



浪潮信息云峦 KeyarchOS 基于 RDT 数据库保护最佳实践

浪潮电子信息产业股份有限公司

2023 年 10 月

目 录

1 测试概述	1
1.1 应用场景	1
1.2 技术背景	1
1.3 测试内容	1
1.4 术语解释	1
2 软硬件环境	2
2.1 硬件	2
2.2 软件	2
2.3 测试工具	3
2.4 技术架构	3
3 测试指导	4
3.1 BIOS 配置及内核配置	4
3.2 服务组件配置设计	5
3.3 服务组件架构设计	5
3.4 RDT 特性技术验证环境搭建	6
4 测试用例及测试数据	8
4.1 测试用例	8
4.1.1 内存延迟测试	错误！未定义书签。
4.1.2 基于 Docker 运行时 CRI-RM 插件 RDT MBA 特性验证	8
4.2 测试数据	16
5 分析与结论	17

6 可能存在的问题与解决方式.....	17
7 附录.....	17

1 测试概述

1.1 应用场景

在今天的云数据中心环境中，存在许多共享资源（如 L3 Cache、内存带宽等）。虽然通过共享资源提供了良好的性能、可扩展性和吞吐量，但某些应用程序（如后台视频流或转码应用程序）可能会过度利用这些共享资源，从而降低关键工作负载的性能。

1.2 技术背景

针对上述问题，Intel 推出了一种名为 **RDT（资源调配技术，Resource Director Technology）** 的硬件技术。该技术旨在通过一系列的 CPU 指令从而允许用户直接对每个 CPU 核心的 L3 缓存（LLC）以及内存带宽进行监控和分配，有助于提高整个数据中心的效率和灵活性，同时降低总体拥有成本。

1.3 测试内容

本文档是针对云平台场景下，针对内存访问延迟和数据库性能，使用 RDT 保护内存访问速度和 redis 性能的测试总结，主要测试内容包括，RDT 对内存访问速度的保护，RDT 对 redis 数据库性能的保护，设计到的测试用例总计 2 条。

1.4 术语解释

名词	描述
RDT	Intel 资源调配技术
ops/sec	吞吐量

2 软硬件环境

2.1 硬件

CPU 型号	配置
6348	112 核

内存大小
500G

测试节点 NUMA 资源信息：

节点 NUMA 资源信息
node0 cpus: 0-27,56-83 node0 size: 257369MB node0 free: 1080MB node1 cpus: 28-55,84-111 node1 size: 257996MB node1 free: 69MB

2.2 软件

配置项	说明
InCloudOS	V6. 8. 0
Kubernetes	1. 26. 2
CRI-RM	0. 8. 1
Docker	20. 10. 23

OS	Inspur K-UX 5.6. SP1
----	----------------------

2.3 测试工具

测试工具	版本	测试内容	备注
Benchmark		Redis 性能测试	

2.4 技术架构

当前 Keyarhos 内核完全支持 Intel RDT 特性。为了支持 RDT 功能，Intel 增加了新的寄存器 MSR 用于资源的监控和控制；内核架构实现了相应的 MSR 寄存器操作，如功能枚举、资源监控和分配、CLOS/RMID 与线程和 CPU 核心关联等。这些功能均通过文件系统展现到用户空间。最终从用户的角度来看，Intel RDT 的监控和分配功能是通过默认装载在 `/sys/fs/resctrl` 下的资源控制文件系统来实现的。

