satellite 服务器。如需更多信息，请参阅 在 *连接的网络环境中安装 Satellite 服务器 中的 使用预定义的配置文件调优 Satellite 服务器*。

根据 Satellite 管理的受管主机数量的估算提供四个大小：您可以在 **/usr/share/foreman-installer/config/foreman.hiera/tuning/sizes** 中包含的配置文件中找到每个配置集的特定调优设置。

Name	受管主机数量	建议 RAM	推荐的内核
default	0 – 5000	20 GiB	4
中	5000 – 10000	32 GiB	8
大	10000 – 20000	64 GiB	16
extra-large	20000 – 60000	128 GiB	32
extra-extra-large	60000+	256 GiB+	48+

流程

1. 选择一个安装大小： **default**、m**ium**、**large**、**extra-large** 或 **extra-extra-large**。默认值为 **default**。
2. 运行 **satellite-installer** ：

```
# satellite-installer --tuning "My_Installation_Size"
```

3. 可选：运行健康检查。更多信息请参阅 第 5.1 节 “应用配置”。
4. 可选：直接使用 Puma Tuning 部分来直接处理 Ruby 应用程序服务器。如需更多信息，请参阅 第 5.2 节 “puma 调优”。

第 3 章 调优系统要求

您可以在 *连接的网络环境中安装 Satellite 服务器中的* 为安装准备 硬件和软件要求。

第 4 章 确定硬件和操作系统配置

CPU

对于 Satellite 可用的物理内核越多，可以为任务实现更高的吞吐量。Puppet 和 PostgreSQL 等一些 Satellite 组件是 CPU 密集型应用，可以从更多数量的可用 CPU 内核中受益。

内存

运行 Satellite 的系统中可用内存越高，最好是 Satellite 操作的响应时间。由于 Satellite 使用 PostgreSQL 作为数据库解决方案，因此任何额外的内存都与调优相结合，将由于内存中的数据保留增加而提高应用程序的响应时间。

磁盘

由于存储库同步、软件包数据检索、对内容主机的订阅记录的高频率数据库更新，建议在高速度 SSD 上安装 Satellite，以避免因为磁盘读或写增加而发生性能瓶颈。Satellite 要求磁盘 IO 达到或超过 60mvapich-PROFILE80MB，读取操作的平均吞吐量为每秒。在低于这个值时，对于 Satellite 操作可能会有严重影响。与 HDD 相比，PostgreSQL 等 Satellite 组件从使用 SSD 中受益。

Network

Satellite 服务器和 Capsule 之间的通信会受到网络性能的影响。需要具有最低 jitter 和低延迟的网络，才能启用一些空闲操作，如 Satellite 服务器和 Capsules 同步（至少确保不会导致连接重置等）。

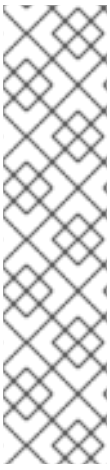
服务器电源管理

默认情况下，您的服务器可能会被配置为节省电源。虽然这是在检查过程中保留最大功耗的好方法，但它也会降低 Satellite 可能会达到的性能。对于运行 Satellite 的服务器，建议将 BIOS 设置为使系统以性能模式运行，以提高 Satellite 可实现的最大性能级别。

4.1. 基准测试磁盘性能

我们正努力更新 **satellite-maintain**，以仅在其内部快速存储基准在我们的推荐吞吐量下产生数字时警告用户。

另外，还可在更新的基准脚本上运行（将来可能会集成到 **satellite-maintain** 中），以获得更准确的实际存储信息。



注意

- 您可能需要临时减少 RAM 才能运行 I/O 基准。例如，如果您的 Satellite 服务器有 256 GiB RAM，则测试将需要 512 GiB 的存储才能运行。作为临时解决方案，您可以在系统引导过程中在 grub 中添加 **mem=20G** 内核选项，以临时减少 RAM 的大小。该基准在指定目录中创建一个文件两倍的 RAM 大小，并对其执行一系列存储 I/O 测试。文件的大小可确保测试不仅仅是测试文件系统缓存。如果您对其他文件系统进行基准测试，如 PostgreSQL 存储等较小的卷，您可能需要缩小上述 RAM 大小。
- 如果您使用不同的存储解决方案，如 SAN 或 iSCSI，您可以预期不同的性能。
- 红帽建议您在执行此脚本前停止所有服务，系统会提示您这样做。

