

## OOM相关参数配置与原因排查

更新时间：2025-12-29 GMT+08:00

### OOM相关概念

OOM（Out Of Memory，简称OOM）指系统内存已用完，在linux系统中，如果内存用完会导致系统无法正常工作，触发系统panic或者OOM killer。

OOM killer是linux内核的一个机制，该机制会监控那些占用内存过大的进程，尤其是短时间内消耗大量内存的进程，在系统的内存即将不够用时结束这些进程从而保障系统的整体可用性。

### OOM相关参数

表1 OOM相关参数

参数名称	参数说明	取值	修改方式
panic_on_oom	<p>panic_on_oom参数是控制系统遇到OOM时如何反应的。当系统遇到OOM的时候，通常会有两种选择：</p> <p>触发系统panic，可能会出现频繁宕机的情况。</p> <p>选择一个或者几个进程，触发OOM killer，结束选中的进程，释放内存，让系统保持整体可用。</p>	<p>可以通过以下命令查看参数取值：</p> <p><b>cat /proc/sys/vm/panic_on_oom</b>或者<b>sysctl -a   grep panic_on_oom</b></p> <p>值为0：内存不足时，触发OOM killer。</p> <p>值为1：内存不足时，根据具体情况可能发生kernel panic，也可能触发OOM killer。</p> <p>值为2：内存不足时，强制触发系统panic，导致系统重启。</p> <p><b>说明：</b></p> <p>HCE中参数默认值为0。</p>	<p>例如将参数设置为0，可用以下方式：</p> <p>临时配置，立即生效，但重启后失效。</p> <p><b>sysctl -w vm.panic_on_oom=0</b></p> <p>持久化配置，系统重启仍生效</p> <p>执行命令<b>vim /etc/sysctl.conf</b>，在该文件中添加一行<b>vm.panic_on_oom =0</b>，再执行<b>sysctl -p</b>或重启系统后生效</p>
oom_kill_allocating_task	<p>当系统选择触发OOM killer，试图结束某些进程时，oom_kill_allocating_task参数会控制选择哪些进程，有以下两种选择：</p> <p>触发OOM的进程。</p> <p>oom_score得分最高的进程。</p>	<p>可以通过以下命令查看参数取值：</p> <p><b>cat /proc/sys/vm/oom_kill_allocating_task</b>或者<b>sysctl -a   grep oom_kill_allocating_task</b></p> <p>值为0：选择oom_score得分最高的进程。</p> <p>值为非0：选择触发OOM的进程。</p> <p><b>说明：</b></p> <p>HCE中参数默认值为0。</p>	<p>例如将该参数设置成1，可用以下方式：</p> <p>临时配置，立即生效，但重启后失效。</p> <p><b>sysctl -w vm.oom_kill_allocating_task=1</b></p> <p>持久化配置，系统重启仍生效</p> <p>执行命令<b>vim /etc/sysctl.conf</b>，在该文件中添加一行<b>vm.oom_kill_allocating_task =1</b>，再执行命令<b>sysctl -p</b>或重启系统后生效。</p>