Kernel 提供了 `resctrl` 的接口，通过 `resctrl` 在 `sysfs` 下的节点，用户可以方便的去控制 RDT 的 feature。

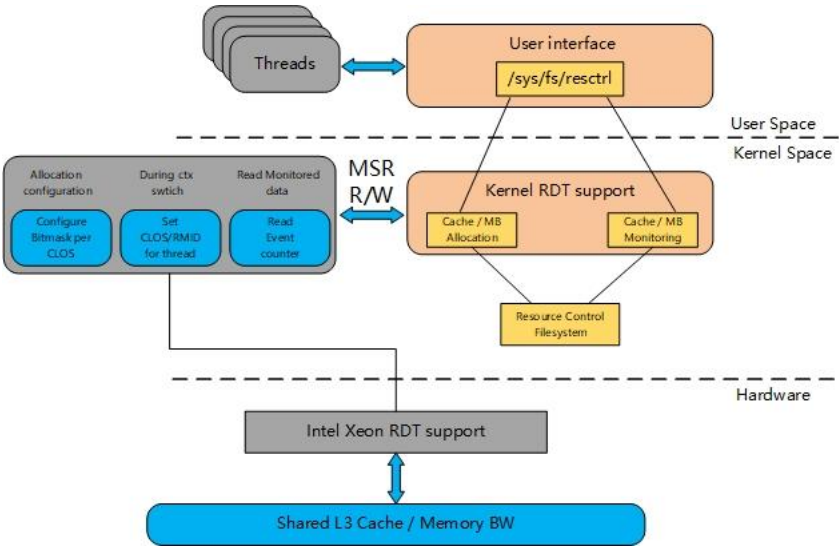
英特尔® RDT 在 `resctrl` 文件系统下的分层结构类似于控制组 (Cgroup)。与 Cgroup 相比，`resctrl` 文件系统界面有着类似的进程管理生命周期和用户界面。但不同于 Cgroup 的分层结构，`resctrl` 文件系统界面是单层文件系统结构。

资源组在 `resctrl` 文件系统中表示为目录。默认组为根目录，在文件系统装载后即拥有系统中的全部任务和 CPU，可以充分使用所有资源。‘info’ 目录包含已启用的资源的信息。

启用 RDT 控制后，可在根目录中创建用户目录（“CG1” 和 “CG2”，见下图：英特尔® RDT 在 `resctrl` 文件系统下的分层结构），为每个共享资源指定不同的控制力度。RDT 控制组包含以下文件：“tasks”：读取该文件会显示该群组所有任务的列表。将任务 ID 写入文件会添加任务到群组。“cpus”：读取该文件会

显示该群组拥有的逻辑 CPU 的位掩码。将掩码写入文件会添加 CPU 到群组或从群组中移除 CPU。“schemata”：该群组可访问的所有资源的列表。

启用 RDT 监控功能后，根目录和其他顶层目录会包含“mon_groups”目录，在此目录中可以创建用户目录（“M1”和“M2”，见下图：英特尔® RDT 在 resctrl 文件系统中的分层结构），以监控任务群组。“Mon_data”目录包含一组按照资源域和 RDT 事件组织的文件。这些目录中，每个目录针对每个事件都有一个文件（“llc_occupancy”、“mbm_total_bytes”和“mbm_local_bytes”）。这些文件为群组中的所有任务提供了事件当前值的计数器。



3 测试指导

3.1 BIOS 配置及内核配置

内核要开启 intel RDT 特性，需要打开配置 CONFIG_X86_CPU_RESCTRL 。要判断当前 cpu 是否支持 intel RDT 特性，可以通过 /proc/cpuinfo 文件查看：

RDT (Resource Director Technology) Allocation	“rdt_a”
CAT (Cache Allocation Technology)	“cat_l3”, “cat_l2”
CDP (Code and Data Prioritization)	“cdp_l3”, “cdp_l2”
MBM (Memory Bandwidth Monitoring)	“cqm_mbm_total”,

	“cqm_mbm_local”
MBA (Memory Bandwidth Allocation)	“mba”