此测试不使用直接 I/O，并将利用文件缓存作为正常操作。

您可以找到我们的脚本 storage-benchmark 的第一个版本。要执行它，请只将脚本下载到 Satellite 中，使其可执行并运行：

```
# ./storage-benchmark /var/lib/pulp
```

如脚本中的 README 块中所述，通常要在以下测试中看到平均 100MB/sec 或更高版本：

- 基于 SSD 的本地 SSD 应该为 600MB/sec 或更高值提供值。
- spinning 磁盘应该提供范围为 100wagon-wagon200MB/sec 或更高值。

如果您看到下面的值，请创建一个支持问题单以获得帮助。

如需更多信息，请参阅 对 Satellite 操作的影响。

4.2. 启用调优配置集

Red Hat Enterprise Linux 8 在安装过程中默认启用 tuned 守护进程。在裸机上，红帽建议在 Satellite 服务器和 Capsules 上运行 **throughput-performance** tuned 配置集。在虚拟机上，红帽建议运行 **virtual-guest** 配置集。

流程

1. 检查 **tuned** 是否正在运行：

```
# systemctl status tuned
```

2. 如果 **tuned** 没有运行，请启用它：

```
# systemctl enable --now tuned
```

3. 可选：查看可用 **tuned** 配置集列表：

```
# tuned-adm list
```

4. 根据您的场景启用 **tuned** 配置集：

```
# tuned-adm profile "My_Tuned_Profile"
```

4.3. 禁用透明巨页

透明巨页是一种内存管理技术，Linux 内核使用较大的大小的内存页面可减少使用 Translation Lookaside Buffer (TLB)的开销。由于具有 Sparse Memory Access 模式而非 Contiguous Memory 访问模式的数据库，当启用 Transparent Hugepage 时，数据库工作负载通常性能不佳。要提高 PostgreSQL 和 Redis 性能，请禁用 Transparent Hugepage。在数据库在单独的服务器上运行的部署中，只有 Satellite 服务器上使用 Transparent Hugepage 可能会有一个小的好处。

有关如何禁用 Transparent Hugepage 的更多信息，请参阅 如何在 Red Hat Enterprise Linux 上禁用透明巨页(THP)。

第 5 章 配置 SATELLITE 的性能

Satellite 附带多个组件，它们相互通信。您可以独立调整这些组件，以达到您的场景的最大可能性能。

5.1. 应用配置

在以下部分中，我们建议各种可调项以及如何应用它们。请始终首先在非生产环境中测试更改，具有有效的备份以及正确的中断窗口，因为大多数情况需要 Satellite 重启。

最好在应用任何更改前设置监控，因为它允许您评估更改的影响。我们的测试环境可能比您认为，尽管我们很难模拟真实环境。

更改 systemd 服务文件

如果您更改了一些 systemd 服务文件，则需要通知 systemd 守护进程重新载入配置：

```
# systemctl daemon-reload
```

重启 Satellite 服务：

```
# satellite-maintain service restart
```

更改配置文件

如果您更改了配置文件，如 **/etc/foreman-installer/custom-hiera.yaml**，请重新运行安装程序以应用您的更改：

```
# satellite-installer
```

使用附加选项运行安装程序

如果您需要重新运行安装程序，并添加了一些新选项：

```
# satellite-installer new options
```

检查设置的基本健全性

可选：更改后，运行这个快速 Satellite 健康检查：

```
# satellite-maintain health check
```

5.2. PUMA 调优

puma 是一个 ruby 应用服务器，用于向客户端提供 Foreman 相关请求。对于处理大量客户端或频繁操作的 Satellite 配置，务必要适当地调整 Puma。

5.2.1. puma Threads

Puma 线程数量（每个 Puma worker）使用两个值进行配置： **threads_min** 和 **threads_max**。

threads_min 的值决定了每个 worker 在启动时生成的线程数量。然后，由于并发请求来自且需要更多的线程，因此 worker 将生成更多 worker，最多生成 **threads_max** 限制。

我们建议将 **threads_min** 设置为与 **threads_max** 的值与具有较少的 Puma 线程的值会导致 Satellite 服务器上的内存使用高。

例如，我们使用并发注册测试比较了这两个设置：

带有 8 个 CPU、40 GiB RAM 的 Satellite 虚拟机	带有 8 个 CPU、40 GiB RAM 的 Satellite 虚拟机
--foreman-foreman-service-puma-threads-min=0	**--foreman-foreman-service-puma-threads-min=16**
--foreman-foreman-service-puma-threads-max=16	**--foreman-foreman-service-puma-threads-max=16**
--foreman-foreman-service-puma-workers=2	**--foreman-foreman-service-puma-workers=2**

