Hilbert Team

Алексей Цыкунов
CTO Hilbert Team

OOM Killer
Основы выживания

Обо мне

Алексей Цыкунов

- CTO в Hilbert Team
- Автор курсов по Linux в Otus
- Более 20 лет в IT
- Более 10 лет опыта оптимизации работы продуктовых команд и R&D-департаментов с помощью DevOps

О компании

IT-интегратор, отраслевой эксперт в **DevOps, DevSecOps, FinOps, DataOps**

Hilbert Team

Мы увеличиваем операционную эффективность и маржинальность бизнеса за счёт миграции в облака, оптимизации и автоматизации IT-инфраструктуры и процессов

- Сертифицированный партнер Yandex Cloud со специализациями по DevOps и Data Platform
- Партнер года 2023 Yandex Cloud в номинациях «Infra & DevOps» и «Финансы и страхование»
- Более 35 Cloud/DevOps, SRE, Data и ML-инженеров

Розыгрыш

@HilbertTeam_DevOops_bot

Для участия в розыгрыше мерча Hilbert Team, перейдите в бота и зарегистрируйтесь.

Розыгрыш состоится в конце выступления.

Roadmap

- DevOps и OOM – неизбежная встреча

Roadmap

- DevOps и OOM – неизбежная встреча
- Кто такой OOM Killer (и кто его нанял)?

Roadmap

- DevOps и OOM – неизбежная встреча
- Кто такой OOM Killer (и кто его нанял)?
- Сколько памяти в системе?

Roadmap

- DevOps и OOM – неизбежная встреча
- Кто такой OOM Killer (и кто его нанял)?
- Сколько памяти в системе?
- Что такое overcommit?

Roadmap

- DevOps и OOM – неизбежная встреча
- Кто такой OOM Killer (и кто его нанял)?
- Сколько памяти в системе?
- Что такое overcommit?
- Когда приходит OOM Killer?

Roadmap

- DevOps и OOM – неизбежная встреча
- Кто такой OOM Killer (и кто его нанял)?
- Сколько памяти в системе?
- Что такое overcommit?
- Когда приходит OOM Killer?
- Как OOM Killer выбирает жертву?

Roadmap

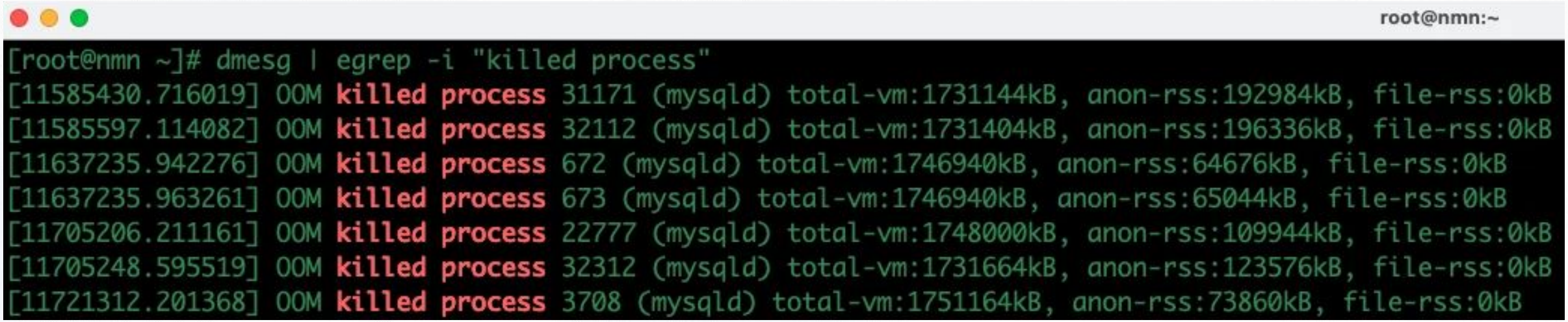
- DevOps и OOM – неизбежная встреча
- Кто такой OOM Killer (и кто его нанял)?
- Сколько памяти в системе?
- Что такое overcommit?
- Когда приходит OOM Killer?
- Как OOM Killer выбирает жертву?
- А как спастись?

