Linux Plumbers Conference

Vienna, Austria | September 18-20, 2024

Transitioning get_user_pages (GUP) to folio based mapping

-Kundan Kumar

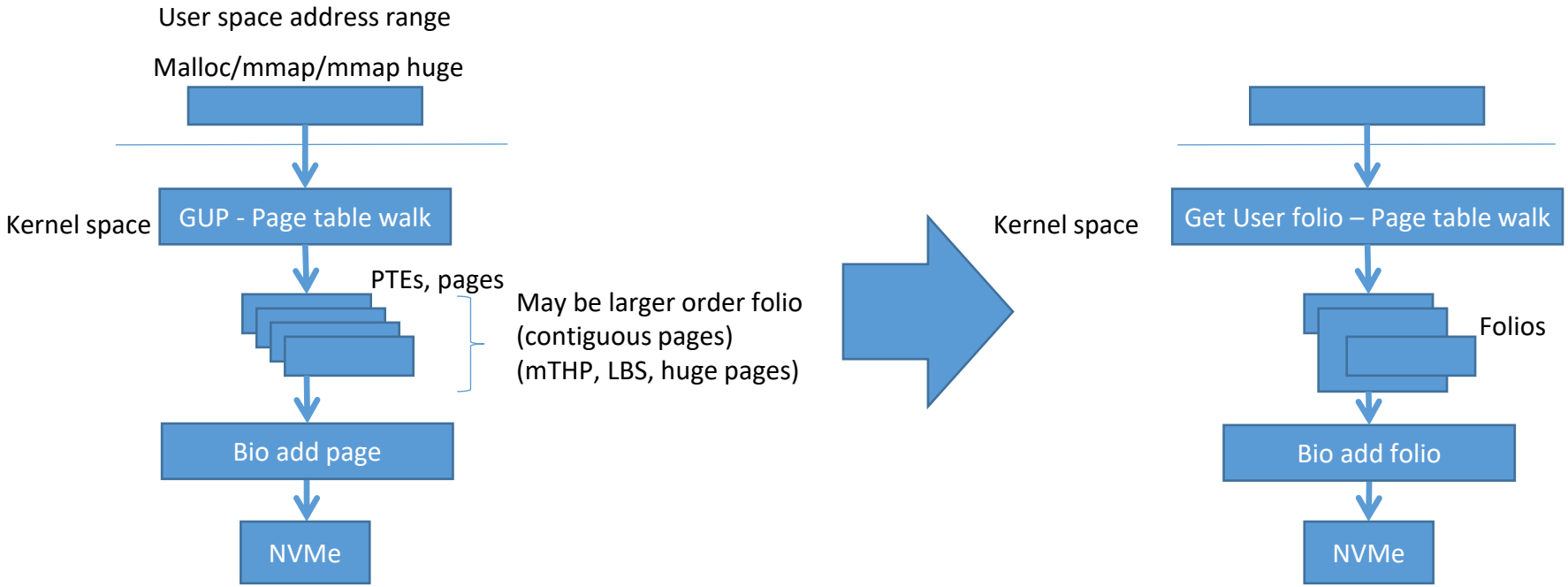
Samsung Semiconductor India Research

LINUX PLUMBERS CONFERENCE | Vienna, Austria
Sept. 18-20, 2024

Agenda

- Current I/O path vs proposed I/O path
- What happens after the conversion
- Current function call stack
- Discussion on GUP to folio-based mapping() solution
- Challenges with conversion to folio based mapping
  - folio_vec allocation
  - function signature change

Current I/O path vs proposed I/O path



What happens after the conversion

```
18)           |          bio_iov_iter_get_pages() {
18)           |            pin_user_pages_fast() {
18)  0.110 us |              is_valid_gup_args();
18)           |              gup_fast_fallback() {
18)           |                __pte_offset_map() {
18)  0.114 us |                  __rcu_read_lock();
18)  0.353 us |                }
18)           |              try_grab_folio() {
18)  0.119 us |                  mod_node_page_state();
18)  0.386 us |                }
18)  0.120 us |                gup_fast_folio_allowed();
18)           |              try_grab_folio() {
18)  0.109 us |                  mod_node_page_state();
18)  0.359 us |                }
 <---------32 times--------->
18) + 20.631 us |              }
18) + 21.062 us |            }
18)  0.117 us |            __bio_add_page();
18)  0.209 us |            bvec_try_merge_page();
18)  0.107 us |            bvec_try_merge_page();
 <---------32 times--------->
18)  0.111 us |            bvec_try_merge_page();
18) + 28.382 us |          }
```