与 **threads_min=0** 相比，将最小 Puma 线程设置为 **16** 会产生大约 12% 的内存用量。

5.2.2. Puma Worker 和 Threads Auto-Tuning

如果您没有通过 **satellite-installer** 提供任何 Puma worker 和线程值，或者 Satellite 配置中没有它们，**satellite-installer** 会配置均衡的 worker 数量。它遵循这个公式：

```
min(CPU_COUNT * 1.5, RAM_IN_GB - 1.5)
```

在大多数情况下，这应该可以正常工作，但有些使用模式调整需要限制专用于 Puma 的资源量（因此其他 Satellite 组件可以使用这些资源）或其它原因。每个 Puma worker 大约消耗 1 GiB 内存。

查看您当前的 Satellite 服务器设置

```
# cat /etc/systemd/system/foreman.service.d/installer.conf
```

查看当前活跃的 Puma worker

```
# systemctl status foreman
```

5.2.3. 手动调整 Puma worker 和线程数

如果您决定不依赖于 第 5.2.2 节 “Puma Worker 和 Threads Auto-Tuning”，您可以为这些可调项应用自定义数字。在以下示例中，我们使用 2 个 worker、5 和 5 个线程：

```
# satellite-installer \
--foreman-foreman-service-puma-workers=2 \
--foreman-foreman-service-puma-threads-min=5 \
--foreman-foreman-service-puma-threads-max=5
```

将您的更改应用到 Satellite 服务器。更多信息请参阅 第 5.1 节 “应用配置”。

5.2.4. Puma Workers 和 Threads Recommendations

为了为不同的调优配置文件推荐线程和 worker 配置，我们使用不同的调优配置文件对 Satellite 进行

Puma 调优测试。此测试中使用的主测试是与以下组合并发注册，以及不同数量的 worker 和线程。我们的建议仅基于并发注册性能，因此可能无法反映您的具体用例。例如，如果您的设置对于大量发布和提升而言非常内容，您可能希望限制 Puma 所消耗的资源，而代之为 Pulp 和 PostgreSQL。

Name	受管主机数量	RAM	内核	推荐的 min 和 max 的 Puma 线程	推荐的 Puma Worker
default	0 – 5000	20 GiB	4	16	4 – 6
中	5000 – 10000	32 GiB	8	16	8 – 12
大	10000 – 20000	64 GiB	16	16	12 – 18
extra-large	20000 – 60000	128 GiB	32	16	16 – 24
extra-extra-large	60000+	256 GiB +	48+	16	20 – 26

此处调优工作线程数量是更重要的方面，在某些情况下，我们看到的性能提高最多 52%。虽然安装程序默认使用 5 分钟/最大线程，但我们推荐使用上表中所有调优配置文件的 16 个线程。这是因为，与设置有 4 个线程时，我们看到的性能提升高达 23%(16 个线程为 14%，8% 为 32)。

要找出这些建议，我们使用了并发注册测试案例，这是非常具体的用例。您的 Satellite 上可能会有不同，这可能具有更均衡的用例（不仅注册）。保留默认的 5 分钟/最大线程是很好的选择。

以下是导致我们这些建议的一些测量：

	4 个 worker、4 个线程	4 个 worker、8 个线程	4 个 worker、16 个线程	4 个 worker，32 个线程
改进了	0%	14%	23%	10%

在默认设置(4 个 CPU)上使用 4 个 workers (4 个 CPU)- 与 2 个 worker 相比，我们看到了大约 25% 的性能，但与 2 个 worker 相比，只有 8 个 worker 的性能较低，请参阅下表：

	2 个 worker，16 个线程	4 个 worker、16 个线程	6 个 worker、16 个线程	8 个 worker、16 个线程
改进了	0%	26%	22%	-8%

在中型设置(8 个 CPU)上使用 8wagon-wagon12 worker - 请参考下表：

	2 个 worker，16 个线程	4 个 worker、16 个线程	8 个 worker、16 个线程	12 个 worker，16 个线程	16 个 worker，16 个线程
改进了	0%	51%	52%	52%	42%

在 32 个 CPU 设置上使用 16 个是 16 个 CPUs 设置（这在 90 GiB RAM 机器上测试，这是系统开始交换的因素 - 正确的 *超大* 应该有 128 GiB），对于我们测试的高注册并发级别，这会出现问题，因此我们无法推荐它。

