Contiguous memory allocation in Linux user-space

Guy Shattah, Christoph Lameter

Linux Plumbers Conference 2017

Mellanox TECHNOLOGIES
Connect. Accelerate. Outperform.™

jumptrading

Contents

- Agenda
  - Existing User-Space Memory Allocation Methods
  - Fragmented Memory vs Contiguous Memory
  - Performance
  - Memory Pinning
  - Proposed Solutions
  - Suggested Implementations
  - Summary

Existing User-space Memory Allocation Methods

- Existing memory allocation techniques
  - On the stack: alloca() / function stack
  - On the heap: malloc()/calloc()/realloc()
  - Memory mapping: mmap(): Anonymous memory or with file descriptor
  - Pre-allocation (i.e. static)

Q: What do these techniques have in common?

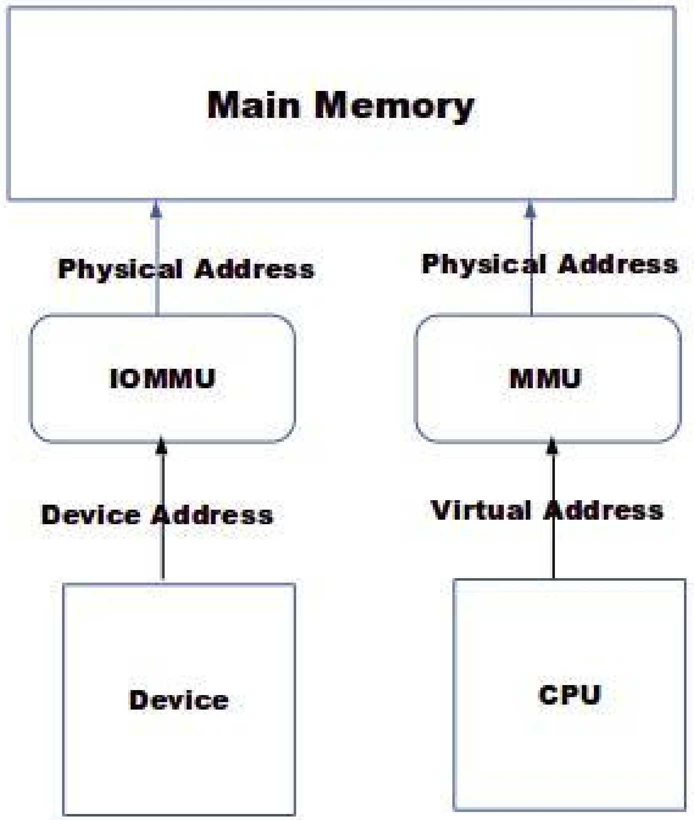
- Fragmented memory
  - Allocated memory resides on separate pages.
  - Virtually: Memory seems to be contiguous to the user.
  - Psychically : Pages are spread over all the ram.

- Contiguous memory:
  - Psychically : A sequence of memory pages without holes.
  - Virtually: the same.
  - No existing userspace API allocates contiguous memory.

Is Fragmented Memory Bad for Us?