Roadmap

- DevOps и OOM – неизбежная встреча
- Кто такой OOM Killer (и кто его нанял)?
- Сколько памяти в системе?
- Что такое overcommit?
- Когда приходит OOM Killer?
- Как OOM Killer выбирает жертву?
- А как спастись?
- OOM Killer на службе ~~его величества~~ Kubernetes

Неизбежные встречи: dmesg


```
dmesg -T | egrep -i 'killed process'
```



Неизбежные встречи: OOM Killed


```
State:          Running
 Started:       Sun, 16 Feb 2020 10:20:09 +0000
Last State:     Terminated
 Reason:        OOMKilled
 Exit Code:     137
 Started:       Sun, 16 Feb 2019 09:27:39 +0000
 Finished:      Sun, 16 Feb 2019 10:20:08 +0000
Restart Count:  7
```

Кто такой OOM Killer

Что такое OOM

- Ошибка **Out of Memory**, возвращаемая процессу при невозможности выделить память на его запрос

Что такое OOM

- Ошибка **Out of Memory**, возвращаемая процессу при невозможности выделить память на его запрос

- Состояние, в которое переходит ядро, при отсутствии достаточного количества свободной памяти

Что такое OOM

- поступает запрос на выделение памяти

Что такое OOM

- поступает запрос на выделение памяти
- проверка наличия свободной памяти

Что такое OOM

- поступает запрос на выделение памяти
- проверка наличия свободной памяти
  - **`vm_enough_memory()`**
    - **`-ENOMEM`**
    - есть «свободная» память

Что такое OOM

- поступает запрос на выделение памяти
- проверка наличия свободной памяти
  - **`vm_enough_memory()`**
    - **`-ENOMEM`**
    - есть «свободная» память
      - может возникнуть состояние OOM, когда не получилось выделить память

Что такое OOM

- поступает запрос на выделение памяти
- проверка наличия свободной памяти
  - **`vm_enough_memory()`**
    - **`-ENOMEM`**
    - есть «свободная» память
      - может возникнуть состояние OOM, когда не получилось выделить память
        - убедиться, что действительно не хватает памяти

Что такое OOM

- поступает запрос на выделение памяти
- проверка наличия свободной памяти
  - **`vm_enough_memory()`**
    - **`-ENOMEM`**
    - есть «свободная» память
      - может возникнуть состояние OOM, когда не получилось выделить память
        - убедиться, что действительно не хватает памяти
        - **выбрать процесс для убийства**

__vm_enough_memory()


```
/*
 * Sometimes we want to use more memory than we have
 */
if (sysctl_overcommit_memory == OVERCOMMIT_ALWAYS)
   return 0;

if (sysctl_overcommit_memory == OVERCOMMIT_GUESS) {
   if (pages > totalram_pages() + total_swap_pages)
         goto error;
   return 0;
}
```

overcommit

- **vm.overcommit_ratio** - уровень overcommit (в процентах)
  - по умолчанию 50

overcommit

- **vm.overcommit_ratio** - уровень overcommit (в процентах)
  - по умолчанию 50
- **vm.overcommit_kbytes** - абсолютное значение оверкоммита в кб

overcommit

- **vm.overcommit_ratio** - уровень overcommit (в процентах)
  - по умолчанию 50
- **vm.overcommit_kbytes** - абсолютное значение оверкоммита в кб
- **vm.overcommit_memory** - стратегия overcommit
  - 0 - OVERCOMMIT_GUESS
  - 1 - OVERCOMMIT_ALWAYS
  - 2 - OVERCOMMIT_NEVER

overcommit

- **vm.overcommit_ratio** - уровень overcommit (в процентах)
  - по умолчанию 50
- **vm.overcommit_kbytes** - абсолютное значение оверкоммита в кб
- **vm.overcommit_memory** - стратегия overcommit
  - 0 - OVERCOMMIT_GUESS
  - 1 - OVERCOMMIT_ALWAYS
  - 2 - OVERCOMMIT_NEVER
    - **swap + ram*overcommit_ratio/100**

Как управлять overcommit

- `echo 1 | sudo tee /proc/sys/vm/overcommit_memory`
- sysctl
  - vi /etc/sysctl.conf
    - vm.overcommit_memory=2
  - sysctl -p