	4 个 worker、16 个线程	8 个 worker、16 个线程	16 个 worker，16 个线程	24 个 worker，16 个线程	32 个 worker、16 个线程	48 个 worker，16 个线程
改进了	0%	37%	44%	52%	失败太多	失败太多

5.2.5. 配置 Puma Worker

如果您有足够的 CPU，则添加更多 worker 会增加性能。例如，我们将 Satellite 设置与 8 和 16 个 CPU 进行比较：

表 5.1. Satellite-installer 选项用于测试 worker 数量的影响

带有 8 个 CPU、40 GiB RAM 的 Satellite 虚拟机	带有 16 个 CPU 的 Satellite 虚拟机，40 GiB RAM
--foreman-foreman-service-puma-threads-min=16	**--foreman-foreman-service-puma-threads-min=16**
--foreman-foreman-service-puma-threads-max=16	**--foreman-foreman-service-puma-threads-max=16**
--foreman-foreman-service-puma-workers={2|4|8|16}	**--foreman-foreman-service-puma-workers={2|4|8|16}**

在 8 个 CPU 设置中，将 worker 数量从 2 改为 16，将并发注册时间增加到 36%。在 16 个 CPU 设置中，同样的更改会导致 55% 的提高。

添加更多 worker 也可以帮助总注册并发 Satellite 处理。在我们的测量中，使用 2 个 worker 设置可以处理最多 480 个并发注册，但添加更多 worker 提高了这种情况。

5.2.6. 配置 Puma 线程

更多线程可以减少时间并行注册主机。例如，我们比较了这两个设置：

带有 8 个 CPU、40 GiB RAM 的 Satellite 虚拟机	带有 8 个 CPU、40 GiB RAM 的 Satellite 虚拟机
--foreman-foreman-service-puma-threads-min=16	**--foreman-foreman-service-puma-threads-min=8**

带有 8 个 CPU、40 GiB RAM 的 Satellite 虚拟机	带有 8 个 CPU、40 GiB RAM 的 Satellite 虚拟机
--foreman-foreman-service-puma-threads-max=16	**--foreman-foreman-service-puma-threads-max=8**
--foreman-foreman-service-puma-workers=2	**--foreman-foreman-service-puma-workers=4**

使用更多 worker 和相同线程总数会导致在高度并发注册场景中大约有 11% 的速度。此外，添加更多 worker 不会消耗更多 CPU 和 RAM，但会获得更好的性能。

5.2.7. 配置 Puma DB 池

$db_pool 的有效值设为 equal **$foreman::foreman_service_puma_threads_max**。它是 **$foreman::db_pool** 和 **$foreman::foreman_service_puma_threads_max** 的最大值，但其默认值为 5，因此对 5 以上的最大线程的增加都会自动增加数据库连接池。

如果您遇到 **ActiveRecord::ConnectionTimeoutError: 无法从 5.000 秒(waited 5.006 秒)内的池获取连接；所有池的连接都在 /var/log/foreman/production.log** 中出现错误，您可能需要增加这个值。

查看当前的 db_pool 设置

```
# grep pool /etc/foreman/database.yml
  pool: 5
```

5.2.8. 手动调整 db_pool

如果您决定不依赖于自动配置的值，您可以应用自定义数字，如下所示：

```
# satellite-installer --foreman-db-pool 10
```

将您的更改应用到 Satellite 服务器。更多信息请参阅 第 5.1 节 “应用配置”。

5.3. APACHE HTTPD 性能调优

Apache httpd 组成了 Satellite 的核心部分，并充当处理通过 Satellite Web UI 或公开的 API 发出的请求的 Web 服务器。为提高操作的并发性，httpd 形成第一点，调优有助于提高 Satellite 的性能。

5.3.1. 为 Apache HTTPD 配置 Open Files 限制

在进行调优后，Apache httpd 可轻松在服务器上打开很多文件描述符，该描述符可能会超过大多数 Linux 系统的默认限制。为了避免因系统上超过最大打开文件限制而导致的任何问题，请创建以下文件和目录并设置文件的内容，如下例所示：

流程

1. 在 **/etc/systemd/system/httpd.service.d/limits.conf** 中设置最大打开文件限制：

```
[Service]
LimitNOFILE=640000
```