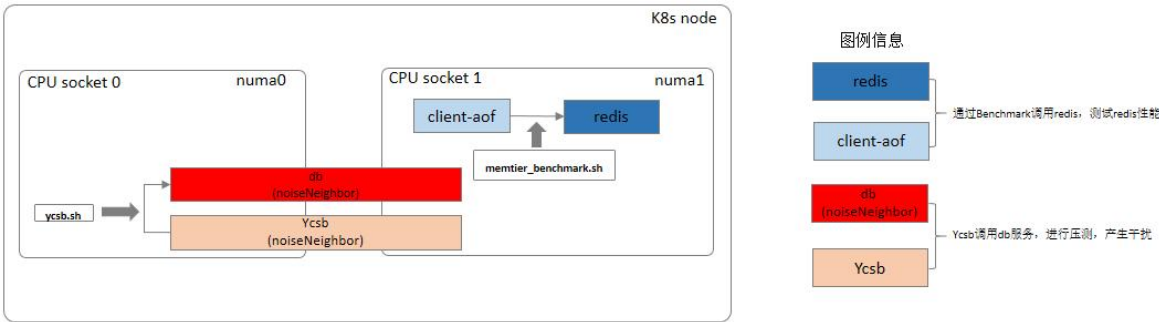
3.2 服务组件配置设计

服务名称	配置信息	Pod 数量
redis	2 核 32Gi	1
client-aof	2 核 8Gi	1
db	requests: 300m 3Gi	88
ycsb	requests: 100m	88

说明：

测试RDT特性主要涉及两对服务，一个是通过 Benchmark 调用 redis, 测试 redis 性能；另一个是通过 Ycsb 调用 db 服务，进行压测，产生干扰。服务组件对应的配置及 Pod 数量如上表格所述。Redis 和 client-of 服务保证是 Pod 的 QOS 为 Guaranteed，Ycsb 和 db 服务保证是 Pod 的 QOS 为 Burstable。

3.3 服务组件架构设计



说明：

对 redis 和 client-of 服务通过 CRI-RM 拓扑感知策略进行优先绑核操作；将 db 和 ycsb 服务根据 CRI-RM 拓扑感知策略使用其他剩余的 CPU 资源。

通过 ycsb.sh 脚本发起 ycsb 调用 db 服务，起到 noiseNeighbor 的作用；通过 memtier_benchmark.sh 脚本，发起 client-aof 调用 redis 服务，模拟 redis 正常读写操作。



ycsb.sh



memtier_benchmark.sh
ark.sh

3.4 RDT 特性技术验证环境搭建

准备 cri-resource-manager-0.8.1-0.centos-7.x86_64.rpm，执行如下命令：

```
#通过 rpm 命令安装部署包

[root@node1 /]# rpm -ivh cri-resource-manager-0.8.1-0.centos-7.x86_64.rpm

[root@node1 /]# cp /etc/cri-resmgr/fallback.cfg.sample /etc/cri-resmgr/fallback.cfg

# Change the config file(/etc/sysconfig/cri-resource-manager)

CONFIG_OPTIONS="--fallback-config /etc/cri-resmgr/fallback.cfg --runtime-socket=/var/run/cri-dockerd.sock
--image-socket=/var/run/cri-dockerd.sock --allow-untested-runtimes"

#开启 cri-resource-manager

[root@node1 /]# systemctl enable cri-resource-manager

[root@node1 /]# systemctl daemon-reload

[root@node1 /]# systemctl start cri-resource-manager

[root@node1 /]# systemctl status cri-resource-manager
```

设置 RDT 相关配置及开启 RDT

```
#修改 /etc/cri-resmgr/fallback.cfg
```

```
[root@node1 /]# vi /etc/cri-resmgr/fallback.cfg
```



```
#挂载 rdt 文件 resctrl
```

```
[root@node1 /]# mount -t resctrl resctrl /sys/fs/resctrl
```

```
#重启 cri-resource-manager
```

```
[root@node1 /]# systemctl restart cri-resource-manager
```

```
[root@node1 /]# systemctl status cri-resource-manager
```