oom_score	<p>指进程的得分，主要由两部分组成：</p> <p>系统打分，主要是根据该进程的内存使用情况由系统自动计算。</p> <p>用户打分，也就是oom_score_adj，可以自定义。</p>	<p>可以通过调整oom_score_adj的值进而调整一个进程最终的得分。通过以下命令查看参数取值：</p> <p><b>cat /proc/进程id/oom_score_adj</b></p> <p>值为0：不调整oom_score。</p> <p>值为负值：在实际打分值上减去一个折扣。</p> <p>值为正值：增加该进程的oom_score。</p> <p><b>说明：</b></p> <p>oom_score_adj的取值范围是-1000 ~ 1000。</p> <p>若设定成OOM_SCORE_ADJ_MIN或-1000，则表示禁止OOM killer结束该进程。</p>	<p>例如将进程id为2939的进程oom_score_adj参数值设置为1000，可用以下命令：</p> <p><b>echo 1000 &gt; /proc/2939/oom_score_adj</b></p>
oom_dump_tasks	<p>oom_dump_tasks参数控制OOM发生时是否记录系统的进程信息和OOM killer信息。</p> <p>例如dump系统中所有的用户空间进程关于内存方面的一些信息，包括：进程标识信息、该进程使用的内存信息、该进程的页表信息等，这些信息有助于了解出现OOM的原因。</p>	<p>可以通过以下命令查看参数取值：</p> <p><b>cat /proc/sys/vm/oom_dump_tasks</b>或者<b>sysctl -a   grep oom_dump_tasks</b></p> <p>值为0：OOM发生时不会打印相关信息。</p> <p>值为非0：以下三种情况会调用dump_tasks打印系统中所有task的内存状况。</p> <ul style="list-style-type: none"> <li>由于OOM导致kernel panic。</li> <li>没有找到需要结束的进程。</li> <li>找到进程并将其结束的时候。</li> </ul> <p><b>说明：</b></p> <p>HCE中参数默认值为1。</p>	<p>例如将该参数设置成0，可用以下方式：</p> <p>临时配置，立即生效，但重启后失效。</p> <p><b>sysctl -w vm.oom_dump_tasks=0</b></p> <p>持久化配置，系统重启仍生效</p> <p>执行命令<b>vim /etc/sysctl.conf</b>，在该文件中添加一行<b>vm.oom_dump_tasks=0</b>，再执行命令<b>sysctl -p</b>或重启系统后生效。</p>

## 触发OOM killer示例

1. 您可以参考表1设置HCE系统参数，示例配置如下：

```
[root@localhost ~]# cat /proc/sys/vm/panic_on_oom
0
[root@localhost ~]# cat /proc/sys/vm/oom_kill_allocating_task
0
[root@localhost ~]# cat /proc/sys/vm/oom_dump_tasks
1
```

panic\_on\_oom=0，表示在发生系统OOM的时候触发OOM killer。

oom\_kill\_allocating\_task=0，表示触发OOM killer的时候优先选择结束得分高的进程。

在系统中同时启动三个相同的测试进程（test、test1、test2），不断申请新的内存，并将test1的oom\_score\_adj设置成最大的1000，表示OOM killer优先结束该进程，直至内存耗尽触发系统OOM。

```
[root@localhost ~]# ps -ef | grep test
root          2938      2783  0 19:08 pts/2    00:00:00 ./test
root          2939      2822  0 19:08 pts/3    00:00:00 ./test1
root          2940      2918  0 19:08 pts/5    00:00:00 ./test2
[root@localhost ~]# echo 1000 > /proc/2939/oom_score_adj
[root@localhost ~]# cat /proc/2939/oom_score_adj
1000
```

### 3. 查看OOM信息。

经过一段时间后系统发生OOM并触发OOM killer，同时在/var/log/messages中打印系统所有进程的内存等信息并结束了test1进程：

图1 查看进程信息

```
[root@localhost ~]# ps -ef | grep test
root      2938      2783    0 19:08 pts/2        00:00:06 ./test
root      2940      2918    0 19:08 pts/5        00:00:06 ./test2
root      3383      2861    0 19:41 pts/4        00:00:00 grep --color=auto test
[root@localhost ~]#
```

