AMD

针对基于AMDEPYC™7002系列处理器的服务器上的低延迟响应的性能调整指南

出版单位 57037_1.0

修订版 1.0

发行日期 2020年11月，

Chapter 1 产品概述

低延迟市场细分市场，如金融交易或实时处理，要求服务器提供系统响应不到10µs的变化。在金融界，从事高频交易(HFT)工作的公司在股票市场上不妥协地快速执行死刑的能力是他们的成功和盈利能力的关键。在这里，即使是微秒也可能产生重大的业务影响，它们需要从计算到网络的超低延迟。

本文档提供了关于使用两个AMDEPYC™7F32处理器或其他高频sku，如7F52和7F72处理器来调整服务器的指导，以通过减少不需要的计算抖动来达到严格的低延迟要求。该指南包括硬件配置、BIOS设置、操作系统内核配置以及控制目标应用程序环境的脚本。请阅读第2章，以了解用于开发此性能调优指南的系统规范和配置。本文档重点介绍了服务器的资源，并提供了有关调整BIOS和OS以优化HFT环境的服务器性能的详细信息。

1.1 系统上的抖动

Sys抖动是一种测量运行在指定内核上的用户进程中的Jitter事件的工具，并报告所发生的中断的统计信息。这允许HFT社区中的用户确保其服务器被正确调整，以满足他们的期望。请参阅第五章关于Sysjittet，以了解更多关于如何使用定制脚本运行Sysjittet以及在阅读结果时要寻找什么的信息。

Chapter 2 软硬件设备 设备配置

本技术论文是使用HPE推进DL385一代10+服务器与最新的HPEBIOS(42BIOS版1.24)和iLOFW版2.30运行红帽企业Linuxx86_648.1与2AMDEPYC7F32处理器，它适用于其他AMDEPYC高频处理器如7F52和7F72。为了在μs范围内实现低延迟，了解被测系统(SUT)的硬件和固件配置是很重要的。影响响应时间的重要因素包括：

- 核数
- 每个核心的执行线程
- 处理器的数量
- NUMA节点数
- NUMA拓扑结构中的CPU和内存排列
- 在NUMA节点中的高速缓存拓扑结构

如果工作负载不受内存带宽的限制，您可以通过以2933MT/s的速度运行内存以与运行在1467MHz的无限结构同步来优化以获得更好的延迟。基于Linux的系统工具，如ip库、中间码等。以不同级别的细节和不同的格式显示配置。

要达到最佳的响应时间，请尽可能优化系统拓扑结构，以满足您的操作需求。注意内存的位置，在NUMA节点上均匀地安装内存，并尝试最大限度地使用本地内存。将执行受时间限制的应用程序的核心与操作系统调度程序隔离，这样其他应用程序和内核线程就不会从应用程序中窃取执行时间。

⚠ 此低延迟性能调优指南文档在AMDEPYC7F32（8核心处理器）上进行了验证，并在配置了两个AMDEPYC7F32处理器的HPEDLLian5Gen10Plus服务器上进行了测试。此外，它适用于AMDEPYC7F52和7F72处理器，但可能
不适合用于具有其他处理器sku的服务器。

为了实现本低延迟性能调优文档，被测试系统(SUT)是一个HPE原型DL385Gen10Plus服务器，具有以下资源/容量。有关此服务器的更多详细信息，请参见它的快速规范：
https://h20195.www2.hpe.com/v2/getdocument.aspx？docname=a00073549enw

2编号生成AMDEPYC™7Fx2处理器	
处理器技术	7纳米
处理器类型/SKU	7F32
核数	8
插座的数量	2
内存速度	3200公吨/秒
内存容量	512GB
存储器	1TBOS驱动器SSD/NVMe
网卡	这是一个基于单节点SUT的测试，1GigE网络就足够了。

Chapter 3 BIOS 设备配置

可以通过RBSU设置实用程序中的BIOS设置来禁用许多系统延迟源。为了使用本文档，我们使用了2xAMDEPYC7F32处理器、3200MT/s的512GBDRAM和红帽企业Linuxx86_648.1操作系统。AMDEPYC™7F32处理器SKU有8个核心(16个线程，跨4个核心复杂死亡(CCD)。

AMDEPYC™7F32处理器	
基频	3.7GHz
最大提升	3.9GHz
核数	8
L1高速缓存器的大小	32KBI+32KBD芯片
L2高速缓存器的大小	每核芯片上的512KBI+D
L3高速缓存器的大小	每台芯片上的128MBI+D，每芯16MB
最大内存速度	3200公吨/秒
最大内存容量	4吨
外设组件互连	128车道PCIeGen4

3.1 注意注意事项

在开始配置低延迟性能优化系统之前，请确保查看下面的重要注释

- 尽可能多地使应用程序的核心与中断隔离。利用本文档中描述的Linux实用程序和技术来最佳地管理硬件组件和属性，如网络适配器的irq。有关详细信息，请参阅有关AMDEPYC™7002系列基于处理器的服务器的Linux®网络调谐指南。

  https://developer.amd.com/wpcontent/resources/56739_Linux%20Network%20tuning%20v0.20.pdf
- 通过启用AMD核心性能提升，可以从CPU核心中提取最大性能。在这样做时，在运行工作负载时监视单个内核的操作频率，以确定是否应限制最大频率(AMDF最大提升限制)，以保持确定性的定时行为。CPU频率的高波动损害系统的系统响应。出于同样的原因，确定性控制应设置为性能确定性，请参见电源/性能确定性文档中的处理器选项设置
- https://www.amd.com/system/files/2017-06/Power-Performance-Determinism.pdfFor其他信息，请参见AMDEPYC™7002系列基于处理器的服务器的工作负载调整指南。https://developer.amd.com/wp-content/resources/56745_0.80.pdf

3.2 BIOS配置文件设置

HPEProLiantDL385Gen10Plus服务器系统的BIOS设置实用程序提供了为各种类型的工作负载设计和调整的HPEBIOS概要文件的列表。您可以选择适合的选项来满足您对工作量的需求。与低延迟相关的两个工作负载有：

- 较低的延迟时间
- 自定义的定义

3.2.1 较低的延迟时间

建议首先选择低延迟工作负载配置文件。若要选择此低延迟工作负载配置文件，

1. 从BIOS/平台配置(RBSU)菜单中，选择低延迟工作负载配置文件。这将选择HPE工程师已确定适合于低延迟操作的BIOS调优选项的组合。
2. 允许服务器完成引导并运行Sys抖动工作负载，以观察核心抖动以及以下章节中建议的Linux操作系统调优。

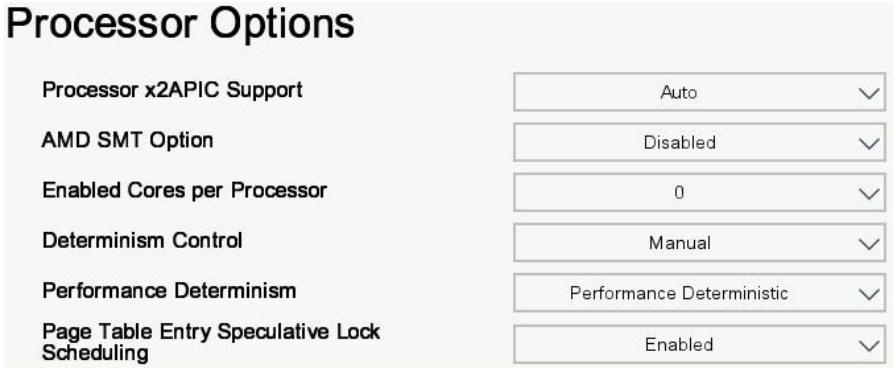
如果您没有获得CPU核心上的预期低延迟性能，那么使用可选的自定义BIOS配置文件，并调整单个BIOS设置旋钮。