验证 RDT 环境配置成功

#验证 RDT 开启成功，/sys/fs/resctrl 目录下会生成 cri-resmgr.BestEffort cri-resmgr.Burstable cri-resmgr.Guaranteed 三个关键文件夹

```
[root@node1 /]# ls /sys/fs/resctrl
```

```
cpus cpus_list cri-resmgr.BestEffort cri-resmgr.Burstable cri-resmgr.Guaranteed info mode mon_data mon_groups  
schemata size tasks
```

```
#验证配置生效
```

```
[root@node1 /]# cat /sys/fs/resctrl/cri-resmgr.Guaranteed/schemata
```

```
L3:0=7ff;1=7ff
```

```
MB:0=100;1=100
```

```
[root@node1 /]# cat /sys/fs/resctrl/cri-resmgr.Burstable/schemata
```

```
L3:0=800;1=800
```

```
MB:0= 10;1= 10
```

补充：还可以通过配置的方式动态加载 RDT 相关配置，需要准备 cri-resmgr-agent 镜像，替换镜像后参照如下 yaml 文件进行服务部署：



agent-deployme
nt.yaml

RDT 动态加载配置的 yaml 文件如下：



cri-resmgr-config
map.example.yamr

4 测试用例及测试数据

4.1 测试用例

4.1.1 基于 Docker 运行时 CRI-RM 插件 RDT MBA 特性验证

4.1.1.1 测试脚本



all_run_cri_rm.sh



run.sh

4.1.1.2 Redis 正常服务性能

测试类型	RDT redis性能测试	测试工具	CRI-RM插件、
测试目的	验证Redis正常服务性能		
前提条件			
1.Redis、client-aof服务为整数核心，生成Pod的QOS为Guaranteed。			
2.db、ycsb服务生成Pod的QOS为Burstable。			
3.都使用CRI-RM插件自带topology aware策略进行CPU分配操作。			
测试过程			
1.通过执行yaml文件，依次部署redis服务、client-aof服务、部署db服务（88个副本）、部署ycsb服务（88个副本）。			
2.执行all_run_cri_rm.sh脚本，除去第一次执行的结果，连续执行三次。			

<p>预期目标</p> <p>Redis服务平稳运行，性能测试数据稳定。</p>	
<p>测试结果：<input type="checkbox"/> 通过 <input type="checkbox"/> 不通过</p>	
备注	<p>在测试节点执行命令：<code>./all_run_cri_rm.sh</code></p> <p>注：<code>all_run_cri_rm.sh</code> 是提前准备好的脚本，通过 <code>client-aof</code> 容器直接调用 <code>redis</code> 服务，模拟 <code>redis</code> 服务的正常运行。</p>

4.1.1.3 88 个 NoiseNeighbor 服务

启动 NoiseNeighbor 未开启 RDT

测试类型	RDT redis性能测试	测试工具	CRI-RM插件、
测试目的	验证未开启 RDT，启动 NoiseNeighbor 干扰的情况下，Redis 服务性能		
前提条件			
1.Redis、client-aof服务为整数核心，生成Pod的QOS为Guaranteed。			
2.db、ycsb服务生成Pod的QOS为Burstable。			
3.都使用CRI-RM插件自带topology aware策略进行CPU分配操作。			
测试过程			
1.执行./run.sh 88 0 load a模拟 NoiseNeighbor 干扰 Redis 服务。			
2.执行./all_run_cri_rm.sh脚本，除去第一次执行的结果，根据观察 NoiseNeighbor 干预的时长，可多次执行脚本。			
预期目标			
Redis服务性能有所下降。			
测试结果： <input type="checkbox"/> 通过 <input type="checkbox"/> 不通过			
备注	在测试节点执行命令：./run.sh 88 0 load a和./all_run_cri_rm.sh 注：run.sh是提前准备好的脚本，通过 ycsb 容器直接调用 db 容器，模拟压力起到性能干扰作用。 all_run_cri_rm.sh是提前准备好的脚本，通过 client-aof 容器直接调用 redis 服务，模拟 redis 服务的正常运行。		