Write I/O bs = 128K
(mTHP enabled)

```
5)           |          bio_iov_iter_get_pages() {
5)           |            __bio_iov_iter_get_pages() {
5)           |              pin_user_folios_fast() {
5)           |                __pte_offset_map() {
5)  0.170 us |                  __rcu_read_lock();
5)  0.510 us |                }
5)           |              try_grab_folio() {
5)  0.170 us |                  mod_node_page_state();
5)  0.510 us |                }
5)  0.170 us |                gup_fast_folio_allowed();
5)  0.170 us |                __rcu_read_unlock();
5)  2.670 us |              }
5)  3.310 us |            }
5)  3.750 us |          }
```

ftrace(function graph)
After the solution

28.382 us to 3.75 us for write I/O : 87% reduction

Current function call stack

- Current GUP code flow is 4K pages based
- It takes care of memory backed by huge pages also.
- Good news is the ref count increment is done on folios. (Thanks to previous work by Willy)

Code flow and the functions involved

```
__bio_iov_iter_get_pages
  iov_iter_extract_pages
    iov_iter_extract_user_pages
      pin_user_pages_fast
        internal_get_user_pages_fast
          gup_fast_fallback(start, nr_pages, gup_flags, pages);
            gup_fast(start, end, gup_flags, pages)
              gup_fast_pgd_range(start, end, gup_flags, pages, &nr_pinned);
                gup_fast_pgd_leaf(pgd, pgdp, addr, next, flags
                  record_subpages
                gup_hugepd(NULL, __hugepd(pgd_val(pgd)), addr,  PGDIR_SHIFT, next, flags, pages, nr)
                  gup_hugepte
                    record_subpages(page, sz, addr, end, pages + *nr);
                gup_fast_p4d_range(pgdp, pgd, addr, next, flags,pages, nr)
                  gup_hugepd
                  gup_fast_pud_range
                    gup_fast_pud_leaf
                    gup_hugepd
                    gup_fast_pmd_range
                      gup_fast_pmd_leaf
                      gup_hugepd
                      gup_fast_pte_range
                        try_grab_folio
                          try_get_folio
                          atomic_add(refs, &folio->_pincount);
                          folio_ref_add
__gup_longterm_locked(current->mm, start, nr_pages - nr_pinned,
  __get_user_pages_locked(mm, start, nr_pages,
    find_vma
    virt_to_page
```

Pages array passed

Per page reference taken(on folio)

GUP to Folio-based Mapping() solution discussion

- Provide a folio_vec array as a result of mapping user space memory (folio_vec used by bcachefs)

```
struct folio_vec {
        struct folio    *fv_folio;
        size_t          fv_offset;
        size_t          fv_len;
};
```

- Converted functions signatures to accept a folio_vec array.
- Move COW consideration to GUP(check for contiguous pages and construct large folio_vec items).
- Take reference on larger folio only once, and fill folio_vec array with larger folios.
- Bio layer can just add the folios, without even taking care of the contiguity.

```
__bio_iov_iter_get_pages
iov_iter_extract_folios nr_folio_vecs = ffff
iov_iter_extract_user_folios *folios = ffff9
want_folios_array count = 1 *res = ffff9f8bfa
want_folios_array *res = ffff9f8bd0c6f8e0
iov_iter_extract_user_folios folios[0] = fff
pin_user_folios_fast gup_flags = 1024
is_valid_gup_folio_args folios = ffff9f8bd0c
is_valid_gup_folio_args assign flags
pin_user_folios_fast gup_flags = 525312
gup_folio_fast_fallback
gup_folio_fast_fallback gup_flags = 80400
gup_folio_fast
gup_folio_fast_pgd_range
gup_folio_fast_p4d_range
gup_folio_fast_pud_range
gup_folio_fast_pmd_range flags = 250
gup_folio_fast_pte_range addr = 7f12a1aa2000
gup_folio_fast_pte_range page = ffffde99862e
gup_folio_fast_pte_range folios[0] = 0
gup_folio_fast_pte_range folio = ffffde99862
gup_folio_fast_pte_range vec = ffff9f8bd0c6f
gup_folio_fast_pte_range
gup_folio_fast_pte_range grab_folio = ffffde
gup_folio_fast_pte_range
gup_folio_fast_pte_range
gup_folio_fast_pte_range *nr_folio_vecs = 1
gup_folio_fast_pte_range
gup_folio_fast_pte_range vec = ffff9f8bd0c6f
gup_folio_fast_pmd_range
gup_folio_fast_pud_range
gup_folio_fast_p4d_range
gup_folio_fast_pgd_range
gup_folio_fast nr_pinned = 1
gup_folio_fast_fallback nr_pinned = 1
gup_folio_fast_fallback
__bio_iov_iter_get_pages folio = ffffde99862
```