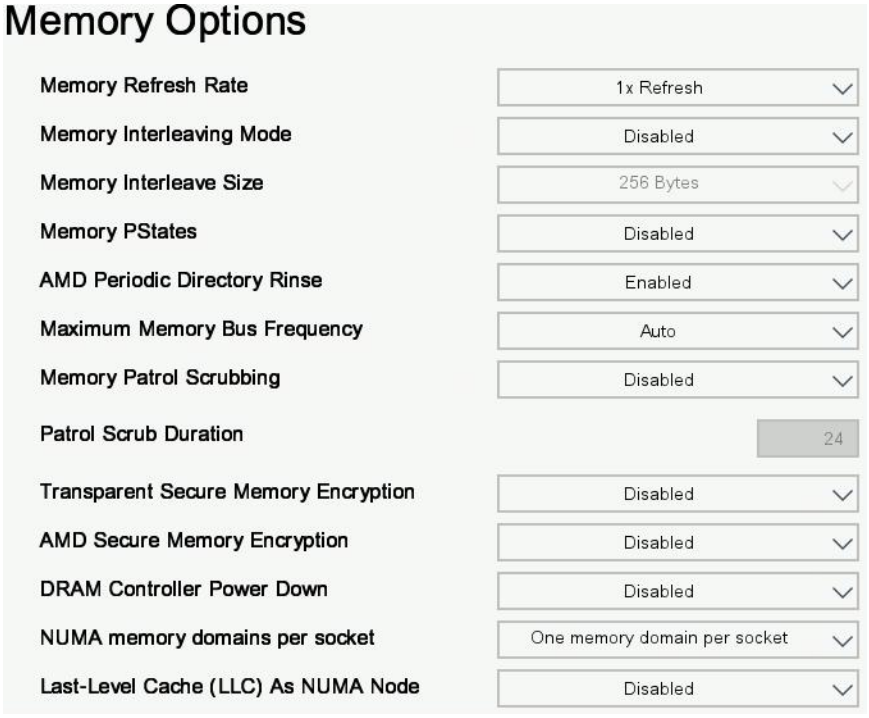
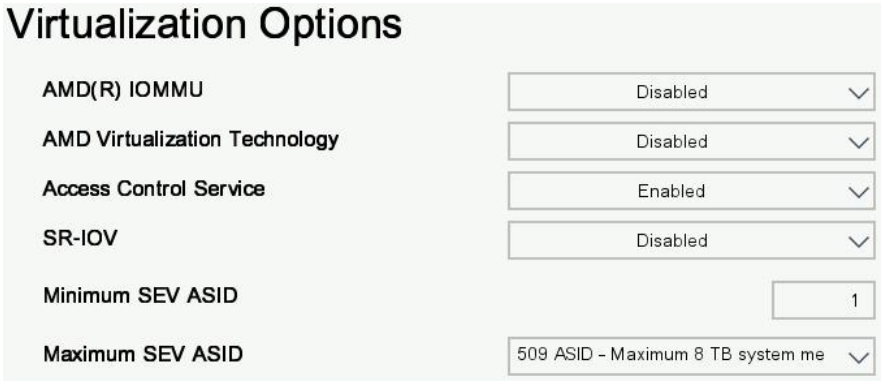
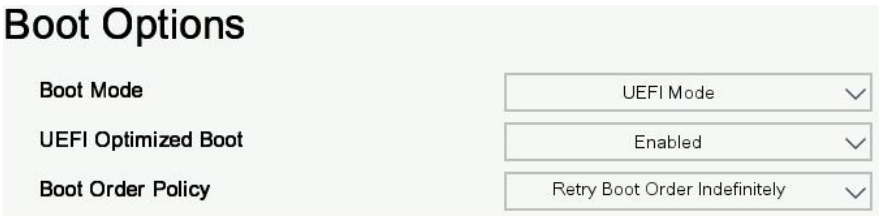
请注意，AMD核心性能提升负载在低延迟工作负载配置文件中显式禁用。如果您希望启用AMD核心性能增强和该工作负载配置文件，请首先选择低延迟工作负载配置文件，然后更改为自定义工作负载配置文件。这样做将保持低延迟BIOS设置不变，但然后允许您启用AMD核心性能提升。请注意，没有必要在选择“低延迟”工作负载配置文件和选择“自定义”工作负载配置文件之间启动系统。

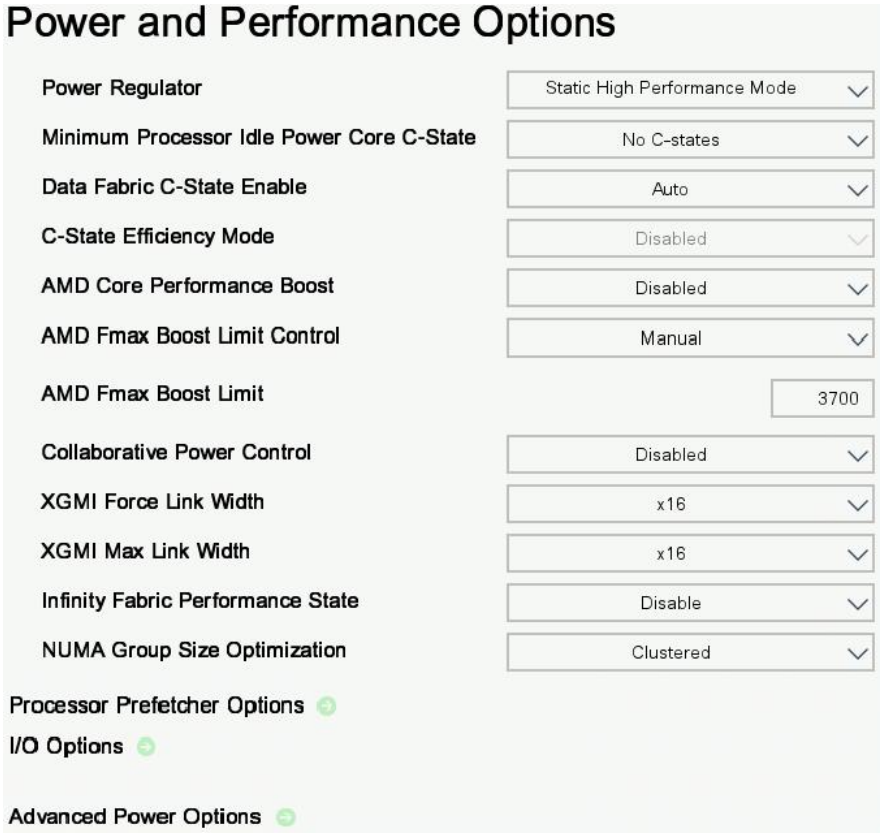
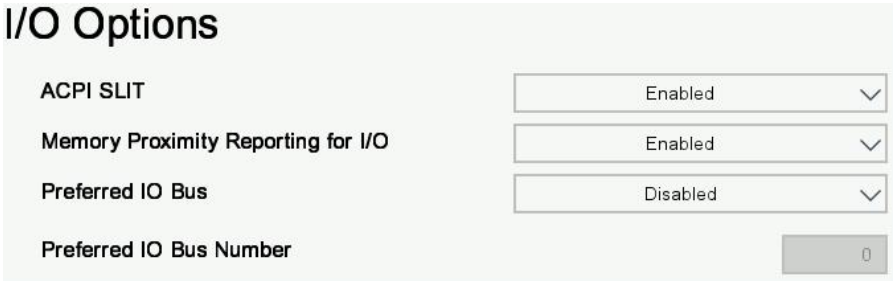
3.2.2 自定义的定义

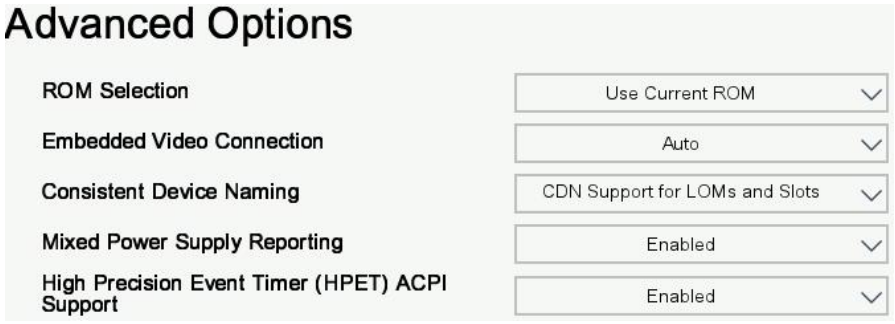
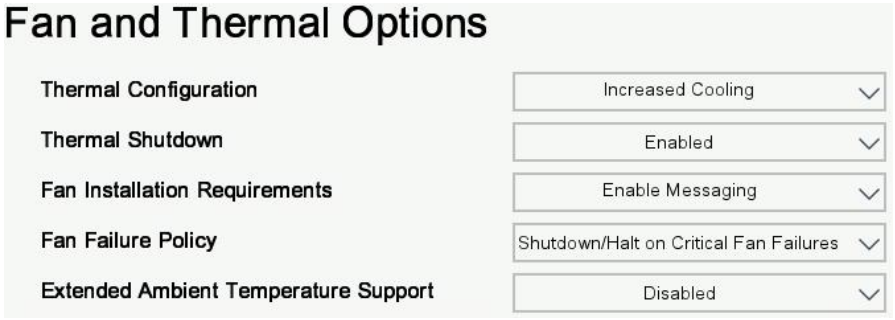
HPEProLiantDL385Gen10Plus服务器系统的BIOS具有一个自定义工作负载配置文件，它不修改任何设置，但允许更改所有设置。通过此选项，您可以启用自定义的低延迟设置，以实现工作负载的延迟选项。