启动 NoiseNeighbor 开启 RDT

测试类型	RDT redis性能测试	测试工具	CRI-RM插件、
测试目的	验证开启 RDT，启动 NoiseNeighbor 干扰的情况下，Redis 服务性能		
前提条件			
1.Redis、client-aof服务为整数核心，生成Pod的QOS为Guaranteed。			
2.db、ycsb服务生成Pod的QOS为Burstable。			
3.都使用CRI-RM插件自带topology aware策略进行CPU分配操作。			
4.开启RDT特性后测试			
测试过程			
2. 执行./run.sh 88 0 load a 模拟 NoiseNeighbor 干扰 Redis 服务。			
2. 执行./all_run_cri_rm.sh 脚本，除去第一次执行的结果，根据观察 NoiseNeighbor 干预的时长，可多次执行脚本。			
预期目标			
开启RDT特性后，在有NoiseNeighbor干扰Redis服务的情况下，Redis服务的性能不受影响。			
测试结果： <input type="checkbox"/> 通过 <input type="checkbox"/> 不通过			
备注	在测试节点执行命令： ./run.sh 88 0 load a 和./all_run_cri_rm.sh 注：run.sh 是提前准备好的脚本，通过 ycsb 容器直接调用 db 容器，模拟压力起到性能干扰作用。 all_run_cri_rm.sh 是提前准备好的脚本，通过 client-aof 容器直接调用 redis 服务，模拟 redis 服务的正常运行。		

4.1.1.4 77 个 NoiseNeighbor 服务

启动 NoiseNeighbor 未开启 RDT

测试类型	RDT redis性能测试	测试工具	CRI-RM插件、
测试目的	验证未开启 RDT，启动 NoiseNeighbor 干扰的情况下，Redis 服务性能		
前提条件			
1.Redis、client-aof服务为整数核心，生成Pod的QOS为Guaranteed。			
2.db、ycsb服务生成Pod的QOS为Burstable。			

3. 都使用CRI-RM插件自带topology aware策略进行CPU分配操作。	
<p>测试过程</p> <p>3. 执行./run.sh 77 0 load a 模拟 NoiseNeighbor 干扰 Redis 服务。</p> <p>2. 执行./all_run_cri_rm.sh 脚本，除去第一次执行的结果，根据观察 NoiseNeighbor 干预的时长，可多次执行脚本。</p>	
<p>预期目标</p> <p>Redis服务性能有所下降。</p>	
测试结果： <input type="checkbox"/> 通过 <input type="checkbox"/> 不通过	
备注	<p>在测试节点执行命令：./run.sh 77 0 load a 和./all_run_cri_rm.sh</p> <p>注：run.sh 是提前准备好的脚本，通过 ycsb 容器直接调用 db 容器，模拟压力起到性能干扰作用。</p> <p>all_run_cri_rm.sh 是提前准备好的脚本，通过 client-aof 容器直接调用 redis 服务，模拟 redis 服务的正常运行。</p>

启动 NoiseNeighbor 并开启 RDT

测试类型	RDT redis性能测试	测试工具	CRI-RM插件、
测试目的	验证开启 RDT，启动 NoiseNeighbor 干扰的情况下，Redis 服务性能		
<p>前提条件</p> <p>1.Redis、client-aof服务为整数核心，生成Pod的QOS为Guaranteed。</p> <p>2.db、ycsb服务生成Pod的QOS为Burstable。</p> <p>3.都使用CRI-RM插件自带topology aware策略进行CPU分配操作。</p> <p>4.开启RDT特性后测试</p>			
<p>测试过程</p> <p>4. 执行./run.sh 77 0 load a 模拟 NoiseNeighbor 干扰 Redis 服务。</p> <p>2. 执行./all_run_cri_rm.sh 脚本，除去第一次执行的结果，根据观察 NoiseNeighbor 干预的时长，可多次执行脚本。</p>			
<p>预期目标</p> <p>开启RDT特性后，在有NoiseNeighbor干扰Redis服务的情况下，Redis服务的性能不受影响。</p>			