2. 将您的更改应用到 Satellite 服务器。如需更多信息，请参阅 第 5.1 节 “应用配置”。

5.3.2. 调整 Apache Httpd Child 进程

默认情况下，httpd 使用事件请求处理机制。当对 httpd 的请求数量超过可启动以处理传入连接的子进程的最大数量时，httpd 会引发 HTTP 503 Service Unavailable 错误。amidst httpd 不足以处理进程，传入的连接也可以在 Satellite 服务一端导致多个组件失败，因为 httpd 进程提供某些组件的依赖项。

您可以调整 httpd 事件的配置，以根据预期的峰值负载处理更多并发请求。

警告

在 **custom-hiera.yaml** 中配置这些数字会锁定它们。如果您使用 **satellite-installer --tuning=*My_Tuning_Option*** 更改了这些数字，则您的 **custom-hiera.yaml** 将覆盖此设置。仅在有特定需要时才设置您的数字。

流程

1. 通过更改或添加以下几行来修改 **/etc/foreman-installer/custom-hiera.yaml** 中的并发请求数：

```
apache::mod::event::serverlimit: 64
apache::mod::event::maxrequestworkers: 1024
apache::mod::event::maxrequestsperchild: 4000
```

   该示例与在 Satellite 服务器上运行 **satellite-installer --tuning=medium** 或更高版本相同。

2. 将您的更改应用到 Satellite 服务器。更多信息请参阅 第 5.1 节 “应用配置”。

5.4. QDROUTERD 和 QPID TUNING

5.4.1. 计算 qdrouterd 的最大打开文件限制

在使用带有大量内容主机的 **katello-agent** 基础架构部署时，可能需要增加 qdrouterd 的最大打开文件。

使用以下公式计算 qdrouterd 中打开文件的限制：**(N x 3)+ 100**，其中 N 是内容主机的数量。每个内容主机最多可能会消耗路由器中的三个文件描述符，并且需要 100 个文件描述符来运行路由器本身。

以下设置允许 Satellite 扩展至 10.000 内容主机。

流程

1. 在 **/etc/foreman-installer/custom-hiera.yaml** 中设置最大打开文件限制：

```
qpid::router::open_file_limit: "My_Value"
```

   默认值为 **150100**。

2. 将您的更改应用到 Satellite 服务器。更多信息请参阅 第 5.1 节 “应用配置”。

5.4.2. 计算 qpidd 的最大打开文件限制

在使用带有大量内容主机的 **katello-agent** 基础架构部署时，可能需要增加 qpidd 的最大打开文件。

使用以下公式计算 qpidd 中打开文件的限制： **(N x 4)+ 500**，其中 N 是内容主机的数量。对于 Broker 操作需要单个内容主机最多可消耗四个文件描述符和 500 个文件描述符(qpidd 的一个组件)。

流程

1. 在 **/etc/foreman-installer/custom-hiera.yaml** 中设置最大打开文件限制：

```
qpid::open_file_limit: "My_Value"
```

   默认值为 **65536**。

2. 将您的更改应用到 Satellite 服务器。更多信息请参阅 第 5.1 节 “应用配置”。

5.4.3. 配置最大异步输入输出请求

在使用带有大量内容主机的 **katello-agent** 基础架构部署中，可能需要增加最大允许的并发 AIO 请求。您可以通过增加内核参数 **fs.aio-max-nr** 来增加允许并发 AIO 请求的最大数量。

流程

1. 将 **fs.aio-max-nr** 的值设置为 **/etc/sysctl.d** 中的文件中所需的最大值：

```
fs.aio-max-nr=My_Maximum_Concurrent_AIO_Requests
```

   确保这个数字大于 33 乘以您计划注册到 Satellite 的最大内容主机数。

2. 应用更改：

```
# sysctl -p
```

3. 可选：重新启动 Satellite 服务器以确保应用了此更改。

5.4.4. 存储注意事项

在规划一个将广泛使用 **katello-agent** 的安装时，请确保提前为 **/var/lib/qpidd** 提供足够的存储空间。在受管主机上，**/var/lib/qpidd** 需要每个内容主机需要 2MiB 磁盘空间。

5.4.5. 配置 QPID mgmt-pub-interval 参数

您可能会在 Red Hat Enterprise Linux 7 中看到以下错误（使用 **journalctl** 命令访问它）：

```
satellite.example.com qpidd[92464]: [Broker] error Channel exception: not-attached: Channel 2 is not attached(/builddir/build/BUILD/qpid-cpp-0.30/src/qpid/amqp_0_10/SessionHandler.cpp: 39
satellite.example.com qpidd[92464]: [Protocol] error Connectionqpid.10.1.10.1:5671-10.1.10.1:53790 timed out: closing
```