folio_vec array passed

- Per large folio reference
- Check for COW

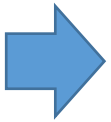
Challenges with conversion to folio based mapping

- Changes in signature of several functions
  GUP to folio mapping conversion prototype, changes the signature of series of functions. This involves the functions from iov_iter, pin_user_pages, huge_pages and page table walk. There are several callers for these functions, how shall we deal with same ?

```
static unsigned long gup_fast(unsigned long start, unsigned long end,
                unsigned int gup_flags, struct page **pages)
```

```
static unsigned long gup_folio_fast(unsigned long start, unsigned long end,
                unsigned int gup_flags, struct folio_vec **folios,
                int *nr_folio_vecs)
```

```
static int gup_fast_pte_range(pmd_t pmd, pmd_t *pmdp, unsigned long addr,
                unsigned long end, unsigned int flags, struct page **pages,
                int *nr)
```



```
static int gup_folio_fast_pte_range(pmd_t pmd, pmd_t *pmdp, unsigned long addr,
                unsigned long end, unsigned int flags, struct folio_vec **folios
                int *nr_folio_vecs)
```

Challenges with conversion to folio based mapping

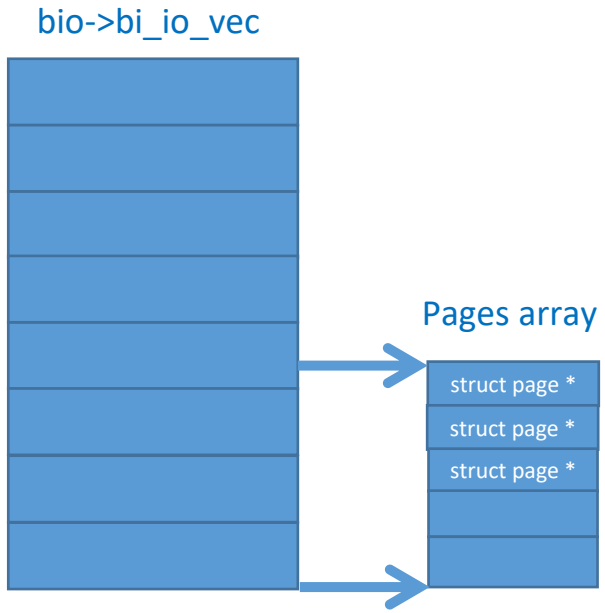
- **Allocation of a folio_vec array**

  An allocation of folio_vec array with size equivalent to number of pages is required.
  This Allocation needs to be done per I/O.
  Currently there is no allocation for pages array as the same memory of bi_io_vec is used

- Shall we avoid allocation using folio_batch approach (Suggested by Hannes) ?

```
struct folio_batch {
        unsigned char nr;
        unsigned char i;
        bool percpu_pvec_drained;
        struct folio *folios[PAGEVEC_SIZE];
};
```

- Shall we just do a allocation of folio_vec array for IOs which occupy more than 32 pages ?



Sample implementation

- Implementation for work shown in todays talk
  https://github.com/SamsungDS/linux/commit/7d18d8266bf5edd9d8db96183b2f63401125777c
- Major changes in this commit:
  1. Passing and retrieving of folio_vec array instead of pages array.
  2. Modification to terminal functions of page table walk :
     A) gup_folio_fast_pte_range(): Changes to pin folio once for contiguous pages belonging to same folio
     B) __get_user_folios(): In case a fault returns a large folio then add a bigger folio.
     C) record_subfolios(): return large folio
  3. Pinning and unpinning per folio_vec instead of pages.

References and pointers

1. Previous work of block folio which saw the need for get_user_folios (Applied!!) https://lore.kernel.org/linux-block/172606129719.167290.14045583678958786569.b4-ty@kernel.dk/T/#t

Thank you