测试结果： <input type="checkbox"/> 通过 <input type="checkbox"/> 不通过	
备注	<p>在测试节点执行命令：./run.sh 77 0 load a 和./all_run_cri_rm.sh</p> <p>注：run.sh 是提前准备好的脚本，通过 ycsb 容器直接调用 db 容器，模拟压力起到性能干扰作用。</p> <p>all_run_cri_rm.sh 是提前准备好的脚本，通过 client-aof 容器直接调用 redis 服务，模拟 redis 服务的正常运行。</p>

4.1.1.5 66 个 NoiseNeighbor 服务

启动 NoiseNeighbor 未开启 RDT

测试类型	RDT redis性能测试	测试工具	CRI-RM插件、
测试目的	验证未开启 RDT，启动 NoiseNeighbor 干扰的情况下，Redis 服务性能		
前提条件			
1.Redis、client-aof服务为整数核心，生成Pod的QOS为Guaranteed。			
2.db、ycsb服务生成Pod的QOS为Burstable。			
3.都使用CRI-RM插件自带topology aware策略进行CPU分配操作。			
测试过程			
5. 执行./run.sh 66 0 load a 模拟 NoiseNeighbor 干扰 Redis 服务。			
2. 执行./all_run_cri_rm.sh 脚本，除去第一次执行的结果，根据观察			
NoiseNeighbor 干预的时长，可多次执行脚本。			
预期目标			
Redis服务性能有所下降。			
测试结果： <input type="checkbox"/> 通过 <input type="checkbox"/> 不通过			
备注	在测试节点执行命令：./run.sh 66 0 load a 和./all_run_cri_rm.sh 注：run.sh 是提前准备好的脚本，通过 ycsb 容器直接调用 db 容器，模拟压力起到性能干扰作用。 all_run_cri_rm.sh 是提前准备好的脚本，通过 client-aof 容器直接调用 redis 服务，模拟 redis 服务的正常运行。		

启动 NoiseNeighbor 并开启 RDT

测试类型	RDT redis性能测试	测试工具	CRI-RM插件、
测试目的	验证开启 RDT，启动 NoiseNeighbor 干扰的情况下，Redis 服务性能		
前提条件			
1.Redis、client-aof服务为整数核心，生成Pod的QOS为Guaranteed。			
2.db、ycsb服务生成Pod的QOS为Burstable。			
3.都使用CRI-RM插件自带topology aware策略进行CPU分配操作。			
4.开启RDT特性后测试			
测试过程			
6. 执行./run.sh 66 0 load a 模拟 NoiseNeighbor 干扰 Redis 服务。			
2. 执行./all_run_cri_rm.sh 脚本，除去第一次执行的结果，根据观察 NoiseNeighbor 干预的时长，可多次执行脚本。			
预期目标			
开启RDT特性后，在有NoiseNeighbor干扰Redis服务的情况下，Redis服务的性能不受影响。			
测试结果： <input type="checkbox"/> 通过 <input type="checkbox"/> 不通过			
备注	在测试节点执行命令： ./run.sh 66 0 load a 和./all_run_cri_rm.sh 注：run.sh 是提前准备好的脚本，通过 ycsb 容器直接调用 db 容器，模拟压力起到性能干扰作用。 all_run_cri_rm.sh 是提前准备好的脚本，通过 client-aof 容器直接调用 redis 服务，模拟 redis 服务的正常运行。		

4.1.1.6 55 个 NoiseNeighbor 服务

启动 NoiseNeighbor 未开启 RDT

测试类型	RDT redis性能测试	测试工具	CRI-RM插件、
测试目的	验证未开启 RDT，启动 NoiseNeighbor 干扰的情况下，Redis 服务性能		
前提条件			
1.Redis、client-aof服务为整数核心，生成Pod的QOS为Guaranteed。			
2.db、ycsb服务生成Pod的QOS为Burstable。			
3.都使用CRI-RM插件自带topology aware策略进行CPU分配操作。			