__vm_enough_memory()

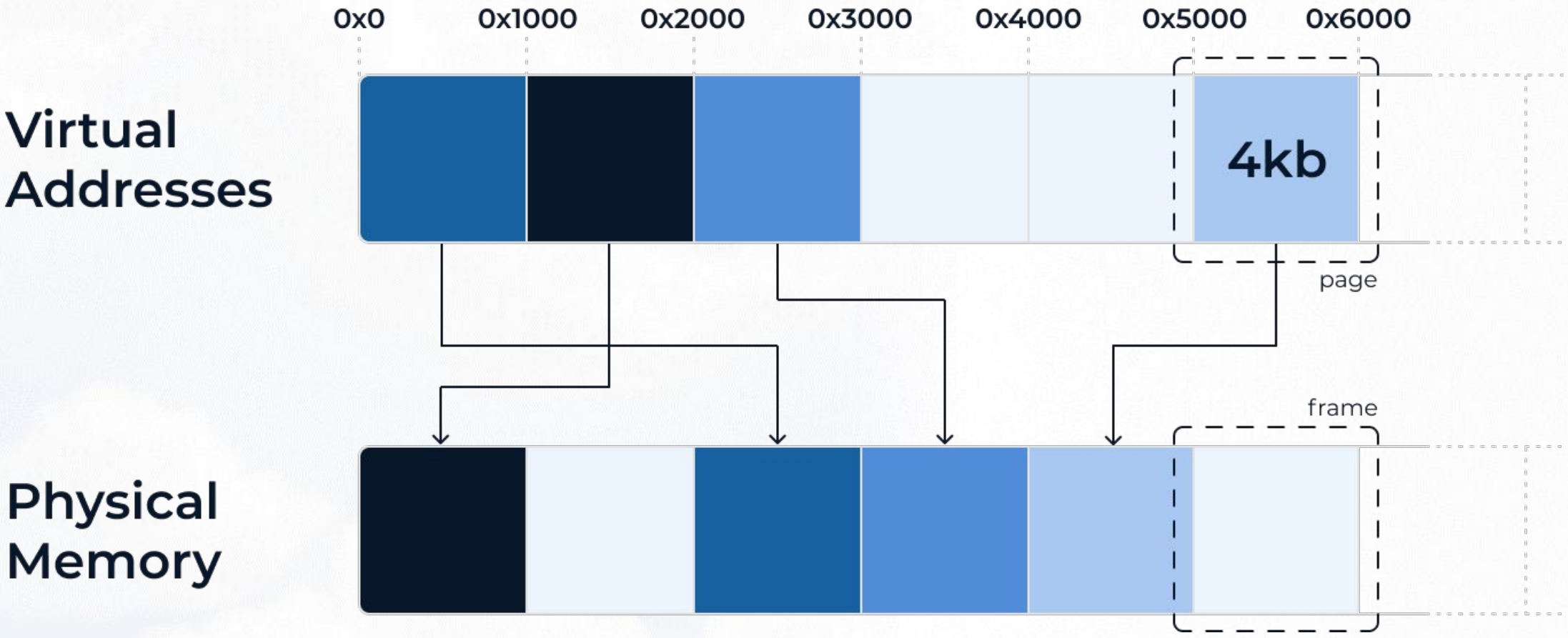
…

```
// Committed memory limit enforced when OVERCOMMIT_NEVER policy is

allowed = vm_commit_limit();

// Reserve some for root

if (!cap_sys_admin)
   allowed -= sysctl_admin_reserve_kbytes >> (PAGE_SHIFT - 10);
```