要配置这种自定义低延迟，请从BIOS/平台配置(RBSU)菜单中选择自定义BIOS配置文件，然后根据特定设置调整BIOS调整选项，将其自定义到您的环境的最佳低延迟配置，以减少抖动和提高性能。

BIOS自定义配置文件	
启动时间优化	
诊断程序的选项	
处理器的选项	禁用SMT，确定性控制设置为性能确定性 

内存选项	推荐：此低延迟测试仅使用NPS=1设置进行。 
虚拟化选项	
启动选项	

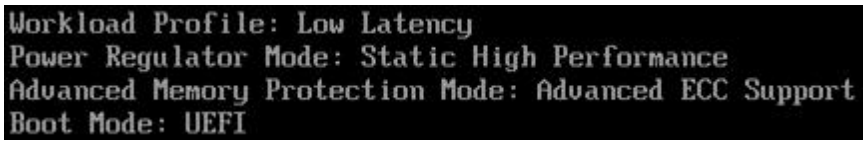
电源和性能选项	选项：您可以通过将AMD核心性能提升更改为“已启用”来提高更高的CPU核心性能 
处理器预取器选项	
输入输出选项	

高级的选项	
高级电源选项	
风扇和热设备选项	选项：要增加更多冷却，将热配置改为“增加冷却” 
高级调试选项	

在BIOS设置完成后，保存这些设置并启动服务器。在服务器启动时，请检查初始化屏幕，其中简要显示您选择的BIOS配置文件选项，以验证工作负载配置文件。

下面是低延迟和自定义配置文件设置选项的屏幕截图示例。

低延迟的配置文件



自定义配置文件

系统启动后，请运行以下系统检查，以确保您的环境准备好以下部分中描述的Linux操作系统级别调整。

3.3 BIOS、内存和CPU版本

下面来自xsos实用程序的输出显示了BIOS、内存和CPU版本。

```
#说明：https://access.redhat.com/discussions/469323#
#yum的安装程序http://people.redhat.com/rsawhill/rpms/latest-rsawaroha-release.rpm
#你要安装xsosrsar
#xsos——bios（产生以下输出）
```

```
  Bios:
    版本：HPE版
    本：A42
    日期：2020年4月29日
    BIOS版本：1.24
    FW版本：     2.30（iLO版本）系统：
    制造商名称：     hpe
    优点：优点DL385Gen10+版本：未指定
    远程存储器和服务器编号：       M77932054N
    UUID：474e4946-4c41-374d-3739-33323035344eCPU：
    2个CPU插槽中的2个，每个CPU8核/16个线程
    共16个核，共32个线程
```

```
      制造商名称：      先进微设备公司公司：   禅
      宗学家
      频率频率：3700MHz
      版本：AMDEPYC7F328核处理器内存：
      总计：524288MiB（512GiB）
      dimm：160个人口群体中的16人
      最大容量：4194304MiB（4096GiB/4.00TiB）
```

3.3.1 NUMA节点

来自numactl-H的输出显示了NUMA节点。

```
可用性：2个节点（0-1）
节点0cpus：01 2 3 4 5 6 7
节点0大小：257464MB
节点0空闲：250483MB
节点1cpus：8 9 10 11 12 13 14 15
节点1大小：258018MB
节点1空闲距离：251048MB节点距
离：
节点     0    1
  0:   10   32
  1:   32   10
```

3.3.2 处理器

来自lscpu的输出显示了两个AMDEPYC2-套接字7F32处理器。

```
#lscpu
体系架构：                x86_64
CPU操作模式(s)：          32位、64位字节的
顺序：                    小端
CPU(s)：                  16
在线CPU(s)列表：0-15
每个铁芯的螺程(s)：       1（SMT在BIOS中关闭）每个插
座的核心(s)：8
插座(s)：                 2
NUMA节点(s)：             2
供应商ID：                进行身份验证的AMD
CPU系列：                 23
产品型号：                49
型号名称：                AMDEPYC7F328核处理器步进：       0
CPUMHz：                    2994.299
BogoMIPS：                7386.00
虚拟化功能：              中v
L1d高速缓存：             32K
L1i高速缓存：             32K
L2高速缓存器：            512K
L3高速缓存器：
                          16384KN
UMA节点0CPU(s)：          0-7
NUMA节点第1个CPU(s)：      8-15
标志：                    我的朋友，我的朋友
mxxextfxsr_optpdpe1gbrdtscplm
```

```
constant_tsc rep_good nopl xtopology nonstop_tsc cpuid extd_apicid aperfmperf pni pclmulqdq monitor
ssse3fma cx16sse4_1sse4_2movbe popcnt aes xsave avx f16c rdrand lahf_lm cmp_legacy svm extapic
cr8_legacy abm sse4a misalignsse3dnowprefetch osvw ibs skinit wdt tce topoext perfctr_core
perfctr_nb bpext perfctr_llc mwaitx cpb cat_l3cdp_l3hw_pstate ssbd mba ibrs ibpb stibp vmmcall
fsgsbase bmi1avx2smep bmi2cqm rdt_a rdseed adx smap clflushopt clwb sha_ni xsaveopt xsavec
xgetbv1xsaves cqm_llc cqm_occup_llc cqm_mbm_total cqm_mbm_local clzero irperf xsaveerptr wbnoinvd
arat npt lbrv svm_lock nrip_save tsc_scale vmcb_clean flushbyasid decodeassists pausefilter
pfthreshold avic v_vmsave_vmload vgif umip rdpid overflow_recov succor
smca
```

3.3.3 内存

来自中间代码(-t17　内存)的输出显示了DRAM的详细信息。

```
句柄0x0027，DMI类型17,84字节内存设备
        阵列手柄：0x0010
        错误信息句柄：未提供总宽度：72位
        数据宽度：64位大小：
        32GB
        形式子：内存设置：无
        定位器：PROC1DIMM1
        银行定位器：未指定的类型：DDR4
        类型详细信息：同步注册（缓冲）速度：3200MT/s
        制造商：三星序列号：39CB333资
        产标签：未指定
        零件号：M393A4G43AB3-CWE
        排名：2
        配置的内存速度：3200MT/s最小电压：1.2V
        最大电压：1.2V配置电压：1.2V内存技
        术：DRAM
        内存操作模式功能：挥发性内存固件版本：未指定
        模块制造商ID：银行1，十六进制0xCE模块产品ID：未知
        内存子系统控制器制造商ID：未知的内存子系统控制器产品ID：未知
        非挥发性大小：无挥发性大
        小：32GB缓存大小：无
        逻辑大小：无
```

Chapter 4 正在操作 系统

SUT中的操作系统是RedHat企业Linux(RHEL)8.1。xsos的输出
显示了操作系统的详细信息。

```
操作系统
   主机名：hpe-lowlat.amd.com
   目录：      [红帽企业Linux8.1发行版(Ootpa)[操作系统发行版]红帽企业
                Linux8.1(Ootpa)8.1(Ootpa)
   rhn：       服务器网址=https://enter.your.server.url.here/XMLRPC启用代理=0
   rhsm：      主机名
               =subscription.rhsm.redhat.comproxy_hostname=
   嗯：        3启用的插件：调试信息安装，产品id，订阅管理器运行级别：N3        （默认为多用户）
   SELinux：   已禁用的功能  （默认值已禁用）
   拱门：      马赫值为=x86_64 cpu=x86_64     平台=x86_64内核：
      已启动的内核：       4.18.0-147.5.1.el8_1.x86_64GRUB
      默认值：
      生成版本：
         Linux版本4.18.0-147.5.1.el8_1.x86_64(mockbuild@x86-vm07.build.eng.bos.redhat.com)(gcc版本
8.3.1 20190507 （红帽8.3.1-4）(GCC))#1
世界2020年1月14日星期二15：50：19
```

4.1 SUT的拓扑结构

安装操作系统后，可以使用命令行界面中的lstopo拓扑工具查看SUT物理拓扑。

对于RHEL8，请使用以下存储库来安装lstopo，它是hwloc包的一部分。安装hwloc-gui软件包，以
获得以多种图形格式格式化lstopo输出的选项。
RHEL8.1中的存储库需要订阅。下面是lstopo所需的订阅量。

```
订阅红帽8
#订阅管理器回购-启用=流-8对x86_64补充rpm#订阅管理器回购-启用=流-8对x86_64基础源rpm
#订阅管理器回购--enable=rhel-8-for-x86_64-appstream-source-rpms#yum安装hwlochwloc-gui-y
```