### 图2 oom时的错误日志

```

Mar 4 19:15:13 localhost kernel: [5865.355934] kernel fault(0x2) notification starting on CPU 16
Mar 4 19:15:14 localhost kernel: [5865.355938] kernel fault(0x2) notification finished on CPU 16
Mar 4 19:15:14 localhost kernel: [5865.355944] top invoked oom: gfp_mask=0x100cca(GFP_HIGHUSER_MOVABLE), order=0
Mar 4 19:15:14 localhost kernel: [5865.355950] CPU: 16 PID: 3070 Comm: top Kdump: loaded tainted: G 0 5.10.0-136.12.0.86.r1466 73.hce2.x86_64 #1
Mar 4 19:15:14 localhost kernel: [5865.355950] Hardware name: OpenStack Foundation OpenStack Nova, BIOS rel-1.12.1-0-ga5cab58-20230319_150422-szxrxtoscil0000 04/01/2014
Mar 4 19:15:14 localhost kernel: [5865.355953] Call Trace:
Mar 4 19:15:14 localhost kernel: [5865.355967] dump_stack+0x57/0x6e
Mar 4 19:15:14 localhost kernel: [5865.355970] oom_show_debug_info.part.0+0x4a/0x131
Mar 4 19:15:14 localhost kernel: [5865.355973] ? printk+0x58/0x73
Mar 4 19:15:14 localhost kernel: [5865.355975] out_of_memory.cold+0x66/0x8b
Mar 4 19:15:14 localhost kernel: [5865.355978] __alloc_pages+0xeb9/0x1050
Mar 4 19:15:14 localhost kernel: [5865.355980] pagecache_get_page+0x1e/0x560
Mar 4 19:15:14 localhost kernel: [5865.355981] filemap_fault+0x2a2/0x790
Mar 4 19:15:14 localhost kernel: [5865.356060] ext4_filemap_fault+0x2d/0x50 [ext4]
Mar 4 19:15:14 localhost kernel: [5865.356070] do_fault+0x37/0x1a0
Mar 4 19:15:14 localhost kernel: [5865.356071] do_read_fault+0x31/0xf0
Mar 4 19:15:14 localhost kernel: [5865.356073] do_fault+0x75/0x150
Mar 4 19:15:14 localhost kernel: [5865.356074] handle_mm_fault+0x448/0x7b0
Mar 4 19:15:14 localhost kernel: [5865.356076] handle_mm_fault+0x9bc/0x2f0
Mar 4 19:15:14 localhost kernel: [5865.356079] exc_page_fault+0x1db/0x5e0
Mar 4 19:15:14 localhost kernel: [5865.356081] ? asm_exc_page_fault+0x8/0x30
Mar 4 19:15:14 localhost kernel: [5865.356082] asm_exc_page_fault+0x1e/0x30
Mar 4 19:15:14 localhost kernel: [5865.356086] RIP: 0033:0x7f4b7b88b560
Mar 4 19:15:14 localhost kernel: [5865.356093] Code: Unable to access opcode bytes at RIP 0x7f4b7b88b536.
Mar 4 19:15:14 localhost kernel: [5865.356094] RSP: 002b:00007ffefed523a8 EFLAGS: 00010202
Mar 4 19:15:14 localhost kernel: [5865.356096] RAX: 00007f4b7b88e688 RBX: 0000000000000000 RCX: 0000000000000000
Mar 4 19:15:14 localhost kernel: [5865.356096] RDX: 0000559f8dec1b81 RST: 00007f4b7b86e5a0 RDI: 000000000000005b
Mar 4 19:15:14 localhost kernel: [5865.356097] RBP: 00007f4b7b877008 R08: 00007ffefed523d0 R09: 00007ffefed523e0
Mar 4 19:15:14 localhost kernel: [5865.356098] R10: 0000000000000000 R11: 0000000000000246 R12: 00007ffefed52450
Mar 4 19:15:14 localhost kernel: [5865.356099] R13: 0000000000000001 R14: 0000000000000000 R15: 0000559f8del1d61
Mar 4 19:15:14 localhost kernel: [5865.360040] error: NEW PrintAllMemory is NULL
Mar 4 19:15:14 localhost kernel: [5865.369074] #012slab info:
Mar 4 19:15:14 localhost kernel: [5865.369239] slabinfo - version: 2.1
Mar 4 19:15:14 localhost kernel: [5865.369240] # name <active_objs> <num_objs> <objsize> <objperslab> <pagesperslab> : tunables <limit> <batchcount> <sharedfactor>
<num_slabs> <sharedavail>
Mar 4 19:15:14 localhost kernel: [5865.369241] nf_conntrack_expect 561 561 320 51 4 : tunables 0 0 0 : slabdata 11 11 0
Mar 4 19:15:14 localhost kernel: [5865.369244] nf_conntrack 461 461 704 46 8 : tunables 0 0 0 : slabdata 1 1 0
Mar 4 19:15:14 localhost kernel: [5865.369253] rpc_inode_cache 561 561 704 46 8 : tunables 0 0 0 : slabdata 1 1 0