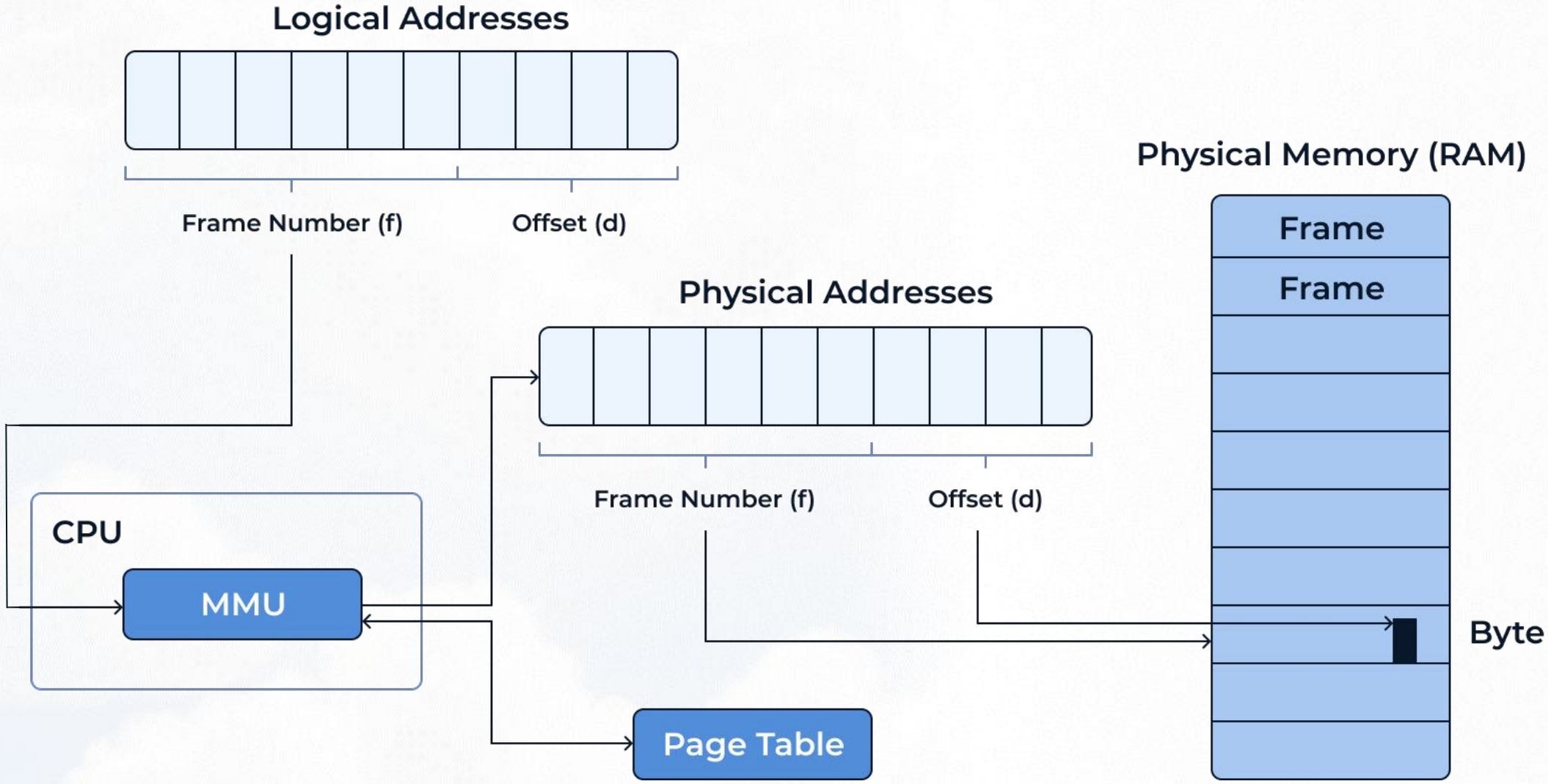
vm_commit_limit()


```
unsigned long allowed;

if (sysctl_overcommit_kbytes)
   allowed = sysctl_overcommit_kbytes >> (PAGE_SHIFT - 10);
else
   allowed = ((totalram_pages() - hugetlb_total_pages())
              * sysctl_overcommit_ratio / 100);
allowed += total_swap_pages;

return allowed;
```

Memory pages



Memory pages



/proc/meminfo


```
cat /proc/meminfo
MemTotal:         994548 kB
MemFree:           65228 kB
MemAvailable:     263724 kB
Buffers:           21396 kB
Cached:           304440 kB
SwapCached:        25260 kB
Active:           267424 kB
Inactive:         503720 kB
Active(anon):     110956 kB
Inactive(anon):   351176 kB
 ...
```

/proc/meminfo: Hugepages


```
cat /proc/meminfo | grep -ie "huge" -e "filepmd" |
grep -v "Shmem"
AnonHugePages:         0 kB
FileHugePages:         0 kB
FilePmdMapped:         0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
Hugetlb:               0 kB
```