4.1.1 通过lstopo实现的系统拓扑结构

lstopo工具提供了多种输出格式。图1是由以下命令行生成的图形输出的一个示例。

```
#-物理-输出格式的pngAMD_EPYC_7F32_8-Core_Processor.png
```

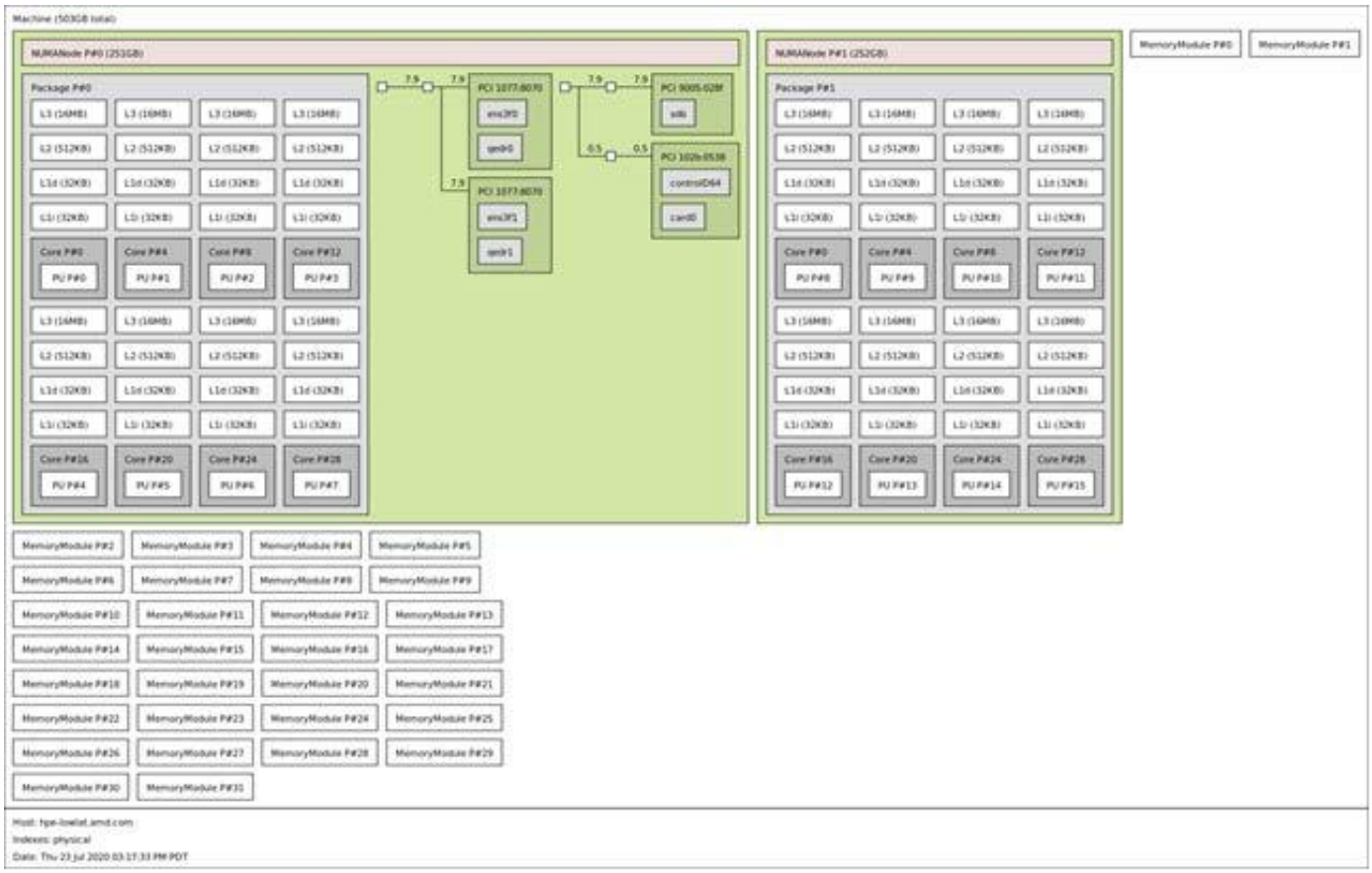


图1AMDEPYC®双处理器高频7F32SUT的停止输出(png)

4.2 配置和调优RHEL8.1的低延迟性能

本节提供了配置RHEL8.1的指南，这将使两个基于AMDEPYC7F32的系统实现最佳性能，以为各种财务、交易和匹配用例获得最佳的低延迟性能。

4.2.1 /proc文件系统

/proc文件系统接口可以公开每个处理器的信息，如内部内核数据、内核子系统和系统设备。要实现低延迟性能，请使用以下内核参数配置SUT。

4211 使用sysctl调优内核参数

这些参数的设置已经通过各种迭代和实验确定，可以通过将值写入自定义sysctl配置文件并将其添加到/etc/sysctl.d目录来使它们持久化。

下表包含了实验过的系统内核参数，以覆盖写入一个名为100-lowlatency.conf的文件中的默认内核参数值，以帮助实现无抖动的系统。文件名前面的数字被选择为此目录中任何命名文件中最大的数字，以便最后读取它。此conf文件的文件内容如下：

这些设置将在启动时从/etc/sysctl.d/100-lowlatency.conf文件中读取。

1. **创建一个文件/etc/sysctl.d/100-lowlatency.conf，并使用以下参数填充它**

#vi/etc/sysctl.d/100-lowlatency.conf

注：将以下脚本插入其中（请注意任何行包装问题）

```
kernel.hung_task_timeout_secs=600
kernel.numa_balancing=0
kernel.numa_balancing_scan_delay_ms=1000
kernel.numa_balancing_scan_period_max_ms=60000
kernel.numa_balancing_scan_period_min_ms=1000
kernel.numa_balancing_scan_size_mb=256
kernel.sched_latency_ns=24000000
kernel.sched_migration_cost_ns=50000000
kernel.sched_min_granularity_ns=100000000
kernel.sched_rt_runtime_us=-1
kernel.timer_migration=1
kernel.watchdog=0
kernel.watchdog_cpumask=0,8
kernel.watchdog_thresh=10
vm.dirty_background_ratio=3
vm.dirty_ratio=10
vm.stat_interval=3600
vm.swappiness=0
vm.zone_reclaim_mode=0
net.ipv4.conf.all.rp_filter=          0
```

要读取sysctlconf文件并立即生效的设置，可以使用以下命令。运行以下命令作为根目录来应用这些新参数，并确保没有错误。

#sysctl-p/etc/sysctl.d/100-lowlatency.conf

4.3 使用Linux调优系统

您可以使用调整的Linux调优其他参数。Linux调整是一个可定制的应用程序，它可以帮助调整和优化系统，以达到其最佳性能。调优后的应用程序包括具有预定义调优选项的配置文件，如吞吐量性能、延迟性能和计算分区。您可以使用这些调优的adm配置文件，而不是直接写入/proc文件系统，使用sysctl，并应用引导命令选项。但是，如果停止或禁用了调优的后台进程，则必须重新启动它以应用调优的配置文件。

4.3.1 使用Linux设置自定义HPELow延迟配置文件

本节介绍使用自定义调优的低延迟调整配置文件准备服务器的过程。HPELow延迟配置文件还修改了多个系统内核设置，以及配置启动时间选项和执行其他调优操作。此外，在系统启动过程中添加了一个“一次性”的Linux服务，并在系统启动期间执行一个“one-shot.sh”shell脚本。您可以使用下面的步骤创建指定的文件，并将它们应用于没有任何额外修改的HPEProLiantGen10Plus服务器，具有两个AMDEPYC高频处理器，如7F32。