此时会出现此错误消息，因为 qpid 为队列、会话和连接维护管理对象，并默认每 10 秒回收一次它们。具有相同 ID 的同一对象会被创建、删除并再次创建。旧的管理对象尚未清除，因此 qpid 会抛出此错误。

流程

1. 在 **/etc/foreman-installer/custom-hiera.yaml** 中设置 **mgmt-pub-interval** 参数：

```
qpid::mgmt_pub_interval: 5
```

2. 将您的更改应用到 Satellite 服务器。更多信息请参阅 第 5.1 节 “应用配置”。
如需更多信息，请参阅 BZ 1335694。

5.5. DYNFLOW TUNING

dynflow 是工作流管理系统和任务编排器，它是 Satellite 插件，用于以不顺序执行方式执行 Satellite 的不同任务。在 Satellite 上检查许多客户端并运行多个任务的条件下，Dynflow 可以从添加的调优中获取一些帮助，指定可以启动哪些 executor。

有关涉及与 Dynflow 相关的调整的更多信息，请参阅 **https://satellite.example.com/foreman_tasks/sidekiq**。

增加 Sidekiq worker 的数量

Satellite 包含一个 Dynflow 服务，名为 **dynflow-sidekiq**，它执行由 Dynflow 调度的任务。sidekiq worker 可以分组到不同的队列中，以确保一种类型的许多任务不会阻止执行其他类型的任务。

红帽建议增加 sidekiq worker 的数量，为批量并发任务扩展 Foreman 任务系统，例如，多个内容视图发布和提升、内容同步和同步到 Capsule 服务器。有两个可用选项：

- 您可以增加 worker 使用的线程数量(worker 的并发性)。由于 Ruby 在线程并发实现时，这对值大于 5 的影响有限。
- 您可以增加 worker 的数量。

流程

1. 将 worker 数量从一个 worker 增加到三个，同时剩余五个线程/并发：

```
# satellite-installer --foreman-dynflow-worker-instances 3    # optionally, add --foreman-dynflow-worker-concurrency 5
```

2. 可选：检查是否有三个 worker 服务：

```
# systemctl -a | grep dynflow-sidekiq@worker-[0-9]
dynflow-sidekiq@worker-1.service       loaded    active   running   Foreman jobs daemon - worker-1 on sidekiq
dynflow-sidekiq@worker-2.service       loaded    active   running   Foreman jobs daemon - worker-2 on sidekiq
dynflow-sidekiq@worker-3.service       loaded    active   running   Foreman jobs daemon - worker-3 on sidekiq
```

如需更多信息，请参阅如何在 Satellite6 中添加 sidekiq worker？

5.6. 基于拉取的 REX 传输调整

Satellite 具有基于拉取的传输模式用于远程执行。此传输模式使用 MQTT 作为其消息传递协议，并包括在每个主机上运行的 MQTT 客户端。如需更多信息，请参阅 *管理主机* 中的 远程执行的传输模式。

5.6.1. 为基于 Pull REX 传输增加主机限制

您可以调整 **mosquitto** MQTT 服务器，并增加与之连接的主机数量。

流程

1. 在 Satellite 服务器或 Capsule 服务器上启用基于拉取的远程执行：

```
# satellite-installer --foreman-proxy-plugin-remote-execution-script-mode pull-mqtt
```

请注意，您的 Satellite 服务器或 Capsule 服务器只能使用一种传输模式，可以是 SSH 或 MQTT。

2. 创建配置文件以增加 MQTT 服务接受的默认主机数：

```
cat >/etc/systemd/system/mosquitto.service.d/limits.conf <<EOF
[Service]
LimitNOFILE=5000
EOF
```

本例设置了限制，以允许 **mosquitto** 服务处理 5000 主机。

3. 运行以下命令以应用您的更改：

```
# systemctl daemon-reload
# systemctl restart mosquitto.service
```

5.6.2. 降低基于 Pull REX 传输的性能影响

当使用 Script provider 为远程执行作业配置 Satellite 服务器时，胶囊服务器通过 MQTT 向客户端发送有关新作业的通知。此通知不包括客户端应该执行的实际工作负载。客户端收到有关新远程执行作业的通知后，它会查询胶囊服务器以获取其实际工作负载。在作业期间，客户端定期将作业输出发送到胶囊服务器，进一步增加对胶囊服务器的请求数量。