```

### 图3 触发oom的进程

```

Mar 4 19:15:15 localhost kernel: 5805.370656] [ 2918] 0 2918 5974 2 69632 377 0 bash
Mar 4 19:15:15 localhost kernel: 5805.370657] [ 2938] 0 2938 656135 285536 5267456 365022 0 test
Mar 4 19:15:15 localhost kernel: 5805.370659] [ 2939] 0 2939 630530 291569 5062656 34344 1000 test1
Mar 4 19:15:15 localhost kernel: 5805.370660] [ 2940] 0 2940 6085 290489 4857056 309316 0 test2
Mar 4 19:15:15 localhost kernel: 5805.370662] [ 2941] 0 2941 4555 40 65536 293 0 sshd
Mar 4 19:15:15 localhost kernel: 5805.370664] [ 2945] 0 2945 4640 141 65536 238 0 sshd
Mar 4 19:15:15 localhost kernel: 5805.370665] [ 2946] 0 2946 4557 44 73728 201 0 sshd
Mar 4 19:15:15 localhost kernel: 5805.370667] [ 2950] 0 2950 4557 45 73728 201 0 sshd
Mar 4 19:15:15 localhost kernel: 5805.370668] [ 2951] 0 2951 2827 0 57344 135 0 sftp-server
Mar 4 19:15:15 localhost kernel: 5805.370669] [ 2961] 0 2961 5974 81 69632 324 0 bash
Mar 4 19:15:15 localhost kernel: 5805.370671] [ 3010] 0 3010 6435 199 73728 82 0 top
Mar 4 19:15:15 localhost kernel: 5805.370672] oom-kill:constraint=CONSTRAINT_NONE, nodemask=(null), cpuset=/, mems_allowed=0, global_oom, task_memcg=/system.slice/ssh.service, task=test1, pid=2939, uid=0
Mar 4 19:15:15 localhost kernel: 5805.370673] oom-kill:constraint=CONSTRAINT_NONE, nodemask=(null), cpuset=/, mems_allowed=0, global_oom, task=test1, pid=2939, uid=0
Mar 4 19:20:03 localhost systemd[1]: Starting system activity accounting tool...
Mar 4 19:20:03 localhost systemd[1]: sysstat-collect.service: Deactivated successfully.

```

## OOM可能的原因

## cgroup内存不足

使用的内存超出了cgroup中memory.limit\_in\_bytes配置的大小，如下示例演示memory.limit\_in\_bytes配置为80M，使用r  
模拟分配100M，触发OOM，/var/log/messages部分日志如下，可以从日志中看到memhog所在进程（PID: 2021820）使  
81920kB内存，超出了限制，触发了OOM：

```
warning|kernel[-]|[2919920.414131] memnog invoked oom-killer: gfp_mask=0xccc0(GFP_KERNEL), order=0,
oom_score_adj=0
info|kernel[-]|[2919920.414220] memory: usage 81920kB, limit 81920kB, failcnt 30
err|kernel[-]|[2919920.414272] Memory cgroup out of memory: Killed process 2021820 (memhog) total-
vm:105048kB, anon-rss:81884kB, file-rss:1544kB, shmem-rss:0kB, UID:0 pgtables:208kB
oom_score_adj:0
```

### 父cgroup内存不足

在子cgroup中内存仍然足够，但是父cgroup的内存不足，超过了内存限制，如下示例演示父cgroup memory.limit\_in\_bytes配置为80M，两个子cgroup memory.limit\_in\_bytes均配置为50M，在两个子cgroup中使用程序循环分配内存，触发OOM，/var/log/messages部分日志如下：

```
warning|kernel[-]|[2925796.529231] main invoked oom-killer: gfp_mask=0xccc0(GFP_KERNEL), order=0,
oom_score_adj=0
info|kernel[-]|[2925796.529315] memory: usage 81920kB, limit 81920kB, failcnt 199
err|kernel[-]|[2925796.529366] Memory cgroup out of memory: Killed process 3238866 (main) total-
vm:46792kB, anon-rss:44148kB, file-rss:1264kB, shmem-rss:0kB, UID:0 pgtables:124kB oom_score_adj:0
```

