Bulk Moving Mechanism on LRU for DRM/TTM

Huang Rui (Ray Huang)

<ray.huang@amd.com>

AGENDA

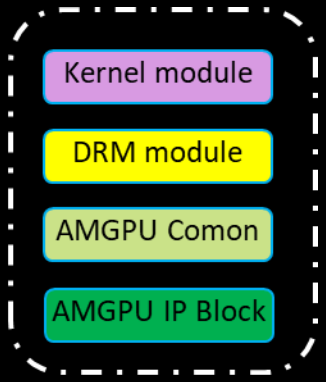
- Background
- LINUX® GPU Kernel
- GPUVM - GART memory setup (1-level)
- GPUVM - 4-level paging
- Per-VM Buffer Object
- Eviction (Buffer Migration)
- VM Key List Definitions
- New List Operation for Bulk Moving
- LRU Policy for Buffer Migration in Bulk Moving
- Bulk Moving Approach in TTM
- Bulk Moving use case in AMDGPU
- Performance Improvement Data
- Contribution
- Q & A

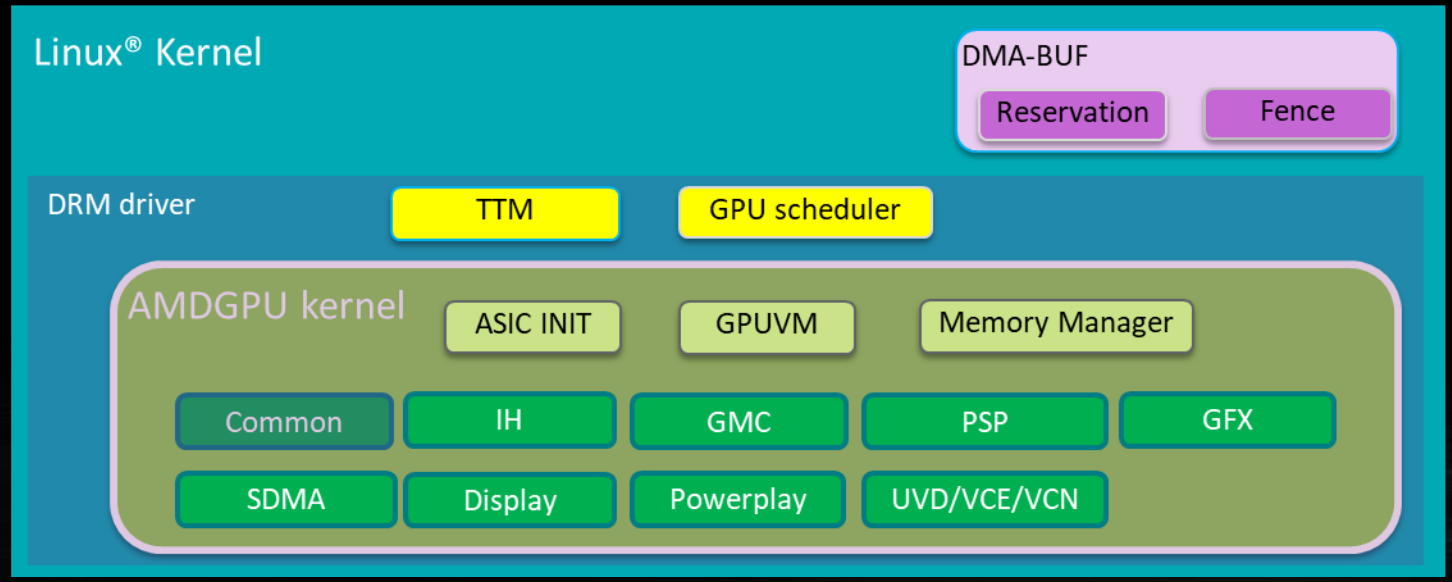
Background

- Who am I?
  - My name is Huang Rui (Ray), I am from AMD Linux® graphic driver team and focus on kernel driver and libdrm for new ASIC bring-up and new feature development several years.
  - Patch work profile:
    - https://git.kernel.org/pub/scm/linux/kernel/git/rui/linux.git
- Why we proposed the solution of bulk moving?
  - Investigating performance with the F1 2017 game benchmark showed that the application caused a large number of buffers being created
  - The validation and LRU management of these buffers in the TTM and driver infrastructure was found to be non-optimal for this scenario.
  - This led to a redesign of the buffer migration process in the code.
  - This talk demonstrates the practical techniques to efficiently profile and analyze the scenario and identify the design changes needed to address it.

