HugeTLB Controller

HugeTLB controller can be created by first mounting the cgroup filesystem.

# mount -t cgroup -o hugetlb none /sys/fs/cgroup

With the above step, the initial or the parent HugeTLB group becomes visible at /sys/fs/cgroup. At bootup, this group includes all the tasks in the system. /sys/fs/cgroup/tasks lists the tasks in this cgroup.

New groups can be created under the parent group /sys/fs/cgroup:

```
    # cd /sys/fs/cgroup
    # mkdir g1
    # echo $$ > g1/tasks
```

The above steps create a new group g1 and move the current shell process (bash) into it.

Brief summary of control files:

```
    hugetlb.<hugepagesize>.rsvd.limit_in_bytes            # set/show limit of "hugepagesize" hugetlb reservations
    hugetlb.<hugepagesize>.rsvd.max_usage_in_bytes        # show max "hugepagesize" hugetlb reservations and no-reserve
    hugetlb.<hugepagesize>.rsvd.usage_in_bytes            # show current reservations and no-reserve faults for "hugepage
    hugetlb.<hugepagesize>.rsvd.failcnt                   # show the number of allocation failure due to HugeTLB reservat
    hugetlb.<hugepagesize>.limit_in_bytes                 # set/show limit of "hugepagesize" hugetlb faults
    hugetlb.<hugepagesize>.max_usage_in_bytes             # show max "hugepagesize" hugetlb  usage recorded
    hugetlb.<hugepagesize>.usage_in_bytes                 # show current usage for "hugepagesize" hugetlb
    hugetlb.<hugepagesize>.failcnt                        # show the number of allocation failure due to HugeTLB usage l
    hugetlb.<hugepagesize>.numa_stat                      # show the numa information of the hugetlb memory charged to t
```

For a system supporting three hugepage sizes (64k, 32M and 1G), the control files include:

```
    hugetlb.1GB.limit_in_bytes
    hugetlb.1GB.max_usage_in_bytes
    hugetlb.1GB.numa_stat
    hugetlb.1GB.usage_in_bytes
    hugetlb.1GB.failcnt
    hugetlb.1GB.rsvd.limit_in_bytes
    hugetlb.1GB.rsvd.max_usage_in_bytes
    hugetlb.1GB.rsvd.usage_in_bytes
    hugetlb.1GB.rsvd.failcnt
    hugetlb.64KB.limit_in_bytes
    hugetlb.64KB.max_usage_in_bytes
    hugetlb.64KB.numa_stat
    hugetlb.64KB.usage_in_bytes
    hugetlb.64KB.failcnt
    hugetlb.64KB.rsvd.limit_in_bytes
    hugetlb.64KB.rsvd.max_usage_in_bytes
    hugetlb.64KB.rsvd.usage_in_bytes
    hugetlb.64KB.rsvd.failcnt
    hugetlb.32MB.limit_in_bytes
    hugetlb.32MB.max_usage_in_bytes
    hugetlb.32MB.numa_stat
    hugetlb.32MB.usage_in_bytes
    hugetlb.32MB.failcnt
    hugetlb.32MB.rsvd.limit_in_bytes
    hugetlb.32MB.rsvd.max_usage_in_bytes
    hugetlb.32MB.rsvd.usage_in_bytes
    hugetlb.32MB.rsvd.failcnt
```

1. Page fault accounting

hugetlb.<hugepagesize>.limit_in_bytes hugetlb.<hugepagesize>.max_usage_in_bytes hugetlb.<hugepagesize>.usage_in_bytes hugetlb.<hugepagesize>.failcnt

The HugeTLB controller allows users to limit the HugeTLB usage (page fault) per control group and enforces the limit during page fault. Since HugeTLB doesn't support page reclaim, enforcing the limit at page fault time implies that, the application will get SIGBUS signal if it tries to fault in HugeTLB pages beyond its limit. Therefore the application needs to know exactly how many HugeTLB pages it uses before hand, and the sysadmin needs to make sure that there are enough available on the machine for all the users to avoid processes getting SIGBUS.

2. Reservation accounting

hugetlb.<hugepagesize>.rsvd.limit_in_bytes hugetlb.<hugepagesize>.rsvd.max_usage_in_bytes hugetlb.<hugepagesize>.rsvd.usage_in_bytes hugetlb.<hugepagesize>.rsvd.failcnt

The HugeTLB controller allows to limit the HugeTLB reservations per control group and enforces the controller limit at reservation time and at the fault of HugeTLB memory for which no reservation exists. Since reservation limits are enforced at reservation time (on mmap or shget), reservation limits never causes the application to get SIGBUS signal if the memory was reserved before hand. For MAP_NORESERVE allocations, the reservation limit behaves the same as the fault limit, enforcing memory usage at fault time and causing the application to receive a SIGBUS if it's crossing its limit.

Reservation limits are superior to page fault limits described above, since reservation limits are enforced at reservation time (on mmap or shget), and never causes the application to get SIGBUS signal if the memory was reserved before hand. This allows for easier fallback to alternatives such as non-HugeTLB memory for example. In the case of page fault accounting, it's very hard to avoid processes getting SIGBUS since the sysadmin needs precisely know the HugeTLB usage of all the tasks in the system and make sure there is enough pages to satisfy all requests. Avoiding tasks getting SIGBUS on overcommited systems is practically impossible with page fault accounting.

3. Caveats with shared memory

For shared HugeTLB memory, both HugeTLB reservation and page faults are charged to the first task that causes the memory to be reserved or faulted, and all subsequent uses of this reserved or faulted memory is done without charging.

Shared HugeTLB memory is only uncharged when it is unreserved or deallocated. This is usually when the HugeTLB file is deleted, and not when the task that caused the reservation or fault has exited.

4. Caveats with HugeTLB cgroup offline.

When a HugeTLB cgroup goes offline with some reservations or faults still charged to it, the behavior is as follows:

- The fault charges are charged to the parent HugeTLB cgroup (reparented),
- the reservation charges remain on the offline HugeTLB cgroup.

This means that if a HugeTLB cgroup gets offlined while there is still HugeTLB reservations charged to it, that cgroup persists as a zombie until all HugeTLB reservations are uncharged. HugeTLB reservations behave in this manner to match the memory controller whose cgroups also persist as zombie until all charged memory is uncharged. Also, the tracking of HugeTLB reservations is a bit more complex compared to the tracking of HugeTLB faults, so it is significantly harder to reparent reservations at offline time.