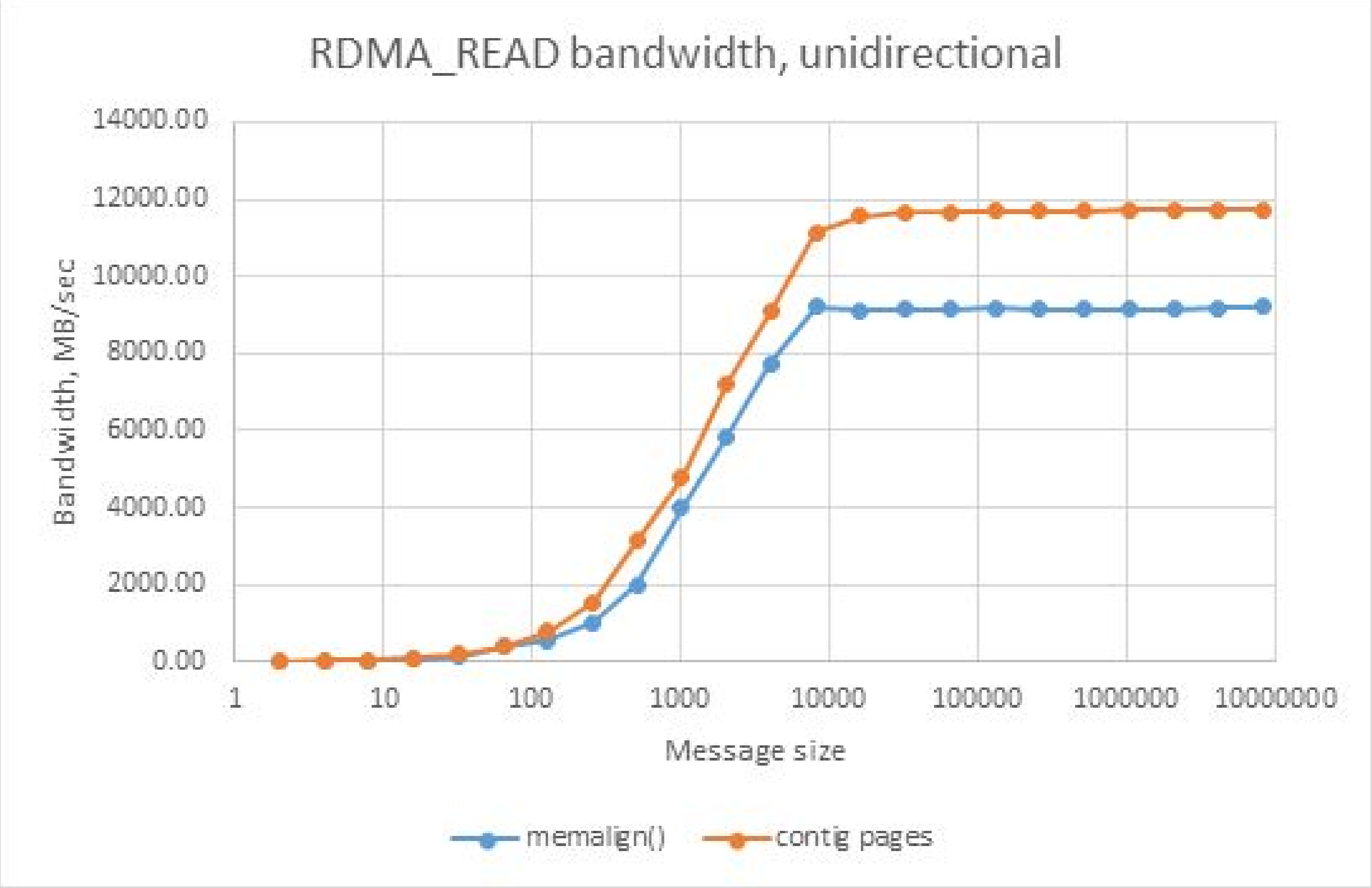
Is Fragmented Memory Bad for Us?

- **Software Solution:**
  - Virtually mapped contiguous areas.
    MMU Maps: Virtual Address → physical address
  - In Linux: Demand paging and reclaim.
- **Hardware Solution:**
  - IOMMU serves as MMU for devices
  - DMA can do vector I/O
    - Gather data from fragmented memory blocks
    - Scatter data to fragmented memory blocks
    - Hence DMA scatter/gather



So why bother?

Performance Comparison: Memalign vs Contiguous



	ib_read_bw		
size	memalign()	contiguous pages	improvement
2	8.74	10.65	22%
4	14.83	23.47	58%
8	35.64	48.52	36%
16	76.10	96.02	26%
32	141.45	195.92	39%
64	406.52	383.80	-6%
128	543.58	780.68	44%
256	1,018.24	1,545.88	52%
512	2,003.87	3,133.44	56%
1,024	4,000.76	4,761.60	19%
2,048	5,837.46	7,216.94	24%
4,096	7,747.90	9,077.66	17%
8,192	9,224.03	11,140.40	21%
16,384	9,109.37	11,561.70	27%
32,768	9,133.99	11,647.37	28%
65,536	9,133.65	11,662.10	28%
131,072	9,179.69	11,694.01	27%
262,144	9,150.13	11,691.50	28%
524,288	9,149.54	11,706.13	28%
1,048,576	9,149.36	11,714.85	28%
2,097,152	9,161.54	11,715.61	28%
4,194,304	9,176.27	11,716.65	28%
8,388,608	9,199.97	11,716.78	27%

Mellanox ConnectX-5 Ex, EDR, back-to-back
Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz
MLNX_OFED_LINUX-4.1-4.0.8.0

Requirement: Contiguous Memory Area

- Contiguous Memory Area
  - Scatter/gather has performance issues.
  - Better cache hits.

Memory pining

Memory Pinning and Compaction Thread

- DMA Operations Typically Require Memory Pinning
  - Memory pinning prevents kernel from:
    - Swapping out memory.
    - Relocation of pages [change mapping of Virtual page -> physical page]
    - Replacing small pages with huge pages (THP)
    - Hot Plug
    - Other compaction thread issues?

- How is memory pinning implemented?
  - Increasing the RefCount on a page struct .
    - Userspace - Using sub-system specific API
    - Kernel - memory registration in the RDMA subsystem.

kernel needs to be built with the CONFIG_HUGETLBFS (present under "File systems") and CONFIG_HUGETLB_PAGE

Huge Pages

Solution #1: Using Huge / Giant Pages

- Huge Pages:
  - mmap using the MAP_HUGETLB flag
  - mmap fails if no huge pages are available.
  - Memory continuity is guaranteed (within a single page).
  - Memory has to be divided to several memory pools (4K,2M,1G on x64)
  - Requires Memory reservation.