> 当您复制和粘贴时，请一定要检查该代码。由于此文档的页大小，代码页行被包装，因此在复制后检查任何不需要的空格、注释和换行符，并在执行之前解决这些问题。

请确保为静态设置配置了已调整的内容。为了减少抖动和获得低延迟性能，我们不是使用预定定义的配置文件，而是创建一个定制的HPEL低延迟调整配置文件。

按照下面描述的步骤创建自定义HPEL低延迟调优配置文件，并修改tuned.conf和cpu-partitioning.conf文件。

4.3.2 使用调优的自定义配置文件准备HPELow延迟

4321 配置1-HPELow延迟

4.3.2.1.1 在/etc/调整/HPEL低延迟目录下添加“tuned.conf”配置设置

```
#mkdir/等/调优/HPELow延迟
#vi/etc/tuned/HPELowLatency/tuned.conf
注：将以下脚本插入其中（请注意任何行包装问题）
##############################start#########################################
#! HPELow潜伏期包括几个文件：\
#! 这个文件，/etc/tuned/HPELowLatency/tuned.conf，它是一个基于cpu分区配置文件的调整配置文件
#! 一个辅助脚本文件/etc/tuned/HPELowLatency/script.sh，被cpupartitioning-variables.conf、
/usr/local/bin/oneshot_script.sh和/opt/run_sysjit.sh脚本用于运行第五章中的系统抖动程序
#! 经过全局调谐的配置文件/etc/tuned/tuned-main.conf
#! /etc/tuned/cpu-partitioning-variables.conf，它为计算分区配置文件设置变量
#!
#! 除了“调优”的文件外，还有这些系统的启动文件
#!    /etc/systemd/system/one-time.service，它定义了要在启动时执行命令的服务
#!    /usr/local/bin/oneshot_script.sh，它是一个可以在启动时运行以影响其他更改的脚本。

[主页页）
摘要=与HPE低延迟修改相关的=及其cpu分区配置文件描述=修改cpu分区配置文件，使每个节点的第一个CPU进行管
理，允许接下来的几个平衡，其余禁用平衡。

包括=cpu分区[磁盘]
电梯=没有

[引导加载程序
#! 调音和/或食物的问题导致第一个单词被吃掉。#! 建议如下，引用我的红帽联系人：
#! 使用的解决方法是添加截断的引导标志
#! 在/etc/默认/grub文件中缺少GRUB_CMDLINE_LINUX变量。
#! 然后运行“grub2-mkconfig-o/boot/grub2/grub.cfg”来获取它，并重新启动。
#! 使用一个被丢弃的第一个单词的解决方法，添加“安静”作为第一个选项到#! 控制线。请确保下面的cmd线是一
条线的条目，并注意#! 为行包装。

提示页=安静的自乐乐=0提示=ignore_ce ipv6.disable=1审计=0nmi_watchdog=0提示页=2MB提示页
=6000default_hugepagesz=2MBtransparent_hugepage=从不tsc=可靠的pcie_aspm=从cpuidle.off=1rcu_nocb_poll空闲
=轮询processor.max_cstate=0

#! 来自https://tuned-project.org/docs/tuned_devconf_2019.pdf[调度程
序]group.ksoftirqd=${f：exec：/etc/tuned/HPELowLatency/script.sh：
ksoftirqd}group.rcub=${f：exec：/etc/tuned/HPELowLatency/script.sh：
rcub}##############################END#########################################的参考
#注：
#a.      内核启动参数“cmdline”使用tuned.conf文件实现，并#将您的首选启动时间参数作为永久参数进
行实验。
#b.      并确保您的更改在#最终重新启动后生效。
##############################end#############################################
```

这个新的HPELow延迟优化配置文件还包括“cpu分区”配置文件及其相关文件cpu-partitioning-variables.conf（参见4.3.2.1.3节），它为已安装的AMDEPYC高频处理器设置适当的内务管理和隔离核心。

4.3.2.1.2 在/etc/调优/HPEL低延迟”目录下添加调优的“script.sh”

#vi/etc/tuned/HPELowLatency/script.sh

注：放置以下脚本（请注意任何行包装问题）

```
#! /bin/bash
##############################start##########################################
函数range_expand(){
  eval/usr/bin/打印文件“%s”$(/usr/bin/printf$*|\
  使用：。*：{&}：；s：（[0-9]+)-([0-9]+)：{\1..\2}：g；s：^\{（[^，]*)\}$：\1：‘）；
}

#! 未使用的信息
函数获取填充节点(){本地节点列表
  本地节点

  如果[“${NodeList}”=”]，则节点=($(range_expand$(cat/sys/设备/系统/节点
  /has_cpu))
    用于节点中的节点\
        $(cd/sys/设备/系统/节点/；|节点*|”/节点[0-9]\{1，/s/节点//‘)
    做些什么
     节点列表=(${节点列表[*]}$(/usr/bin/echo$(($(/usr/bin/echo\
     “16i$(使用的/，//g；s/。*/\U&/‘/sys/设备/系统/节点/节点${节点}/cpumap)p“|\dc)==0？0：$节
     点））））
    已完成了
  Fi
  “${节点列表[*]}”
}

函数_HousekeepingCpus(){本地参数
  本地列表本地HT
  本地核心
  本地处理器的本地计
  数

#! “参数”得到线程执行核心并发处理器处理器参数=($(/usr/bin/lscpu|\
  /usr/bin/awk\
 ‘/(^线程(s)每个核心：|^核心(s)：|^[CPU]*[sS]套接字(s)：)/{print$NF}))未设置列表
  对于$中的HT(seq0$(($(Params[0]}-1)))；为
    ${Params[1]}做；
      对于$中的处理器(seq0$(($(Params[2]}-1)))；做
        用于$计数（seq0$（${家庭代码}-1））；do
List=(${List[*]]$(($(HT}*${Params[1]}*${Params[2]}+${Processors}*${Cores}+${Count})))已完成
      已完
    成的工
    作
  已完成了
  /usr/bin/printf“$(echo${列表[*]}|使用‘//，/g’)”
}

函数_NoBalanceCpus(){本地参数
  本地的列表
```

```
   本地HT本地核心
   本地处理器的本地计
   数

   参数=($(/usr/bin/lscpu|\
   /usr/bin/awk\
  '/(^线程(s)每个核心：|^核心(s)：|^[CPU]*[sS]套接字(s)：)/{print$NF}))未设置列表
   对于$中的HT(seq0$(($(Params[0]}-1)))；为
      ${Params[1]}做；
         对于$中的处理器(seq0$(($(Params[2]}-1)))；做
            为$计数（参见${家庭保持核心}${无平衡核心}）；do列表
=($(List[*]}$(($(HT}*${Params[1]}*${Params[2]}+${Processors}*${Cores}+${Count})))
            已完
         成的工
         作
      已完
   成的工
   作
   /usr/bin/printf“$(/usr/bin/echo${列表[*]}|使用‘//，/g’)”
}

函数_IsolatedCpus(){
   _NoBalanceCpus
}

函数_HousekeepingMask(){本地参数
   本地掩码=0本地
   HT本地核心
   本地处理器的本地计
   数

#!“参数”得到线程执行核心并发处理器处理器参数=($(/usr/bin/lscpu|\
   /usr/bin/awk\
  '/(^线程\(s)的|^Core：|^[CPU]*[sS]套接字：)/{打印$NF}))($0$(($(Params[0]}-1)))；
      为$的核心{Params[1]}；做
         对于$中的处理器(seq0$(($(Params[2]}-1)))；做
            为$计数(seq0$（($(家庭维护代码}-1))）；do面具=$((面具|=1<<($(HT}*${参数[1]}*${参数[2]}+${处
理器}*${Cores}+${Count})))
            已完
         成的工
         作
      已完
   成的工
   作
   /usr/bin/printf“%x“$掩码
}

函数_NoBalanceMask(){
   本地HKMask=0x$(_HousekeepingMask)
   本地AllMask=0x$((-1<<$(grep-c处理器/proc/cpuinfo)))本地掩码
   Mask=$((~$HKMask^$AllMask))
   /usr/bin/printf“%x“$掩码
}


函数_IsolatedMask(){
   _NoBalanceMask
}
```

```
函数_ksoftirqd(){
   本地掩码=$(_HousekeepingMask)
```

```
   /usr/bin/打印文件 '0：f：2：%s：ksoftirqd。*'$掩码
}

函数_rcub(){
   本地掩码=$(_HousekeepingMask)
   /usr/bin/打印文件 '0：f：4：%s：rcub。*'$掩码
}

#! 函数主(){
#! 这些是具有此名称的cpu的数量。                                    负数的意思是“除了”#! 根据编号：
   =1无=$(($(/usr/bin/lscpu|\
   /usr/bin/awk '/^Core\(s\)：/{print$NF}')-${家庭核心}))如果[“${定义功能”}！=“1”]；然
   后
      工作负载=_${1}轮
      班
      如果[“$(类型-t${工作负载})”=”函数”]，则
       ${工作负荷}$*fi
   Fi
 #! }
################################end##########################################
```