这些对胶囊服务器的请求与 MQTT 协议允许的高并发性一起可能会导致胶囊服务器上的可用连接耗尽。有些请求可能会失败，使远程执行作业的一些子任务无响应。这还取决于实际的作业工作负载，因为一些作业会给 Satellite 服务器造成额外的负载，从而在客户端注册到 Satellite 服务器时使其争用资源。

要避免这种情况，请使用以下参数配置 Satellite 服务器和 Capsule 服务器：

- MQTT Time To Live - 在考虑作业未发送前，给主机获取该作业的时间间隔（以秒为单位）
- MQTT Resend Interval - 应重新指向主机通知的时间间隔（以秒为单位），直到作业被获取或取消。
- MQTT Rate Limit - 允许同时运行的作业数量。您可以通过调整速率限制来限制远程执行的并发性，这意味着您要给 Satellite 带来更多负载。

流程

- 在 Satellite 服务器上调整 MQTT 参数：

```
# satellite-installer \
--foreman-proxy-plugin-remote-execution-script-mqtt-rate-limit My_MQTT_Rate_Limit \
--foreman-proxy-plugin-remote-execution-script-mqtt-resend-interval My_MQTT_Resend_Interval \
```

```
--foreman-proxy-plugin-remote-execution-script-mqtt-ttl My_MQTT_Time_To_Live
```

Capsule 服务器日志位于 **/var/log/foreman-proxy/proxy.log** 中。胶囊服务器使用 Webrick HTTP 服务器（不包括 httpd 或 Puma），因此无法简单地提高其容量。

注意

根据工作负载，主机数量、可用资源和应用调整，您可能会达到 Bug 2244811，这会导致 Capsule 消耗太多内存并最终被终止，从而使其余作业失败。目前，没有通用的临时解决方案。

5.7. POSTGRESQL 调优

PostgreSQL 是基于 SQL 的主数据库，供 Satellite 用于在各种任务中存储持久上下文。数据库会发现广泛的使用量，通常是为了为 Satellite 提供其可平稳运行所需的数据。这使得 PostgreSQL 成为大量使用的过程，如果 tuned 对 Satellite 的整体操作响应有很多优点。

PostgreSQL 作者建议在运行 PostgreSQL 的服务器上禁用 Transparent Hugepage。更多信息请参阅 第 4.3 节 “禁用透明巨页”。

您可以将一组调优应用到 PostgreSQL 以提高其响应时间，这将修改 **postgresql.conf** 文件。

流程

1. 附加 **/etc/foreman-installer/custom-hiera.yaml** 以调整 PostgreSQL：

```
postgresql::server::config_entries:
  max_connections: 1000
  shared_buffers: 2GB
  work_mem: 8MB
  autovacuum_vacuum_cost_limit: 2000
```

   您可以使用它来有效地调优 Satellite 实例，而不考虑调优配置文件。

2. 将您的更改应用到 Satellite 服务器。更多信息请参阅 第 5.1 节 “应用配置”。

在上面的调优配置中，我们有一些更改的密钥：

- **max_CONNECTIONS ： 键定义可被运行的 PostgreSQL 进程接受的最大连接数。**
- **shared_buffers** ：共享缓冲区定义 PostgreSQL 中所有活跃连接用来存储不同数据库操作数据的内存。这的最佳值将因 Satellite 上执行操作的频率而异 2 GiB 到总系统内存最多 25%。
- **work_mem**: work_mem 是 PostgreSQL 的每个进程分配的内存，用于存储进程正在执行的操作的中间结果。将此值设置为 8 MB，对于 Satellite 上的大多数密集型操作而言，应该有超过 8 MB。
- **autovacuum_vacuum_cost_limit**: 键定义 autovacuum 进程中 vacuuming 操作的成本限制值，以清理数据库关系中的死元组。成本限制定义了进程可在单个运行中处理的元组数。红帽建议将值设为 **2000**，因为它适用于 *介质*、*大*、*超大和 extra-extra-large* 配置集，具体取决于 Satellite 推送到 PostgreSQL 服务器进程的一般负载。