Advantage:

- Allocated memory is less fragmented:

- Disadvantage :
  - Pool allocation requires root-user intervention.
  - Pool size has to be pre-determined.
  - Memory continuity is not guaranteed (over multiple pages).

kernel needs to be built with the CONFIG_HUGETLBFS (present under "File systems") and CONFIG_HUGETLB_PAGE

Transparent Huge Pages

Solution #2: Allocating Huge Pages On the Fly

- Transparent Huge Pages (THP):
  - mmap with a hint from madvice (MADV_HUGEPAGE flag) instructs the kernel to try construction of huge pages on the fly.
  - Fall back to small pages when huge pages not available.
- Advantages:
  - Doesn’t require root-user intervention.
  - Doesn't require memory pool reservation.
- Disadvantage :
  - THP Allocation isn’t guaranteed to succeed (but there is a fallback).
  - Memory continuity is not guaranteed (over multiple pages).
  - Background CPU work: Consolidate small pages and replace small ones to huge ones.
  - Transparent huge pages are for performance optimization only.

The Continuous Memory Allocator

Implementation of Contiguous Memory Allocator in User Space Using Huge Pages

- Availability:
  - Huge pages might not be configured on the machine.
  - When configured – pool is limited.

- Non-continuity:
  - Allocation of multiple huge pages does not guarantee continuity.

The Continuous Memory Allocator

Solution #3: Specific HW support – ARMv8-A

- Contiguous block entries.
  - ARMv8-A architecture provides a feature known as contiguous block entries - efficiently uses TLB space.
  - Each TLB entry contains a *contiguous bit*. When set, bit signals to TLB that it can cache a single entry covering translations for multiple blocks.
  - The TLB can cache one entry for a defined range of addresses. Makes it possible to store a larger range of Virtual Addresses within the TLB than is otherwise possible.
- The contiguous blocks must be adjacent and correspond to a contiguous range of Virtual Addresses.
  - 16 × 4KB adjacent blocks giving a 64KB entry with 4KB granule.
  - 128 × 16KB adjacent blocks giving a 2MB entry for L3 descriptors when using a 16KB granule.
  - 32 × 64Kb adjacent blocks giving a 2MB entry with a 64KB granule.
  - 32 × 32MB adjacent blocks giving a 1GB entry for L2 descriptors

The Continuous Memory Allocator

Solution #4: Allocating Physically contiguous Memory on Bootup

- CMA: Contiguous Memory Allocator
  - Unavailable for user-mode.
  - Kernel code can request allocation of contiguous memory.
  - CMA requires memory reservation during machine startup.
  - CMA has to be integrated with the DMA subsystem:
  - CMA lets moveable pages use reserved area. Prioritizes clients with special needs.
  - GCMA (Guaranteed CMA) - an improvement over CMA (latency, moving other process' pages)
    - Uses a client to allocate and vacate memory.

- Issues:
  - Memory reservation technique.
  - Kernel-Space only.
- **Would it be possible to implement a user space API for CMA?**

Proposed Solution

The mmap MAP_CONTIG flag

- Flag instructs mmap to allocate contiguous memory
- Idea was first introduced 13 years ago
  (in IEEE Std 1003.1, 2004 Edition as POSIX_TYPED_MEM_ALLOCATE_CONTIG.)
  - Originally implemented on small number of devices of which most are embedded devices, e.g., BlackBerry.
  - Has since been forgotten.

Proposed solution

The mmap MAP_CONTIG flag

- Populates all pages tables for a mapping
  - ✓ Hence implies MAP_POPULATE

  - Anonymous memory allocation:

    p = mmap(0, size, PROT_READ|PROT_WRITE, MAP_PRIVATE | MAP_ANONYMOUS | MAP_CONTIG, -1, 0);

  - Or with a file descriptor:

    int fd = open("/dev/zero", O_RDWR);
    p = mmap(0, size, PROT_READ|PROT_WRITE, MAP_PRIVATE | MAP_CONTIG, fd, 0);

Suggested Implementation

The mmap MAP_CONTIG flag – Suggested Implementations

- System tries to use largest available memory blocks to construct contiguous memory area.

- Use case: user requested 3,072Kb (3M)
  - On supported machines, where it is possible to allocate contiguous memory areas larger than a single page:
    - Look for contiguous memory block of 4,096Kb, put the spare 1,024 back in the pool (256 pages, 4Kb each).
  - Divide request:
    - Where larger allocation not supported:
    - i.e.: Allocate 2,048 (one huge page) and than allocate multiple small pages which sum to 1,024.

- Perhaps existing defragmentation passes could attempt to make those areas as contiguous as possible when the pass attempts to constructs THPs?

Summary

- Suggested Implementations
  - Allocate contiguous areas larger than a single page (on supported architectures).
  - Improve existing defragmentation passes to help make memory areas as contiguous as possible when the pass attempts to constructs THPs.

- Additional Suggestions
  - Add userspace api for CMA.
  - Add madvise hint to suggest memory should be contiguous.

Q&A

Discussion

Thank You

Mellanox
TECHNOLOGIES
Connect. Accelerate. Outperform.™