#chmod+x/etc/tuned/HPELowLatency/script.sh

请注意以下事项：

- 由于GRUB生成错误，“cmdline”中的第一个令牌被消耗和丢失。作为一个解决方案，添加单令牌选项“安静”作为第一个令牌。
- 内核启动参数“cmdline”使用tuned.conf文件实现。请在该部分中指定您所需的启动时间选项。
- 在重新启动服务器之前，请检查文件中的TUNED_BOOT_CMDLINE
  在应用调优后的配置文件后，进行调整后的启动线。
- 启动后，请查看重新启动时考虑了调整参数的/proc/cmdline。

4.3.2.1.3 *在/etc/调整后的目录下添加调整后的“cpu-partitioning-variables.conf”*

编辑现有的cpu-partitioning-variables.conf文件，并添加/修改以下条目，以反映孤立的CPU(s)。

#vi/etc/tuned/cpu-partitioning-variables.conf

注：放置以下脚本（请注意任何行包装问题）

```
#############################start#########################################
#! 此文件包含在#/usr/lib/tuned/cpu-partitioning/tuned.conf的[变
量]部分中
#
housekeeping_cpus=${f：exec：/etc/tuned/HPELowLatency/script.sh：
HousekeepingCpus}isolated_cores=${f：cpulist_invert：
${housekeeping_cpus}}no_balance_cores=${f：exec：/etc/tuned/HPELowLatency/script.sh：
NoBalanceCpus}
#############################end###########################################
```

4.3.3 正在准备和设置one-time.service脚本

在引导过程中执行BASH脚本，该脚本已为具有两个AMDEPYC高频处理器，如7F32 (8 Core)处理器的服务器定制，以设置所需的Linux内核参数。

4331 配置2 设置one-time.service和oneshot_script.sh

4.3.3.1.1 在系统/系统/目录下的“one-time.service”中添加系统服务设置

下面的命令将在VI编辑器中打开一个名为“one-time.service”的新服务文件，并添加以下配置设置

#SYSTEMD_EDITOR=/bin/vi/usr/bin/系统ctl编辑-力--完整的one-time.service

注：放置以下脚本（请注意任何行包装问题）

```
#############################start#########################################

[单位）
说明=执行此脚本以设置从CPU核心到网络After=network.target的低抖动环境

[服务]类型=类
型
ExecStart=/usr/local/bin/oneshot_script.sh时间开始
秒=0

[安装]
WantedBy=default.target

#############################end###########################################
```

4.3.3.1.2 在/usr/local/bin目录下添加/usr/local/bin/oneshot_script.sh

按照以下步骤完成重新配置并重新启动系统：

#vi/usr/local/bin/oneshot_script.sh

注：放置以下脚本（请注意任何行包装问题）

```
#! /bin/bash##########################################START##############################################
#
#脚本编写者
#西尔维斯特·拉贾斯卡兰，DEAE，AMD公司，            查克·纽曼，HPE#10月
30日第，2020
#
#
############################################################################################## #
#/usr/local/bin/oneshot_script.sh是一个环境设置脚本
#AMDEPYC基于高频处理器的服务器（例如）。7F32（2个套接字插座x8芯）。#这个自动脚本将帮助HFT/FSI域中
的客户进行观察
#抖动通过任何抖动观察程序，如“系统抖动“#”
##################################T          O   P   O   L   G   Y####################################
#示例：在此系统拓扑场景中，存在#(a) 2 套接字7F32（2套
ocketx8Cores）
#(b)512GBDRAM（内存），在BIOS中的NUMA拓扑设置为NPS=1#
####NUMA-系统################################################可用节点清单-H
#                   可用性：2个节点（0-1）
#                   节点0cpus：01 2 3 4 5 6 7
#                   节点0大小：257578MB
#                   节点0空闲：254656MB
#                   节点1cpus：8 9 10 11 12 13 14 15
#                   节点1大小：258043MB
#                   节点1空闲：255865MB
#                   节点的距离：
#                   节点     0    1
#                   0:   10   32
#                   1:   32   10
#
####NUMA-NUMA策略设置#数字数-s
#
#                   策略：默认值
#                   首选节点：当前节点
#                   系统：0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
#                   伙伴：       0 1
#                   产品编号：01
#                   模因绑定：   0 1
#
##################################S C R I P T             D E T A I L S#############################
#此脚本在启动期间作为系统d#(系统ctl{启用|启动|停止|状态}one-time.service)的
一部分执行
#   由下面的服务脚本调用以执行
#                   /etc/systemd/system/one-time.service
#                                       ==>/usr/local/bin/oneshot_script.sh
```

```
# ############################################################################################## #
#         参考文献：https://access.redhat.com/solutions/3152271#
#         例如：AMDEPYC7F32（8芯）双插座系统#
#                   插座1                          套接字字0
#                   0 0 0 0 0 0 0 0                000000000->默认情况下都允许
#                   1 1 1 1 1 1 1 0                1 1 1 1 1 1 1 0 -->
#                   (其中核心#0和8是专门为Linux操作系统和#提供的管家核心                    留下剩下的14个球
体作为孤立的球体
#                    观察结果）
#                   进入（11111110111111110）                    ===>合金碳硬币=“01011”#CPU十六进制
垫                          fe                          fe        ===>合金CPUMask=“fefe”#CPU核心列
表                          8                           0===>合金OSCPUCores=“0,8”#
# ############################################################################################## #
#
mpid=$$
LOGDIR=/opt/$(日期+%m%d%Y_%H%M%S)mkdir-
p${LOGDIR}
#
#+++重要说明+++
#示例：
#合金CPUMask=“fefe”
=“0,8“#合金碳碳
Cores=”0101”#
#变量分配通过调用/etc/tuned/HPELowLatency/script.shShell脚本，根据AMDEPYC处理器SKU#的类型自动分配
并将单个参数传递给下面相应的变量#       （合金CPUMask，合金Cores）
#

定义函数。只有=1；。/etc/tuned/HPELowLatency/script.sh合金
CPUMask=“$(_IsolatedMask)”$(_HousekeepingCpus)=“$(_HousekeepingMask)
”
“允许的CPUMASK（十六进制）：                    ${AlloWeDCPUMask}“回声-
e”允许操作系统的CPU代码：                       $允许操作系统(BIN)的CPU核
心：${循环OSBINCores}“#
#0.设置和读取较少抖动#所需的新Sysctl值
sysctl-p/etc/sysctl.d/100-lowlatency.conf#
#1.正在检查系统的调整配置文件是否设置为“HPELow低延迟“#”       其中包括“cpu分区”
echo“检查当前调谐的配置文件”
如果[“${货币配置文件}”！=“HPELow低延迟”]，则
    调优的adm配置文件HPELow低延迟
    调整-adm活动|tee-a${LOGDIR}/tuned_profile_active_${mpid}。
    logcat${LOGDIR}/tuned_profile_active_${mpid}。log
Fi
#
#2.设置对写回线程的亲和力。这些线程应该只移动到#        在我们的案例中，$
#
echo0>/sys、总线、工作队列、设备、写回/numa
```

```
cat、sys、总线、工作队列、设备、写回/numa
echo${合金OSBINCores}>/sys/总线/工作队列/设备/写回/cpu掩码
cat/sys、总线、工作队列/设备、写回/cpu掩码|tee-a${LOGDIR}/cpumask_${mpid}。log#
#3.将所有CPU核心调节器设置为性能NP=$(grep-c处理器/过程/cpu=
信息)
如果[-e/sys/devices/system/cpu/cpu0/cpufreq/scaling_governor]；则对于$中的
          sg(seq0$(($(NP}-1)))；do
              回声性能>/sys/devices/system/cpu/cpu${sg}/cpufreq/scaling_governor已完成
          $中的sg（seq0$（${NP}-1））；
              cat/sys/devices/system/cpu/cpu${sg}/cpufreq/scaling_governor做了|tee-
          a${LOGDIR}/cpu_scaling_gov_${mpid}。log
Fi
#
#4.关闭和打开核心（除了客房核心0和8）回声-e”它还需要几秒钟。请稍候。”
NP=$(grep-c处理器/程序/cpuinfo)
          对于$(seq1$(($(NP}-1)))$(seq$(($(NP}/2+1))$(($(NP}-1)))；做回波${OffOn}>/系统/设备/系
              统/cpu/cpu/cpu${Cpu}/在线
              睡眠1已
          完成
已完成了
如果[-e/sys/devices/system/cpu/cpu0/cpufreq/scaling_governor]；则对于$中的
          sg(seq1$(($(NP}-1)))；do
              cat/sys/devices/system/cpu/cpu${sg}/cpufreq/scaling_governor做了|tee-
          a${LOGDIR}/cpu_scaling_gov_${mpid}。log
Fi
#
#5.冲洗/清除不需要的IPV4/IPV6网络相关的内容
iptables，F，X，ip6表，X
模式探测器-rebtable_nat电子表
模针-ript_SYNPROXY nf_synproxy_core xt_CT nf_conntrack_ftp\nf_conntrack_tftp
               nf_conntrack_irc nf_nat_tftp ipt_MASQUERADE\iptable_nat nf_nat
               nf_conntrack_ipv4nf_nat\
               nf_conntrack_ipv6xt_state xt_conntrack iptable_raw\nf_conntrack iptable_filter
               iptable_raw iptable_mangle\ipt_REJECT xt_CHECKSUM ip_tables
               nf_defrag_ipv4ip6table_filter\ip6_tables nf_defrag_ipv6ipv6t_REJECT xt_LOG
               xt_multiport\nf_conntrack
#
#6.RCU（读取副本更新）只使用管家核心#我们已经在步骤1中将cpu分区配置文件设置
为
#排除(引导后的rcu任务，本例为Core0和8)，#${AllowedOSCPUCores}的管家cpus
#
为i在`pgreprcu`；完成任务集-pc${合金OSCPUCores}$i；完成#
#7.通过强制IRQ值使用${$OSBINCores}#来进行修改
在伊拉克，伊拉克，smp_affinity；做
          echo$}>$irq2>/dev/null完成了|tee-
a${LOGDIR}/IRQ_new_values_${mpid}。log
#cat${LOGDIR}/IRQ_new_values_${mpid}。log#
#8.修改kswapd以仅使用操作系统核心
对于我在`交换？`？任务集？我在`交换？`；做
    任务集-cp$i
完成了|tee-a${LOGDIR}/kswapd_new_values_${mpid}。log
```

```
#9.正在关闭不需要的系统管理服务
在中的服务          avahi-daemon.service bluetooth.service chronyd.service\crond.service dbus.service
                    dnsmasq.service dnsmasq.service\firewalld.service firewalld.service
                    iprdump.service iprinit.service\iprupdate.service kdump.service ksm.service
                    libstoragemgmt.service\libvirtd.service lvm2-monitor.service
                    mcelog.service\mdmonitor.service messagebus.service ModemManager.service\
                    nfs-client.target postfix.service rhnsd.service rhsmcertd.service\rpcbind.service
                    rpcbind.socket systemd-journald.service已调整了。服务\systemd-journald.socket
                    virtlogd.socket wpa_supplicant.service
做些什么
          系统停止$服务系统是-活动的$服务|\
                    三通-a${LOGDIR}/inactive_service_status_${mpid}。log
已完成的##########################################END##########################################
```

```
#chmod+x/etc/systemd/system/one-time.service
#chmod+x/usr/local/bin/oneshot_script.sh
#-重新加载系统ctl守护进程
#系统控制器将启用one-time.service
#系统系统列表-单元文件one-time.service-状态=已启用
单位文件              状态
one-time.service已启用
列出了1个单元文件。
```

```
#系统系统正在重新启动
```