LINUX® GPU Kernel

- Linux® GPU Kernel for AMD
  - Device Init
  - Interrupt Handle
  - GMC (GFX memory controller)
  - PSP (Platform Security Processor)
  - Display
  - GFX
  - SDMA
  - MM block (UVD/VCE/VCN)

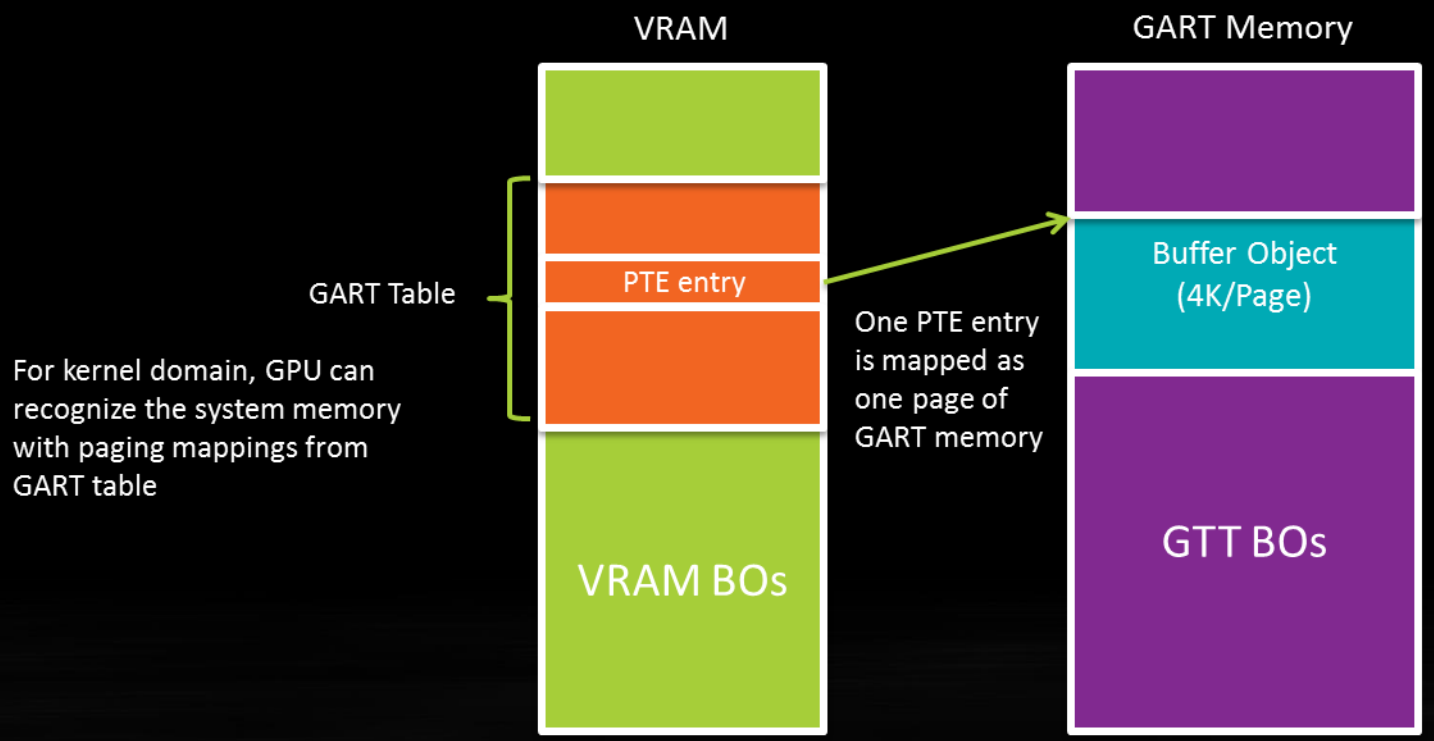




AMD

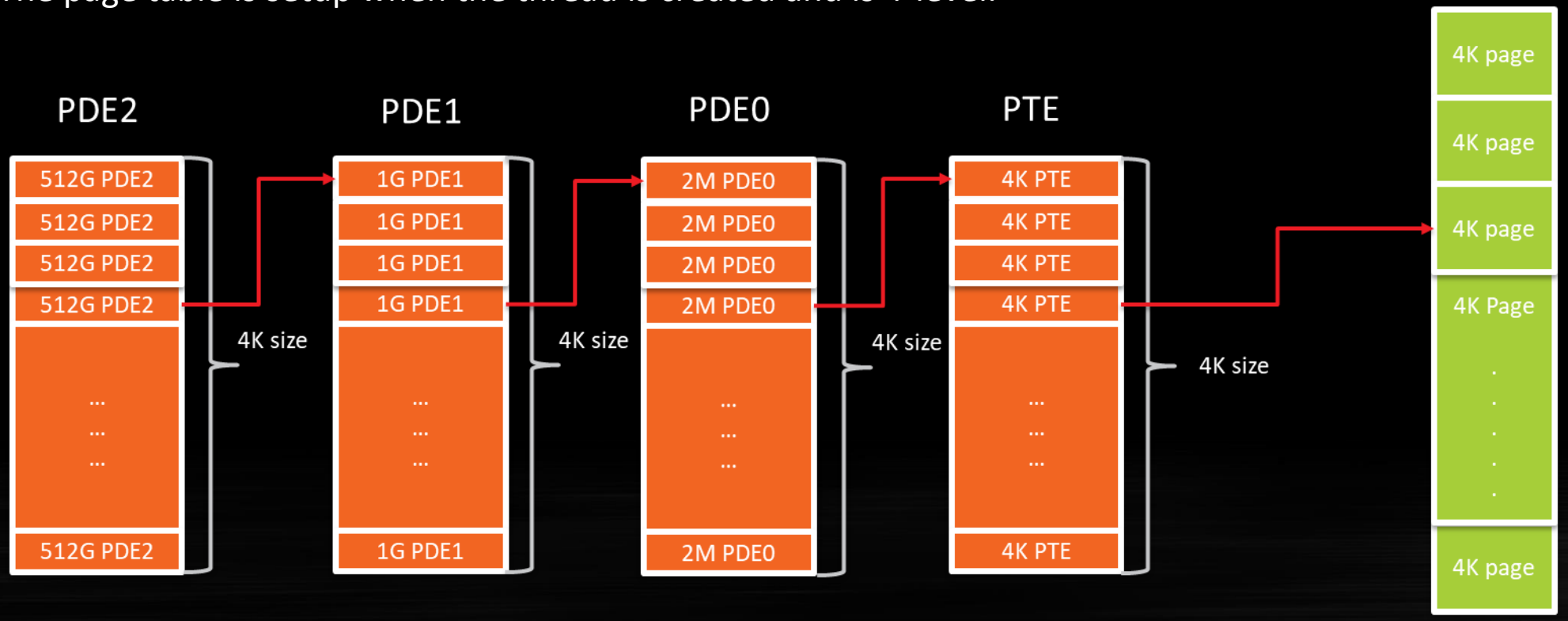
GPUVM - GART memory setup (1-level)

- GART memory is GPU visible system memory
  - Allocate GART table BO in the video memory for mapping to system memory.
  - GPU will read the data from the page table entries in the GART table BO to convert to the physical address (DMA bus address).



GPUVM - 4-level paging

- VMID 0
  - System context domain that only used by kernel mode.
  - GART table is created by 1-level paging (flat page table)
- VMID 1 ~ 15
  - Other context domain that used by user mode.
  - The page table is setup when the thread is created and is 4-level.