### 系统全局内存不足

一方面由于OS的空闲内存不足，有程序一直在申请内存，另一方面也无法通过内存回收机制解决内存不足的问题，因此触发了OOM，如下示例演示OS中使用程序循环分配内存，触发OOM，/var/log/messages部分日志如下，可以从日志中看到内存节点Node 0的空闲内存（free）已经低于了内存最低水位线（low），触发了OOM：

```
kernel: [ 1475.869152] main invoked oom: gfp_mask=0x100dca(GFP_HIGHUSER_MOVABLE|__GFP_ZERO),
order=0
kernel: [ 1477.959960] Node 0 DMA32 free:22324kB min:44676kB low:55844kB high:67012kB
reserved_highatomic:0kB active_anon:174212kB inactive_anon:1539340kB active_file:0kB
inactive_file:64kB unevictable:0kB writepending:0kB present:2080636kB managed:1840628kB
mlocked:0kB pagetables:7536kB bounce:0kB free_pcp:0kB local_pcp:0kB free_cma:0kB
kernel: [ 1477.960064] oom-kill:constraint=CONSTRAINT_NONE,nodemask=
(null),cpuset=/,mems_allowed=0,global_oom,task_memcg=/system.slice/sshd.service,task=main,pid=1822
,uid=0
kernel: [ 1477.960084] Out of memory: Killed process 1822 (main) total-vm:742748kB, anon-
rss:397884kB, file-rss:4kB, shmem-rss:0kB, UID:0 pgtables:1492kB oom_score_adj:1000
```

### 内存节点（Node）的内存不足

在NUMA存储模式下，OS会存在多个内存节点，如果程序指定使用特定节点的内存，可能在OS内存充足的情况下触发OOM，如下示例演示在两个内存节点的情况下，使用程序循环在Node 1分配内存，导致Node 1内存不足，但是OS内存足够，/var/log/messages部分日志如下：

```
kernel: [ 465.863160] main invoked oom: gfp_mask=0x100dca(GFP_HIGHUSER_MOVABLE|__GFP_ZERO),
order=0
kernel: [ 465.878286] active_anon:218 inactive_anon:202527 isolated_anon:0#012 active_file:5979
inactive_file:5231 isolated_file:0#012 unevictable:0 dirty:0 writeback:0#012 slab_reclaimable:6164
slab_unreclaimable:9671#012 mapped:4663 shmem:2556 pagetables:846 bounce:0#012 free:226231
```

```
reserved_highatomic:0KB active_anon:188kB inactive_anon:778076kB active_file:20kB
inactive_file:40kB unevictable:0kB writepending:0kB present:1048444kB managed:866920kB mlocked:0kB
pagetables:2752kB bounce:0kB free_pcp:144kB local_pcp:0kB free_cma:0kB
kernel: [ 933.264779] oom-
kill:constraint=CONSTRAINT_MEMORY_POLICY,nodemask=1,cpuset=/,mems_allowed=0-
1,global_oom,task_memcg=/system.slice/sshd.service,task=main,pid=1733,uid=0
kernel: [ 465.878438] Out of memory: Killed process 1734 (main) total-vm:239028kB, anon-
rss:236300kB, file-rss:200kB, shmem-rss:0kB, UID:0 pgtables:504kB oom_score_adj:1000
```

#### 其他可能原因

OS在内存分配的过程中，如果伙伴系统的内存不足，则系统会通过OOM Killer释放内存，并将内存提供至伙伴系统。

## OOM问题解决方法

从业务进程排查，确认是否有内存泄漏，导致OOM。

排查cgroup limit\_in\_bytes配置是否与业务内存规划匹配，如需要调整，可以手动执行以下命令修改配置参数：

```
echo <value> > /sys/fs/cgroup/memory/<cgroup_name>/memory.limit_in_bytes
```

如果确认业务需要比较多的内存，建议升级弹性云服务器内存规格。

### 相关文档

[节点插件状态离线问题排查](#)

[系统配置XPS的方法及影响说明](#)

[IPVS报错问题说明](#)

[panic\\_on\\_oom系统参数变更说明](#)

[如何检查节点是否支持eBPF功能？](#)

### 意见反馈

文档内容是否对您有帮助？

[提供反馈](#)

如您有其它疑问，您也可以通过华为云社区问答频道来与我们联系探讨

[盘古Doer提问](#) [云社区提问](#)