<p>测试过程</p> <p>7. 执行./run.sh 55 0 load a 模拟 NoiseNeighbor 干扰 Redis 服务。</p> <p>2. 执行./all_run_cri_rm.sh 脚本，除去第一次执行的结果，根据观察 NoiseNeighbor 干预的时长，可多次执行脚本。</p>	
<p>预期目标</p> <p>Redis服务性能有所下降。</p>	
<p>测试结果：<input type="checkbox"/> 通过 <input type="checkbox"/> 不通过</p>	
备注	<p>在测试节点执行命令：./run.sh 55 0 load a 和./all_run_cri_rm.sh</p> <p>注：run.sh 是提前准备好的脚本，通过 ycsb 容器直接调用 db 容器，模拟压力起到性能干扰作用。</p> <p>all_run_cri_rm.sh 是提前准备好的脚本，通过 client-aof 容器直接调用 redis 服务，模拟 redis 服务的正常运行。</p>

启动 NoiseNeighbor 并开启 RDT

测试类型	RDT redis性能测试	测试工具	CRI-RM插件、
测试目的	验证开启 RDT，启动 NoiseNeighbor 干扰的情况下，Redis 服务性能		
前提条件			
1.Redis、client-aof服务为整数核心，生成Pod的QOS为Guaranteed。			
2.db、ycsb服务生成Pod的QOS为Burstable。			
3.都使用CRI-RM插件自带topology aware策略进行CPU分配操作。			
4.开启RDT特性后测试			
测试过程			
8. 执行./run.sh 55 0 load a 模拟 NoiseNeighbor 干扰 Redis 服务。			
2. 执行./all_run_cri_rm.sh 脚本，除去第一次执行的结果，根据观察 NoiseNeighbor 干预的时长，可多次执行脚本。			
预期目标			
开启RDT特性后，在有NoiseNeighbor干扰Redis服务的情况下，Redis服务的性能不受影响。			
测试结果： <input type="checkbox"/> 通过 <input type="checkbox"/> 不通过			
备注	在测试节点执行命令： ./run.sh 55 0 load a 和./all_run_cri_rm.sh 注：run.sh 是提前准备好的脚本，通过 ycsb 容器直接调用 db 容器，模拟压力起到性能干扰作用。		

	all_run_cri_rm.sh 是提前准备好的脚本，通过 client-aof 容器直接调用 redis 服务，模拟 redis 服务的正常运行。
--	---

4.1.1.7 44 个 NoiseNeighbor 服务

启动 NoiseNeighbor 未开启 RDT

测试类型	RDT redis性能测试	测试工具	CRI-RM插件、
测试目的	验证未开启 RDT，启动 NoiseNeighbor 干扰的情况下，Redis 服务性能		
前提条件			
1. Redis、client-aof服务为整数核心，生成Pod的QOS为Guaranteed。			
2. db、ycsb服务生成Pod的QOS为Burstable。			
3. 都使用CRI-RM插件自带topology aware策略进行CPU分配操作。			
测试过程			
9. 执行./run.sh 44 0 load a 模拟 NoiseNeighbor 干扰 Redis 服务。			
2. 执行./all_run_cri_rm.sh 脚本，除去第一次执行的结果，根据观察 NoiseNeighbor 干预的时长，可多次执行脚本。			
预期目标			
Redis服务性能有所下降。			
测试结果： <input type="checkbox"/> 通过 <input type="checkbox"/> 不通过			
备注	在测试节点执行命令： ./run.sh 44 0 load a 和./all_run_cri_rm.sh 注： run.sh 是提前准备好的脚本，通过 ycsb 容器直接调用 db 容器，模拟压力起到性能干扰作用。 all_run_cri_rm.sh 是提前准备好的脚本，通过 client-aof 容器直接调用 redis 服务，模拟 redis 服务的正常运行。		

