

[PATCH] hugetlb: vmstat events for huge page allocations

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-03/msg12052.html>

- *From:* Eric B Munson <ebmunson@xxxxxxxxxx>
 - *Date:* Mon, 31 Mar 2008 23:49:08 +0800
-

Andrew, please merge the following

From: Adam Litke <agl@xxxxxxxxxx>

Following feedback here is v3

Changes from v2: Renamed stats to HTLB_BUDDY_PGALLOC[_FAIL] and
htlb_buddy_alloc_[success|fail]

Calling __count_vm_event from alloc_buddy_huge_page with lock held

Allocating huge pages directly from the buddy allocator is not guaranteed to succeed. Success depends on several factors (such as the amount of physical memory available and the level of fragmentation). With the addition of dynamic hugetlb pool resizing, allocations can occur much more frequently. For these reasons it is desirable to keep track of huge page allocation successes and failures.

Add two new vmstat entries to track huge page allocations that succeed and fail. The presence of the two entries is contingent upon CONFIG_HUGETLB_PAGE being enabled.

This patch was created against linux-2.6.25-rc5

Signed-off-by: Adam Litke <agl@xxxxxxxxxx>

Signed-off-by: Eric Munson <ebmunson@xxxxxxxxxx>

It applied cleanly to 2.6.25-rc5-mm1 as well so I tested using a small x86 machine and found no problems. There was one curiosity with libhugetlbf with dynamic pool resizing was that two failures were accounted for when the mapping was larger than the number of huge pages could be allocated. However, it turned out that the first failure was libhugetlbf calling mlock() but not existing on failure and the second was the page fault. So, while I didn't expect it in advance, it is still correct accounting.

Tested-by: Mel Gorman <mel@xxxxxxxxxx>

Reviewed-by: Andy Whitcroft <apw@xxxxxxxxxx>

include/linux/vmstat.h | 8 ++++++--
mm/hugetlb.c | 7 ++++++

[PATCH] hugetlb: vmstat events for huge page allocations

mm/vmstat.c | 4 ++++

3 files changed, 18 insertions(+), 1 deletion(-)

diff --git a/include/linux/vmstat.h b/include/linux/vmstat.h

index 9f1b4b4..f68f538 100644

--- a/include/linux/vmstat.h

+++ b/include/linux/vmstat.h

@@ -25,6 +25,12 @@

#define HIGHMEM_ZONE(xx)

#endif

+#ifdef CONFIG_HUGETLB_PAGE

+#define HTLB_STATS HTLB_BUDDY_PGALLOC, HTLB_BUDDY_PGALLOC_FAIL,

+#else

+#define HTLB_STATS

+#endif

+

#define FOR_ALL_ZONES(xx) DMA_ZONE(xx) DMA32_ZONE(xx) xx##_NORMAL

HIGHMEM_ZONE(xx) , xx##_MOVABLE

enum vm_event_item { PGPGIN, PGPGOUT, PSWPIN, PSWPOUT,

@@ -36,7 +42,7 @@ enum vm_event_item { PGPGIN, PGPGOUT, PSWPIN, PSWPOUT,

FOR_ALL_ZONES(PGSCAN_KSWAPD),

FOR_ALL_ZONES(PGSCAN_DIRECT),

PGINODESTEAL, SLABS_SCANNED, KSWAPD_STEAL, KSWAPD_INODESTEAL,

- PAGEOUTRUN, ALLOCSTALL, PGROTATED,

+ PAGEOUTRUN, ALLOCSTALL, PGROTATED, HTLB_STATS

NR_VM_EVENT_ITEMS

};

diff --git a/mm/hugetlb.c b/mm/hugetlb.c

index 74c1b6b..dd20cb0 100644

--- a/mm/hugetlb.c

+++ b/mm/hugetlb.c

@@ -239,6 +239,11 @@ static int alloc_fresh_huge_page(void)

hugetlb_next_nid = next_nid;

} while (!page && hugetlb_next_nid != start_nid);

+ if (ret)

+ count_vm_event(HTLB_BUDDY_PGALLOC);

+ else

+ count_vm_event(HTLB_BUDDY_PGALLOC_FAIL);

+

return ret;

}

@@ -299,9 +304,11 @@ static struct page *alloc_buddy_huge_page(struct vm_area_struct *vma,

*/

nr_huge_pages_node[nid]++;

surplus_huge_pages_node[nid]++;

+ __count_vm_event(HTLB_BUDDY_PGALLOC);

[PATCH] hugetlb: vmstat events for huge page allocations

[PATCH] hugetlb: vmstat events for huge page allocations

```
} else {
nr_huge_pages--;
surplus_huge_pages--;
+ __count_vm_event(HTLB_BUDDY_PGALLOC_FAIL);
}
spin_unlock(&hugetlb_lock);

diff --git a/mm/vmstat.c b/mm/vmstat.c
index 422d960..34d1fd9 100644
--- a/mm/vmstat.c
+++ b/mm/vmstat.c
@@ -644,6 +644,10 @@ static const char * const vmstat_text[] = {
"allocstall",

"pgrotated",
+#ifdef CONFIG_HUGETLB_PAGE
+ "htlb_buddy_alloc_success",
+ "htlb_buddy_alloc_fail",
+#endif
#endif
};
```

Attachment: signature.asc

Description: This is a digitally signed message part