Hewlett Packard
Enterprise

On-Demand Paging - ODP

Ian Ziemba, Chuck Fossen

May 28, 2024

Use Cases

Use Case	Description
Page Migration	• Application, OS, or some other entity has triggered a page migration • Example: System memory migrating to GPU memory • OS has NUMA balance enabled • Memory is migrating to balance NUMA node memory usage
Fork	• Avoids page pinning with copy-on-fork
Dynamic Memory Regions	• Application registers a large region of its virtual address range • Range may not have VMAs backing it (i.e., the range is sparsely populated or has holes). • Application dynamically maps and unmaps memory into this range • Expects ODP to detect and resolve the page changes
Transparent Huge Pages (THP)	• Running with larger page sizes is typically beneficial for both applications and NICs • Not all applications are set up to use libhugetlbfs • They use standard malloc • THP may help applications by dynamically generating a system huge page size (e.g., 2MiB on x86) and split the huge page into base page size (e.g., 4KiB on x86) if memory pressure becomes an issue

ODP Requirements

- Provider must use ODP to resolve missing pages associated with MR
  - ODP will typically occur in the data path of an RDMA operation
- Unmap and memory migration must not invalidate MR handle
  - Providers must ensure pages backing MR handle are invalidated
  - MR handle is still usable for future RDMA operations
- Page migration must not trigger MR cache eviction
  - Only free/munmap will trigger ODP MR cache eviction
- User-provided MRs must be accepted as input for local data transfers
  - Only supported with FI_MR_LOCAL today
  - Non-FI_MR_LOCAL providers rely on the MR cache for this
- Must support non-ODP and ODP memory registrations within the same process
  - Need a mechanism to selectively control ODP behavior per MR
- Hint indicating ODP behavior
- Ability to notify provider to refresh pages of an ODP MR
- User hint to provide huge page size
  - Needed to support non-CPU standard page sizes
  - IOTLB optimization

Core ODP Support

- Meets requirements for page migration, fork, and THP use-cases
- Foundation for dynamic memory regions use-case

New FI_ODP Capability

- Issue: ODP is a new provider behavior which needs advertising
  - Forcing ODP for all memory registrations is wrong
- Need new capability flags defined
- Proposal: New FI_ODP secondary capability to signal if ODP is supported
  - Definition: When set and used in conjunction with FI_MR_ALLOCATED, the provider can support memory registration without physical pages backing the region and resolve missing pages when the corresponding memory region is used for RDMA.
- Proposal: Update FI_MR_ALLOCATED description with ODP support
  - Definition: Indicates that memory registration occurs on allocated data buffers, and physical pages must back all virtual addresses being registered. When the FI_ODP secondary capability is set and FI_ODP is specified in the message flags, physical pages need not back the virtual address range.
- ODP will not be supported without FI_MR_ALLOCATED
- Running without FI_MR_ALLOCATED is **not** the same as ODP
  - Fails to address what happens when this memory region is used for RDMA
    - fi_mr_refresh() could be used for this but requires FI_MR_MMU_NOTIFY to be set

Hint Describing ODP Behavior

- Issue: Providers can implement various ODP policies resulting in various performance profiles
- Need to generalize ODP policies
  - Policies should be aligned to OS capabilities
- Need mechanism for users to select ODP policy during initial memory registration
  - Policy becomes meaningless after ODP has happened
- Proposal: New domain attribute field, `odp_hint`, which can be used to provide an ODP policy hint
  - FI_ODP_HINT_NONE: ODP memory/buffer registrations should not fault or prefetch pages
  - FI_ODP_HINT_PREFETCH: ODP memory/buffer registrations should prefetch resident pages
    - Maps to IBV_ADVISE_MR_ADVICE_PREFETCH_NO_FAULT
  - FI_ODP_HINT_FAULT: ODP memory/buffer registrations should fault in all pages of the memory region that are backed by a VMA
    - Maps to IBV_ADVISE_MR_ADVICE_PREFETCH_WRITE
- Providers should treat this as a hint
  - Memory registration should not fail due to ODP policy being applied
- Can be passed to fi_mr_regattr
  - FI_ODP_HINT_* can be supplied to fi_mr_regattr to override the domain attribute odp_hint