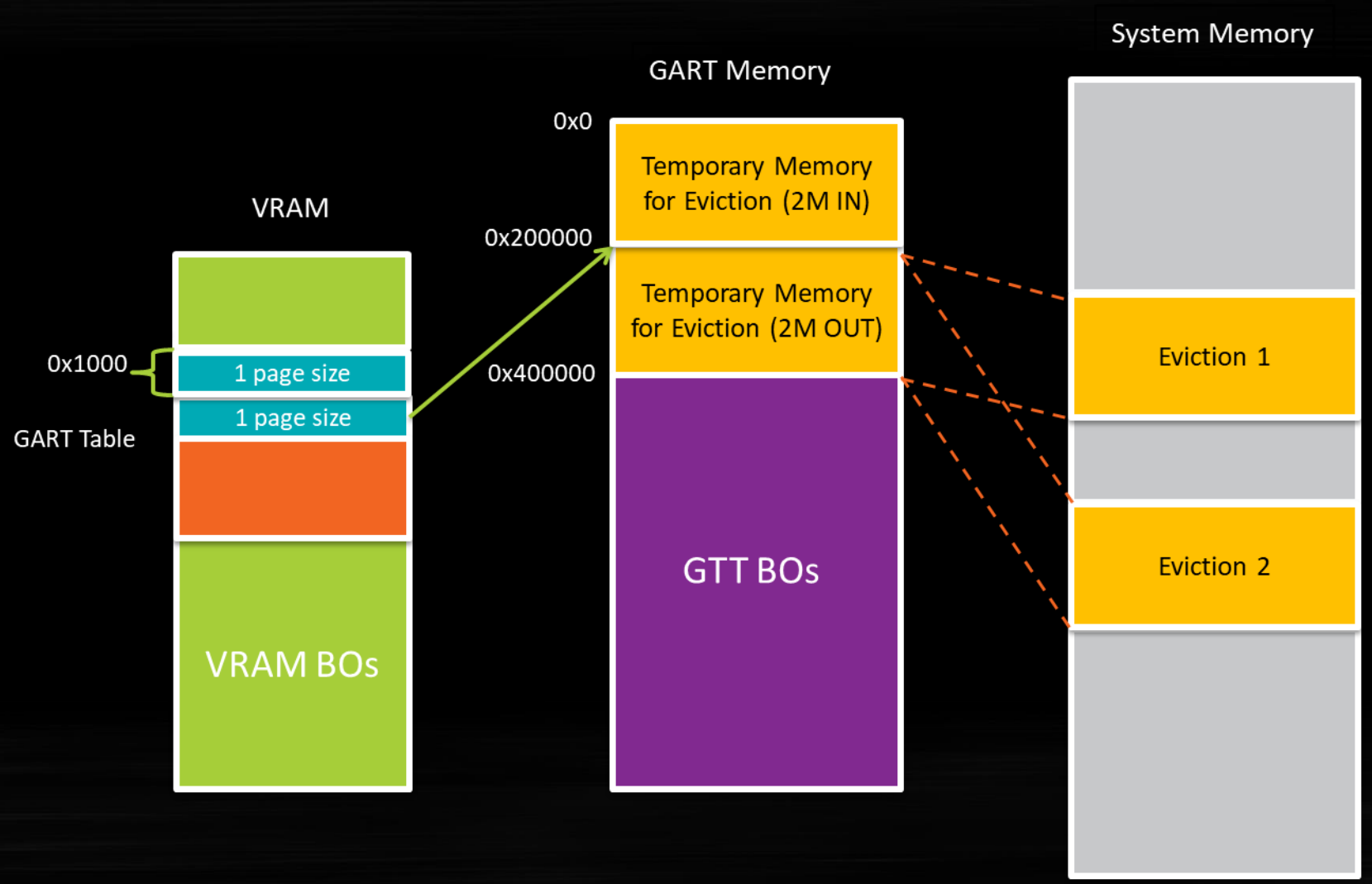
AMD

Per-VM Buffer Object

- History:
  - Each BO (too many) in the BO list that needs be validated during CPU bound games
  - Solution: decrease the work of BO list parser relevant.
- New mechanism (Per-VM) that is to ensure the BO always valid for command submission.
  - Add flag for UMD (Vulkan)
  - Share reservation object with VM root BO
  - Allow eviction and swap out when sharing same reservation
  - Ensure the Per-VM BO always valid

Eviction (Buffer Migration)

- Buffer migration approach:



AMD

VM Key List Definitions

```
233
234         /* BOs who needs a validation */
235         struct list_head        evicted;
236
237         /* PT BOs which relocated and their parent need an update */
238         struct list_head        relocated;
239
240         /* per VM BOs moved, but not yet updated in the PT */
241         struct list_head        moved;
242
243         /* All BOs of this VM not currently in the state machine */
244         struct list_head        idle;
245
246         /* regular invalidated BOs, but not yet updated in the PT */
247         struct list_head        invalidated;
248         spinlock_t              invalidated_lock;
249
250         /* BO mappings freed, but not yet updated in the PT */
251         struct list_head        freed;
252
```