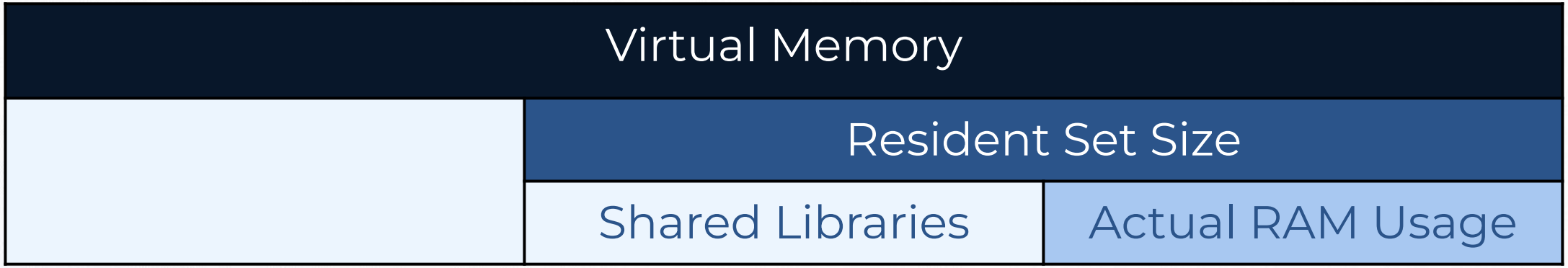
out_of_memory()


```
struct oom_control {
   /* Used to determine cpuset */
   struct zonelist *zonelist;

   /* Used to determine mempolicy */
   nodemask_t *nodemask;

   /* Memory cgroup in which oom is invoked, or NULL for
global oom */
   struct mem_cgroup *memcg;
…
```

out_of_memory()


```
/**
 * out_of_memory - kill the "best" process when we run out of memory
 * @oc: pointer to struct oom_control
 *
 * If we run out of memory, we have the choice between either
 * killing a random task (bad), letting the system crash (worse)
 * OR try to be smart about which process to kill. Note that we
 * don't have to be perfect here, we just have to be good.
 */
bool out_of_memory(struct oom_control *oc)
{
    unsigned long freed = 0;

    if (oom_killer_disabled)
        return false;
..  ..
    select_bad_process(oc);
```

Ещё немного про память




```
  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
23999 registry   20   0 1925M 29496 11008 S 66.1  0.2 29h17:25 /opt/gitlab
 2658 git        20   0 4553M 1109M 18484 S 54.9  6.9    305h sidekiq 6.5
 3117 git        20   0 4553M 1109M 18484 S 14.4  6.9 28h38:05 sidekiq 6.5
```

Hilbert Team

oom_badness()


```
unsigned long oom_badness(struct task_struct *p, unsigned long totalpages)
{
 ...

    /*
     * The baseline for the badness score is the proportion of RAM that each
     * task's rss, pagetable and swap space use.
     */
    points = get_mm_rss(p->mm) + get_mm_counter(p->mm, MM_SWAPENTS) +
        mm_pgtables_bytes(p->mm) / PAGE_SIZE;
    task_unlock(p);

    /* Normalize to oom_score_adj units */
    adj *= totalpages / 1000;
    points += adj;
```

Выбор жертвы

- `select_bad_process()`

Выбор жертвы

- `select_bad_process()`
  - рассчитывается количество поинтов

Выбор жертвы

- **`select_bad_process()`**
  - рассчитывается количество поинтов
    - на основе пропорций использования **`rss, pagetable и swap`**
    - с учетом **`oom_score_adj`**

Выбор жертвы

- **`select_bad_process()`**
  - рассчитывается количество поинтов
    - на основе пропорций использования **`rss, pagetable и swap`**
    - с учетом **`oom_score_adj`**
  - на основании этих поинтов выбирается процесс-жертва

Выбор жертвы

- **`select_bad_process()`**
  - рассчитывается количество поинтов
    - на основе пропорций использования **`rss, pagetable и swap`**
    - с учетом **`oom_score_adj`**
  - на основании этих поинтов выбирается процесс-жертва
  - эти поинты отражаются в **`/proc/<PID>/oom_score`**

oom_score

- показывает нам вероятность убийства процесса OOM Killer'ом
  - чем выше, тем вероятнее

- `cat /proc/<PID>/oom_score`

oom_score_adj

- регулирующий параметр
  - `/proc/<PID>/oom_score_adj`

- чем меньше, тем меньше oom_score
  - `#define OOM_SCORE_ADJ_MIN  (-1000)`
  - `#define OOM_SCORE_ADJ_MAX  1000`

- можно задать для каждого процесса

Управление oom_score_adj

- `echo -17 > /proc/12465/oom_adj`
- `echo -17 > /proc/12465/oom_score_adj`
- `choom -p 12465 -n -1000`

- systemd
  - `OOMScoreAdjust=-500`

Legacy ... oom_adj

Since Linux 2.6.36, use of this file is deprecated in favor of */proc/[pid]/oom_score_adj*.

