author	Gavin Shan <gshan@redhat.com>	2024-06-27 10:39:52 +1000
committer	Andrew Morton <akpm@linux-foundation.org>	2024-07-03 22:40:37 -0700
commit	9fd154ba926b34c833b7bfc4c14ee2e931b3d743 (patch)	
tree	1277bb1fb828ddf2ab4085bcd4a0929146b618eb	
parent	3390916aca7af1893ed2ebcdfee1d6fdb65bb058 (diff)	
download	linux-9fd154ba926b34c833b7bfc4c14ee2e931b3d743.tar.gz	

diff options

context: 3

space: include

mode: unified

mm/shmem: disable PMD-sized page cache if needed

```
For shmem files, it's possible that PMD-sized page cache can't be
supported by xarray.  For example, 512MB page cache on ARM64 when the base
page size is 64KB can't be supported by xarray.  It leads to errors as the
following messages indicate when this sort of xarray entry is split.

WARNING: CPU: 34 PID: 7578 at lib/xarray.c:1025 xas_split_alloc+0xf8/0x128
Modules linked in: binfmt_misc nft_fib_inet nft_fib_ipv4 nft_fib_ipv6   \
nft_fib nft_reject_inet nf_reject_ipv4 nf_reject_ipv6 nft_reject        \
nft_ct nft_chain_nat nf_nat nf_conntrack nf_defrag_ipv6 nf_defrag_ipv4  \
ip_set rfkill nf_tables nfnetlink vfat fat virtio_balloon drm fuse xfs  \
libcrc32c crct10dif_ce ghash_ce sha2_ce sha256_arm64 sha1_ce virtio_net \
net_failover virtio_console virtio_blk failover dimlib virtio_mmio
CPU: 34 PID: 7578 Comm: test Kdump: loaded Tainted: G W 6.10.0-rc5-gavin+ #9
Hardware name: QEMU KVM Virtual Machine, BIOS edk2-20240524-1.el9 05/24/2024
pstate: 83400005 (Nzcv daif +PAN -UAO +TCO +DIT -SSBS BTYPE=--)
pc : xas_split_alloc+0xf8/0x128
lr : split_huge_page_to_list_to_order+0x1c4/0x720
sp : ffff8000882af5f0
x29: ffff8000882af5f0 x28: ffff8000882af650 x27: ffff8000882af768
x26: 0000000000000cc0 x25: 000000000000000d x24: ffff00010625b858
x23: ffff8000882af650 x22: fffffffdfc0900000 x21: 0000000000000000
x20: 0000000000000000 x19: fffffffdfc0900000 x18: 0000000000000000
x17: 0000000000000000 x16: 0000018000000000 x15: 52f8004000000000
x14: 0000e00000000000 x13: 0000000000002000 x12: 0000000000000020
x11: 52f8000000000000 x10: 52f8e1c0ffff6000 x9 : ffffbeb9619a681c
x8 : 0000000000000003 x7 : 0000000000000000 x6 : ffff00010b02ddb0
x5 : ffffbeb96395e378 x4 : 0000000000000000 x3 : 0000000000000cc0
x2 : 000000000000000d x1 : 000000000000000c x0 : 0000000000000000
Call trace:
 xas_split_alloc+0xf8/0x128
 split_huge_page_to_list_to_order+0x1c4/0x720
 truncate_inode_partial_folio+0xdc/0x160
 shmem_undo_range+0x2bc/0x6a8
 shmem_fallocate+0x134/0x430
 vfs_fallocate+0x124/0x2e8
 ksys_fallocate+0x4c/0xa0
 __arm64_sys_fallocate+0x24/0x38
 invoke_syscall.constprop.0+0x7c/0xd8
 do_el0_svc+0xb4/0xd0
 el0_svc+0x44/0x1d8
 el0t_64_sync_handler+0x134/0x150
 el0t_64_sync+0x17c/0x180

Fix it by disabling PMD-sized page cache when HPAGE_PMD_ORDER is larger
than MAX_PAGECACHE_ORDER.  As Matthew Wilcox pointed, the page cache in a
shmem file isn't represented by a multi-index entry and doesn't have this
limitation when the xarry entry is split until commit 6b24ca4a1a8d ("mm:
Use multi-index entries in the page cache").
```

```
Link: https://lkml.kernel.org/r/20240627003953.1262512-5-gshan@redhat.com
Fixes: 6b24ca4a1a8d ("mm: Use multi-index entries in the page cache")
Signed-off-by: Gavin Shan <gshan@redhat.com>
Acked-by: David Hildenbrand <david@redhat.com>
Cc: Darrick J. Wong <djwong@kernel.org>
Cc: Don Dutile <ddutile@redhat.com>
Cc: Hugh Dickins <hughd@google.com>
Cc: Linus Torvalds <torvalds@linux-foundation.org>
Cc: Matthew Wilcox (Oracle) <willy@infradead.org>
Cc: Ryan Roberts <ryan.roberts@arm.com>
Cc: William Kucharski <william.kucharski@oracle.com>
Cc: Zhenyu Zhang <zhenyzha@redhat.com>
Cc: <stable@vger.kernel.org>    [5.17+]
Signed-off-by: Andrew Morton <akpm@linux-foundation.org>
```

Diffstat

-rw-r--r-- mm/shmem.c 15

1 files changed, 13 insertions, 2 deletions

```
diff --git a/mm/shmem.c b/mm/shmem.c
index a8b181a6340296..c1befe046c7eb2 100644
--- a/mm/shmem.c
+++ b/mm/shmem.c
@@ -541,8 +541,9 @@ static bool shmem_confirm_swap(struct address_space *mapping,

 static int shmem_huge __read_mostly = SHMEM_HUGE_NEVER;

-bool shmem_is_huge(struct inode *inode, pgoff_t index, bool shmem_huge_force,
-                   struct mm_struct *mm, unsigned long vm_flags)
+static bool __shmem_is_huge(struct inode *inode, pgoff_t index,
+                            bool shmem_huge_force, struct mm_struct *mm,
+                            unsigned long vm_flags)
 {
        loff_t i_size;

@@ -573,6 +574,16 @@ bool shmem_is_huge(struct inode *inode, pgoff_t index, bool shmem_huge_force,
        }
 }

+bool shmem_is_huge(struct inode *inode, pgoff_t index,
+                   bool shmem_huge_force, struct mm_struct *mm,
+                   unsigned long vm_flags)
+{
+       if (HPAGE_PMD_ORDER > MAX_PAGECACHE_ORDER)
+               return false;
+
+       return __shmem_is_huge(inode, index, shmem_huge_force, mm, vm_flags);
+}
+
 #if defined(CONFIG_SYSFS)
 static int shmem_parse_huge(const char *str)
 {
```