当系统启动并返回时，one-time.service将调用oneshot_script.sh
并根据脚本实现所有所需的设置。

一旦系统恢复联机，以root身份登录，并检查用户PID是否只与每个处理器的第一个CPU关联。

```
#任务集-cp$$
pid7032当前的亲和力列表：0,8
```

4.3.4 验证默认启动内核到RHEL8（8.1）（示例）

```
#肮脏的默认内核
/boot/vmlinuz-4.18.0-147.5.1.el8_1.x86_64
```

4.3.5 RHEL8的Grubby信息（8.1)(示例）

```
#肮脏的——信息/boot/vmlinuz-4.18.0-147.5.1.el8_1.x86_64
索引=1
kernel=“/boot/vmlinuz-4.18.0-147.5.1.el8_1.x86_64”
args=“ro crashkernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap rhgb
quiet kernelopts=root=/dev/mapper/rhel-root ro crashkernel=auto resume=/dev/mapper/rhel-swap
rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap rhgb quiet$tuned_params quiet selinux=0mce=ignore_ce
ipv6.disable=1audit=0nmi_watchdog=0hugepagesz=2MB hugepages=6000default_hugepagesz=2MB
transparent_hugepage=never tsc=reliable pcie_aspm=off cpuidle.off=1rcu_nocb_poll idle=poll
processor.max_cstate=0nohz=on
nohz_full=1,2,3,4,5,6,7,9,10,11,12,13,14,15rcu_nocbs=1,2,3,4,5,6,7,9,10,11,12,13,14,15
tuned.non_isolcpus=00000101intel_pstate=禁用无软件锁定$tuned_params“根
=”/dev/映射器/流变-根”
initrd=“/boot/initramfs-4.18.0-147.5.1.el8_1.x86_64.img$tuned_initrd”
标题=“红帽企业Linux(4.18.0-147.5.1.el8_1.x86_64) 8.1
(Ootpa)” id=“16699b9231234d6f83e12e4549b18673-4.18.0-
147.5.1.el8_1.x86_64”
```

现在，请使用以下命令重新启动系统：

```
#系统系统正在重新启动
```

4.4 重新启动后，正在验证启动参数配置

4.4.1 启动程序的参数

重新启动后，检查正在运行的内核，以验证是否启用了所需的设置，包括HPEL低延迟配置文件设置的启动参数选项。

```
#cat/proc/cmdline
```

注意：此命令行信息显示了具有2个AMDEPYC高频处理器的服务器，如AMDEPYC7F32（每个8个核心），以及如何根据HPEL低延迟调优配置文件隔离内核。

```
BOOT_IMAGE=(hd1, gpt2)/vmlinuz-4.18.0-147.5.1.el8_1.x86_64root=/dev/mapper/rhel-root ro
crashkernel=auto resume=/dev/mapper/rhel-swap rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap rhgb quiet
selinux=0mce=ignore_ce ipv6.disable=1audit=0nmi_watchdog=0hugepagesz=2MB
hugepages=6000default_hugepagesz=2MB transparent_hugepage=never tsc=reliable pcie_aspm=off
cpuidle.off=1rcu_nocb_poll idle=poll
processor.max_cstate=0nohz=onnohz_full=1、2、3、4、5、6、7、9、10、11、12、13、14、14、15rcu_nocbs=1、
2、3、4、5、6、7、9、10、11、12、13、14、15tuned.non_isolcpus=00000101intel_pstate=禁用
```

Chapter 5 系统上的抖动

太阳能照明设备的系统抖动实用程序衡量了系统引入抖动的程度，从而对用户级过程的影响。系统抖动在每个处理器核心上运行一个线程，当线程从核心中断时，它会测量多长时间。造成这种抖动的一个常见原因是，当核心上安排另一个任务时，尽管如果抖动阈值设置得足够低，也可以检测到较短的事件，如缓存丢失事件。引起抖动的任务的例子是调度程序在核心上调度的其他用户进程，甚至调度程序本身也会引起抖动。

系统抖动会为每个处理器核心生成汇总统计信息。系统抖动实用程序可以从www.openonload.org

5.1 正在安装系统系统抖动

sys抖动实用程序可以从https://下载www.openonload.org/download/sysjitter/。在编写此文档时，系统抖动的最新版本是1.4版本。有关构建和运行系统抖动的详细信息，请参阅系统抖动自述文件。