For backward compatibility with previous kernels, */proc/[pid]/oom_adj* can still be used to tune the badness score. Its value is scaled linearly with *oom_score_adj*.

Writing to */proc/[pid]/oom_score_adj* or */proc/[pid]/oom_adj* will change the other with its scaled value.

```
/*
 * /proc/<pid>/oom_adj set to -17 protects from the oom killer
   for legacy purposes.
 */
#define OOM_DISABLE (-17)
/* inclusive */
#define OOM_ADJUST_MIN (-16)
#define OOM_ADJUST_MAX 15
```

Управление поведением OOM Killer’a

- **`vm.panic_on_oom`**
  - 0 - oom killer
  - 1 - kernel panic

- **`vm.oom_kill_allocating_task`**
  - 0 - эвристический поиск жертвы
  - 1 - убивается процесс, которому не хватило памяти

OOM in Kubernetes: cgroups v2

system oom killer

cgroup oom killer

OOM in Kubernetes: cgroups v2


```
/ # ls -c1 /sys/fs/cgroup/memory.*
/sys/fs/cgroup/memory.events.local
/sys/fs/cgroup/memory.high
/sys/fs/cgroup/memory.low
/sys/fs/cgroup/memory.oom.group
/sys/fs/cgroup/memory.pressure
/sys/fs/cgroup/memory.min
/sys/fs/cgroup/memory.current
/sys/fs/cgroup/memory.max
/sys/fs/cgroup/memory.stat
/sys/fs/cgroup/memory.events
/ #
```

OOM in Kubernetes: requests

- memory.min
  - жесткая защита
  - не позволяет освобождать занятую память
  - при отсутствии доступной памяти - вызывается OOM
- memory.low
  - мягкая защита
  - может быть освобождена при отсутствии доступной памяти

OOM in Kubernetes: limits

- memory.max
  - жесткое ограничение
  - при достижении этого лимита вызывается OOM killer внутри cgroup
- memory.high
  - мягкое ограничение
  - при достижении лимита начинается усиленное востребование доступной памяти

OOM in Kubernetes: регулировка

- memory.oom.group
  - выбрать и убить процесс в группе
  - убить всю cgroup

OOM in Kubernetes: мониторинг

- `memory.current`
- `memory.stat`

```
/ $ cat /sys/fs/cgroup/memory.stat
anon 7168000
file 0
kernel_stack 73728
slab 5128192
sock 0
shmem 0
file_mapped 0
file_dirty 0
file_writeback 0
anon_thp 0
inactive_anon 0
active_anon 7163904
inactive_file 0
active_file 0
unevictable 0
slab_reclaimable 2007040
slab_unreclaimable 3121152
```

OOM in Kubernetes: metrics

Метрики:

- container_memory_usage_bytes
- container_memory_working_set_bytes
- container_memory_rss
- container_memory_cache
- kube_pod_container_resource_limits{resource="memory"}

OOM in Kubernetes: cgroups

- считается полный объем занятой памяти в cgroup
  - RSS + cache
- OOM происходит внутри группы
- системы пытается высвободить память внутри группы или снаружи
- если освобождать нечего, приходит OOM и выбирает жертву внутри группы
  - можно настроить, чтобы убить всю группу

oom_score_adj в k8s

Quality of Service	oom_score_adj
Guaranteed	-997
Best Effort	1000
Burstable	*min(max(2, 1000 - (1000 × memoryRequestBytes) / machineMemoryCapacityBytes), 999)*

Выводы

- **OOM killer – ваш друг и помощник**
  - но нужно понимать, как он работает и регулируется

Выводы

- **OOM killer – ваш друг и помощник**
  - но нужно понимать, как он работает и регулируется
- **Читать код ядра не только не страшно**
  - но и полезно

Выводы

- **OOM killer – ваш друг и помощник**
  - но нужно понимать, как он работает и регулируется
- **Читать код ядра не только не страшно**
  - но и полезно
- **Важно понимать, как устроена память в Linux**
  - и проверять, какие изменения в новых версиях ядра

Розыгрыш

@HilbertTeam_DevOops_bot

Для участия в розыгрыше мерча Hilbert Team, перейдите в бота и зарегистрируйтесь.

Hilbert Team

Цыкунов Алексей

CTO Hilbert Team

@erlong15

alex.tsykunov@hilbertteam.com

Наш Telegram-канал

@hilbertteam