如需更多信息，请参阅 BZ1867311: Upgrade failed when checkpoint_segments postgres 参数配置 。

5.7.1. 原始 DB 性能基准测试

若要获取 satellite-support git 存储库中的 Candlepin、Foreman 和 Pulp 检查 postgres-size-report 脚本的磁盘空间大小列表。

pgbench 工具（注意您可能需要将 PostgreSQL 数据目录 **/var/lib/pgsql** 目录调整为 100 GiB，或者基准需要运行什么操作）可能会用于测量系统上的 PostgreSQL 性能。使用 **dnf install postgresql-contrib** 进行安装。如需更多信息，请参阅 github.com/RedHatSatellite/satellite-support。

为 PostgreSQL 数据目录选择文件系统也可能很重要。

警告

- 切勿在生产系统和没有有效备份的情况下进行任何测试。
- 在开始测试前，请查看数据库文件有多大。使用真正小的数据库进行测试不会产生任何有意义的结果。例如，如果 DB 只是 20 GiB，并且缓冲区池为 32 GiB，则不会显示大量连接的问题，因为数据将完全缓冲。

5.8. REDIS TUNING

Redis 是一个内存数据存储。它被 Satellite 中的多个服务使用。Dynflow 和 Pulp 任务系统使用它来跟踪其任务。由于 Satellite 使用 Redis 的方式，其内存消耗应该稳定。

Redis 作者建议在运行 Redis 的服务器上禁用 Transparent Hugepage。有关它的详情请参考 第 4.3 节 “禁用透明巨页”。

5.9. 胶囊配置调优

Capsules 旨在卸载 Satellite 负载的一部分，并提供对与向客户端分发内容相关的不同网络的访问权限，但它们也可用于执行远程执行作业。它们无法满足任何广泛使用 Satellite API 作为主机注册或软件包配置文件更新的信息。

5.9.1. Capsule 性能测试

在多个 Capsule 配置中，我们测量了多个测试案例：

Capsule HW 配置	CPU	RAM
Minimal	4	12 GiB
大	8	24 GiB
额外大	16	46 GiB

内容交付用例

在下载测试中，我们在 100, 200, 上同时下载了 40MB 仓库 2000 个软件包。1000 个主机，我们每次双重胶囊服务器资源时平均下载时间约为 50%。有关更精确的数字，请查看下表。

并发下载主机	最小(4 个 CPU 和 12 GiB RAM) → 大(8 个 CPU 和 24 GiB RAM)	大(8 个 CPU 和 24 GiB RAM) → Extra Large (16 CPU 和 46 GiB RAM)	最小(4 个 CPU 和 12 GiB RAM) → Extra Large (16 CPU 和 46 GiB RAM)
平均改进	~ 50%（例如 700 个并发下载平均 9 秒与.每个软件包的 4.4 秒）	~ 40%（例如 700 个并发下载平均 4.4 秒与.每个软件包 2.5 秒	~ 70%（例如 700 个并发下载平均 9 秒与.每个软件包 2.5 秒

当我们从 Satellite 服务器与 Capsule 服务器下载性能进行比较时，我们只看到 5% 的速度，但正因为胶囊服务器的主要好处是更接近地域分布的客户端（或不同网络中的客户端）以及处理负载服务器的一部分时，必须处理自身。在一些较小的硬件配置(8 个 CPU 和 24 GiB)中，Satellite 服务器无法处理从 500 多个并发客户端下载，而具有相同硬件配置的胶囊服务器可以服务超过 1000，甚至可能更多。

并发注册用例

*对于并发注册，瓶颈通常是 CPU 速度，但所有配置都能够在不交换的情况下处理高并发性。用于 Capsule 的硬件资源只会影响注册性能。例如，与具有 4 个 CPU 和 12 GiB RAM 的胶囊服务器相比，具有 16 个 CPU 和 46 GiB RAM 的 Capsule 服务器最多会提高 9% 的注册速度。在非常高并发的期间，您可能会在胶囊服务器到 Satellite 服务器通信中遇到超时。您可以使用 **/etc/foreman-installer/custom-hiera.yaml** 中的以下可调整来增加默认超时：*

```
apache::mod::proxy::proxy_timeout: 600
```

远程执行用例

我们测试了通过 SSH 和 Ansible 后端在 500、2000 和 4000 主机上执行远程执行作业。所有配置都能够处理所有测试时没有错误，除了在所有 4000 主机上无法完成的最小配置(4 个 CPU 和 12 GiB 内存)。

内容同步用例

在同步 Red Hat Enterprise Linux 6、7、8 BaseOS 和 8 AppStream 的同步测试中，我们没有在 Capsule 配置中看到显著区别。这与并行同步大量内容视图的不同。