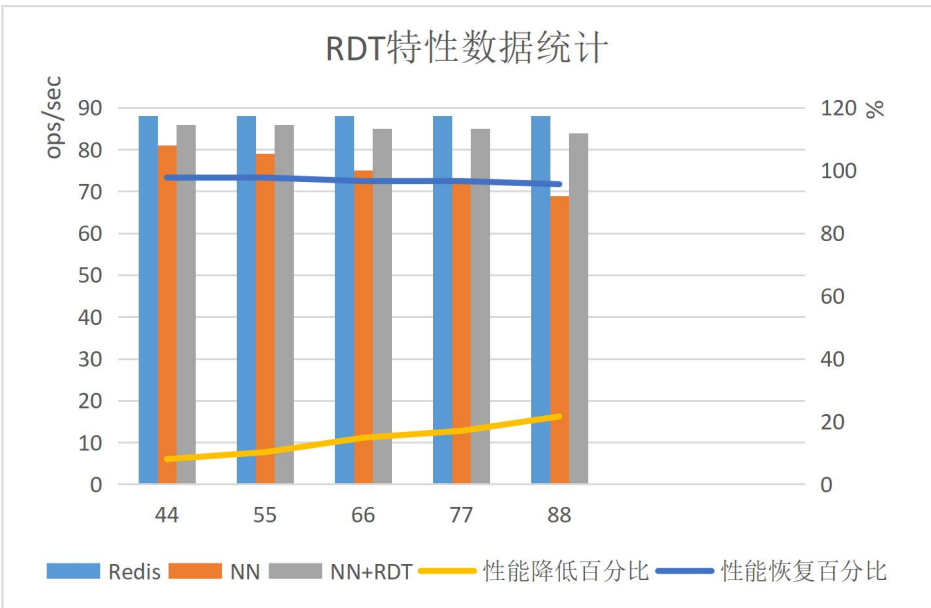
启动 NoiseNeighbor 并开启 RDT

测试类型	RDT redis性能测试	测试工具	CRI-RM插件、
测试目的	验证开启 RDT，启动 NoiseNeighbor 干扰的情况下，Redis 服务性能		

<p>前提条件</p> <ol style="list-style-type: none">1. Redis、client-aof服务为整数核心，生成Pod的QOS为Guaranteed。2. db、ycsb服务生成Pod的QOS为Burstable。3. 都使用CRI-RM插件自带topology aware策略进行CPU分配操作。4. 开启RDT特性后测试	
<p>测试过程</p> <ol style="list-style-type: none">10. 执行./run.sh 44 0 load a 模拟 NoiseNeighbor 干扰 Redis 服务。2. 执行./all_run_cri_rm.sh 脚本，除去第一次执行的结果，根据观察 NoiseNeighbor 干预的时长，可多次执行脚本。	
<p>预期目标</p> <p>开启RDT特性后，在有NoiseNeighbor干扰Redis服务的情况下，Redis服务的性能不受影响。</p>	
<p>测试结果：<input type="checkbox"/> 通过 <input type="checkbox"/> 不通过</p>	
备注	<p>在测试节点执行命令：./run.sh 44 0 load a 和./all_run_cri_rm.sh</p> <p>注：run.sh 是提前准备好的脚本，通过 ycsb 容器直接调用 db 容器，模拟压力起到性能干扰作用。</p> <p>all_run_cri_rm.sh 是提前准备好的脚本，通过 client-aof 容器直接调用 redis 服务，模拟 redis 服务的正常运行。</p>

4.2 测试数据

测试数据 1：RDT 保护数据库性能测试



5 分析与结论

综合上述测试数据来看，开启 RDT 特性后，能够减少 NoiseNeighbor 对 Redis 服务和内存访问延迟的影响，使用 RDT 后，Redis 性能和内存访问延迟基本能恢复到原来的水平。存在 NoiseNeighbor 干扰的情况下，没有开启 RDT 时，Redis 性能下降约 21.59%，内存访问延迟下降约 12.27%；开启 RDT 保护后，Redis 性能可恢复到原来的 95.45%，内存访问延迟可恢复到原来的 97.95%。同时，RDT 对应用的保护能力几乎不受 NoiseNeighbor 数量的干扰。

6 可能存在的问题与解决方式

无

7 附录

无