例如，在我们的案例中，我们已经在/opt/sysjitthy-1.4目录上安装了系统抖动工具。

5.2 正在运行的系统系统抖动

安装系统抖动后，您需要创建下面的Bash脚本，并将其调用为/opt/run_sysjit.sh.请参见下面关于任何AMDEPYC高频处理器的系统抖动执行脚本。

#vi/opt/run_sysjit.sh
#chmod+x/opt/run_sysjit.sh
注：放置以下脚本（请注意任何行包装问题）

```
#! /bin/bash
##############################start########################################
清除我的主机=$(主机名-s)
mydate=$(date+%Y%%m%d%H%M%%S%Z)回复
=/低延迟/${mydate}
“目标目录：${复制器}/${myhost}。${mydate}”mkdir-
p${repdir}jitfile=“${repdir}/sysjitter.${myhost}。${mydate}”
#
#
#函数来获得nobalanccpus的##
#变量分配基于CPU的类型自动分配，并通过#调用/etc/tuned/HPELowLatency/script.shBASH脚本获取核心
#以获取处理器的核心#
#不是管家核心(_NoBalanceCpus)参数，并将其传递给lim_cores变量
#
#示例：
#         在一个2套接字AMDEPYC7F32x8核心没有HT=16核心总# 操作系统管家核心分别为0和8
#         因此，观察到这些核心1、2、3、4、5、6、7、9、10、11、12、13、14、15#中的抖动
定义函数。只有=1；。/etc/tuned/HPELowLatency/script.sh
lim_cores=$(_NoBalanceCpus)
这些是核心列表将被系统抖动\n$\lim_cores}\n“#
#收集抖动器的数据#
回声-“”
“抖动数据收集已经开始使用/opt/run_sysjit.sh脚本“eche-e”使用系统抖动程序的
Cores${lim_cores}”
echo-e“与${1}${2}seconds...的阈值”echo-e“”
请稍候。”请读a
时间/opt/sysjitter-1.4/sysjitter\
          ——原始${jitfile}--运行时${2}--核心${lim_cores}${1}>${jitfile}。txt列-t${jitfile}。txt|tee-
a${jitfile}。tab
回声-“”
回声-“”
“抖动数据收集现在已完成”
“使用系统抖动为Cores${lim_cores}与${1}阈值为${2}seconds...”
“Cores${lim_cores}的抖动数据收集现在已经完成”
回声-e“与${1}阈值为${2}seconds...\n检查文件在${jitfile}。tab文件”echo-e“”
回声-“”
##########################################end##############################################
```

5.2.1 准备和检查

在运行系统抖动之前，请确保您的系统是否健康，并准备好进行抖动观察。

- 负载平均值接近0.00（例如）。请参见下面的内容，并且可以很好地执行该脚本

```
#正常运行时间
 提前46天，                    1:04,    2个用户，    平均负荷：0.00、0.00、0.00

它不应该像下面的那样（当系统负载接近0.00时再等待几分钟）

#正常运行时间
 提前46天，                    1:24,    2个用户，    平均负荷：7.22、2.61、0.93
```

- 检查并确认您的系统是否设置为“高极低延迟”调整配置文件，并且您运行系统抖动程序的终端/控制台当前只设置为管家核心。

```
#调-Adm激活
似乎调优的守护进程没有运行，预设配置文件没有被激活。预设配置文件：HPELow延迟
注：
•       上面的消息显示，系统的调整配置文件现在被设置为HPELow低延迟。
•       有意，调整后的后台进程没有运行，因为我们在步骤9中通过启动Bash脚本(oneshot_script.sh)关闭了系统级后台进程
#任务集-cp$$
pid16382当前的亲和力列表：0,8
```

5.3 启动系统异常抖动

作为根用户，请转到Sys抖动目录/opt/sysjitter-1.4.您在哪里创建了Bash，脚本称为(run_sysjit.sh)

#cd/opt/系统抖动-1.4#。

/run_sysjit.sh100610

其中，

1. 常数100（首选） 是(ns)中的阈值，请忽略短于此期间的任何中断。
2. 变量610 #秒（10分钟加10秒，每分钟增加1秒）。

当程序完成执行时，它将生成一个CSV文件作为输出到一个目录中，并通过该脚本创建当前的日期和时间。

5.4 来自系统抖动输出的抖动分析

5.4.1 AMDEPYC7F32（2套接字）

以下是单个核心统计文件和格式化的选项卡文件的示例输出：

```
rwrr1根根                386年6月22日17：33sysjitter.hpe-lowlat.20200622172324PDT.15
rwrr1根根                 6月26日52日17：33sysjitter.hpe-lowlat.20200622172324PDT.14
rwrr1根根                388年6月22日17：33sysjitter.hpe-lowlat.20200622172324PDT.13
rwrr1根根                 6月26日52日17：33sysjitter.hpe-lowlat.20200622172324PDT.12
rwrr1根根                 6月26日52日17：33sysjitter.hpe-lowlat.20200622172324PDT.11
rwrr1根根                386年6月22日17：33sysjitter.hpe-lowlat.20200622172324PDT.10
rwrr1根根                386年6月22日17：33sysjitter.hpe-lowlat.20200622172324PDT.09
rwrr1根根                 6月26日52日17：33sysjitter.hpe-lowlat.20200622172324PDT.07
rwrr1根根                 6月26日52日17：33sysjitter.hpe-lowlat.20200622172324PDT.06
rwrr1根根                539年6月22日17：33sysjitter.hpe-lowlat.20200622172324PDT.05
rwrr1根根                 6月26日52日17：33sysjitter.hpe-lowlat.20200622172324PDT.04
rwrr1根根                 6月26日52日17：33sysjitter.hpe-lowlat.20200622172324PDT.03
rwrr1根根                386年6月22日17：33sysjitter.hpe-lowlat.20200622172324PDT.02
rwrr1根根                386年6月22日17：33sysjitter.hpe-lowlat.20200622172324PDT.01
rwrr1根根               3669年6月22日17：33sysjitter.hpe-lowlat.20200622172324PDT.tab
```

下面是一个格式化后的输出示例，以便更容易实现可读性。这里的输出选项卡文件sysjitter.hpe-lowlat.20200622172324PDT.tab由Linux命令(列t)格式化。

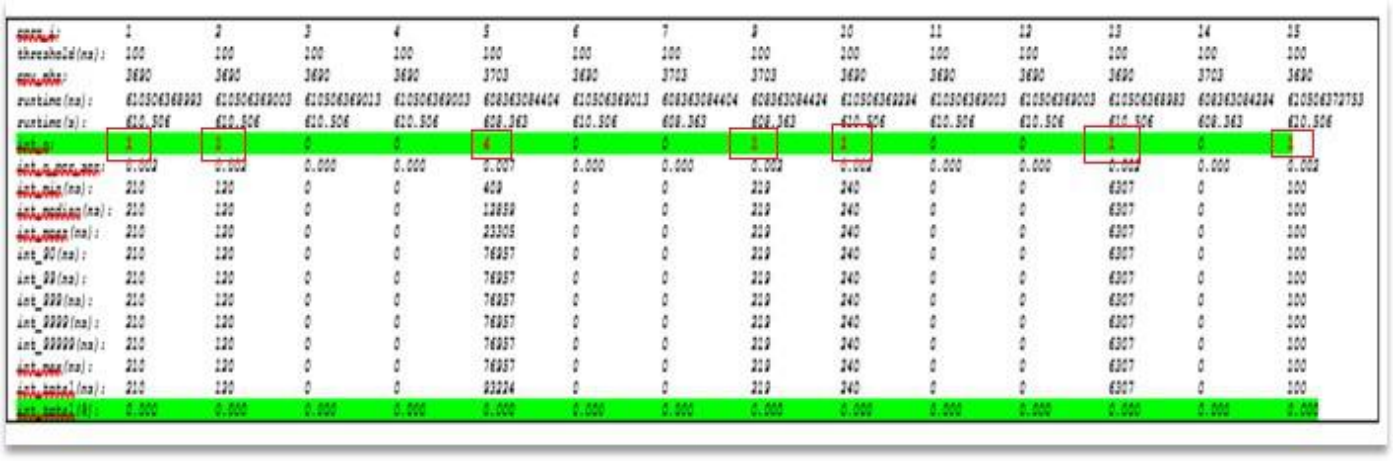


图2系统系统抖动的结果

5.5 快速的分析

从上面格式化的选项卡文件中，sysjitter.hpe-lowlat.20200622172324PDT.tab输出显示了与在610秒运行期间，在2个AMDEPYC7F32处理器（1,2,3,4,5,6,7,9,10,11,12,13,14,15）上的14个隔离核上的系统抖动所观察到的中断相关的各种统计数据。在本示例中，在10分钟的运行过程中，总共遇到了10个中断，在图2中用红色矩形标记。强烈建议您还仔细阅读每个核心的系统抖动输出文件，该文件显示系统抖动执行时间窗口期间抖动事件的时间历史。

这些结果非常好，表明BIOS、RHELOS和调谐设置/调谐减少了抖动，这证明了这个HPE原型DL385Gen10和两个AMDEPYC7F32处理器的服务器是一个调谐良好的服务器，抖动可以忽略不计。因此，对于金融服务行业的高频交易(HFT)次级市场，该服务器是低延迟工作负载的绝佳选择，这对计算机硬件的要求非常高。在构建这些解决方案时必须非常小心，以确保部署按需要执行。惠普企业与行业领先的合作伙伴密切合作，拥有配置和营销服务器的悠久历史，使许多世界领先的金融机构能够应对这一挑战。

5.6 更深入的分析

如果您决定对此低延迟性能优化进行更深入的分析，请联系您的AMD或HPE代表以获得进一步的帮助。