Per MR ODP Support

- Issue: Users should not be forced into doing ODP for all MRs
  - ODP may impact performance and should only be used if use-case requires it
- Need mechanisms to signal if an ODP should be used per MR
  - Supporting ODP and running without FI_MR_LOCAL complicates this
- Proposal: New FI_ODP flag to denote ODP
  - Used to indicate on-demand paging should be used with this buffer
    - Uses the domain attribute odp_hint for ODP registration
    - Memory migration will not trigger MR cache eviction
    - Munmap/free will trigger MR cache eviction
      - This is done to prevent stale MR cache entries
  - Valid for following APIs
    - fi_mr_regattr
    - fi_*msg()
      - Flag is invalid for domains opened with FI_MR_LOCAL
    - EP TX/RX operational flags
      - Flag is invalid for domains opened with FI_MR_LOCAL

Additional ODP Support

- Builds on core ODP support
- Meets requirements for dynamic memory regions use-case

Update MR Local Mode behavior

- Issue: For persistent, dynamic MRs, the MR must be reused for performance reasons
  - Relying on MR cache must be avoided
    - MR cache will evict entries on munmap/free
  - Problematic for providers running without FI_MR_LOCAL
    - Current definition of running without FI_MR_LOCAL prevents users from passing MR descriptor for local data transfers
- Forcing libfabric users to FI_MR_LOCAL is not acceptable
  - Some libfabric users rely on the convenience running without FI_MR_LOCAL provides
- Proposal: Update FI_MR_LOCAL definition
  - The section: When FI_MR_LOCAL is zero, applications are not required to register data buffers before using them for local operations (e.g. send and receive data buffers). The desc parameter into data transfer operations will be ignored in this case, unless otherwise required (e.g. se FI_MR_HMEM). It is recommended that applications pass in NULL for desc when not required.
  - Change to: When FI_MR_LOCAL is zero, applications are not required to register data buffers before using them for local operations (e.g. send and receive data buffers). Passing a NULL desc parameter indicates this behavior. Passing a valid desc parameter allows an application to use a registered buffer.

- FI_MR_HYBRID + ODP with dynamic MRs will work regardless of MR cache behavior
  - MR will always be valid until fi_close is called

~~Hybrid MR Mode~~ – original proposal

- Issue: For persistent, dynamic MRs, the MR must be reused for performance reasons
  - Relying on MR cache must be avoided
    - MR cache will evict entries on munmap/free
  - Problematic for providers running without FI_MR_LOCAL
    - Current definition of running without FI_MR_LOCAL prevents users from passing MR descriptor for local data transfers
- Forcing libfabric users to FI_MR_LOCAL is not acceptable
  - Some libfabric users rely on the convenience running without FI_MR_LOCAL provides
- Need new API which blends running with and without FI_MR_LOCAL
- Proposal: New MR mode flag FI_MR_HYBRID defined
  - FI_MR_HYBRID definition: When set, applications can optionally provide a pre-allocated MR via the descriptor field for local data transfers. If the descriptor field is NULL, the provider will internally perform memory registration. While internal memory registration may be convenient for users, it may negatively impact performance due to the provider having to potentially track all inflight memory registrations. FI_MR_HYBRIB is mutually exclusive with FI_MR_LOCAL and FI_MR_HMEM. FI_ODP flag is valid for FI_MR_HYBRID.
- FI_MR_HYBRID + ODP with dynamic MRs will work regardless of MR cache behavior
  - MR will always be valid until fi_close is called

Page Size Hint

- Issue: Since the dynamic MR may initially have zero VMAs backing the VA range, the provider may not know the page size to be used for MR
  - x86 page size options: 4 KiB, 2 MiB, and 1 GiB
  - Typically, drivers try to use largest page size possible to optimize IOTLB usage
    - Defaulting to system base page size may negatively impact performance
- Need new way for users to specify page size hint
- Proposal: New MR attr field `odp_page_size`
  - Definition: User provided hint to indicate what page size is requested for an on-demand page registration. Must be power of 2. Provider selected page size will be used if supplied hint is not supported or is zero.

MR Refresh Support for ODP MR

- Issue: Since the dynamic MR could have large regions being mmap and munmap, this can lead to lots of ODP activity
  - Having NIC fault in pages for large region may be slow
- Need way for user to signal to that an ODP MR region should have translations faulted in
  - Avoids having NIC fault in pages
- Proposal: Extended fi_mr_refresh to work with ODP MR
  - Definition: fi_mr_refresh is also supported for ODP MRs (i.e., there is no requirement for FI_MR_MMU_NOTIFY). When fi_mr_refresh is called against a region of an ODP MR, the provider must fault pages in and update translation tables for the corresponding region. If the provider is unable to fault in pages (e.g., no VMA backing the region), fi_mr_refresh will fail and return appropriate errno.

Thank you