New List Operation For Bulk Moving

```
216
217 /**
218  * list_bulk_move_tail - move a subsection of a list to its tail
219  * @head: the head that will follow our entry
220  * @first: first entry to move
221  * @last: last entry to move, can be the same as first
222  *
223  * Move all entries between @first and including @last before @head.
224  * All three entries must belong to the same linked list.
225  */
226 static inline void list_bulk_move_tail(struct list_head *head,
227                                        struct list_head *first,
228                                        struct list_head *last)
229 {
230         first->prev->next = last->next;
231         last->next->prev = first->prev;
232
233         head->prev->next = first;
234         first->prev = head->prev;
235
236         last->next = head;
237         head->prev = last;
238 }
239
```

LRU Policy for Buffer Migration in Bulk Moving

- Least Recently Used (LRU) algorithm is used for TTM on eviction (buffer migration)

```
236 static void ttm_bo_bulk_move_set_pos(struct ttm_lru_bulk_move_pos *pos,
237                                        struct ttm_buffer_object *bo)
238 {
239         if (!pos->first)
240                 pos->first = bo;
241         pos->last = bo;
242 }
243
244 void ttm_bo_move_to_lru_tail(struct ttm_buffer_object *bo,
245                              struct ttm_lru_bulk_move *bulk)
246 {
247         dma_resv_assert_held(bo->base.resv);
248
249         ttm_bo_del_from_lru(bo);
250         ttm_bo_add_to_lru(bo);
251
252         if (bulk && !(bo->mem.placement & TTM_PL_FLAG_NO_EVICT)) {
253                 switch (bo->mem.mem_type) {
254                 case TTM_PL_TT:
255                         ttm_bo_bulk_move_set_pos(&bulk->tt[bo->priority], bo);
256                         break;
257
258                 case TTM_PL_VRAM:
259                         ttm_bo_bulk_move_set_pos(&bulk->vram[bo->priority], bo);
260                         break;
261                 }
262                 if (bo->ttm && !(bo->ttm->page_flags &
263                                  (TTM_PAGE_FLAG_SG | TTM_PAGE_FLAG_SWAPPED)))
264                         ttm_bo_bulk_move_set_pos(&bulk->swap[bo->priority], bo);
265         }
266 }
267 EXPORT_SYMBOL(ttm_bo_move_to_lru_tail);
```

Bulk Moving Approach in TTM

```
268
269 void ttm_bo_bulk_move_lru_tail(struct ttm_lru_bulk_move *bulk)
270 {
271         unsigned i;
272
273         for (i = 0; i < TTM_MAX_BO_PRIORITY; ++i) {
274                 struct ttm_lru_bulk_move_pos *pos = &bulk->tt[i];
275                 struct ttm_mem_type_manager *man;
276
277                 if (!pos->first)
278                         continue;
279
280                 dma_resv_assert_held(pos->first->base.resv);
281                 dma_resv_assert_held(pos->last->base.resv);
282
283                 man = &pos->first->bdev->man[TTM_PL_TT];
284                 list_bulk_move_tail(&man->lru[i], &pos->first->lru,
285                                     &pos->last->lru);
286         }
287
288         for (i = 0; i < TTM_MAX_BO_PRIORITY; ++i) {
289                 struct ttm_lru_bulk_move_pos *pos = &bulk->vram[i];
290                 struct ttm_mem_type_manager *man;
291
292                 if (!pos->first)
293                         continue;
294
295                 dma_resv_assert_held(pos->first->base.resv);
296                 dma_resv_assert_held(pos->last->base.resv);
297
298                 man = &pos->first->bdev->man[TTM_PL_VRAM];
299                 list_bulk_move_tail(&man->lru[i], &pos->first->lru,
300                                     &pos->last->lru);
301         }
302
303         for (i = 0; i < TTM_MAX_BO_PRIORITY; ++i) {
304                 struct ttm_lru_bulk_move_pos *pos = &bulk->swap[i];
305                 struct list_head *lru;
306
307                 if (!pos->first)
308                         continue;
309
310                 dma_resv_assert_held(pos->first->base.resv);
311                 dma_resv_assert_held(pos->last->base.resv);
312
313                 lru = &pos->first->bdev->glob->swap_lru[i];
314                 list_bulk_move_tail(lru, &pos->first->swap, &pos->last->swap);
315         }
316 }
317 EXPORT_SYMBOL(ttm_bo_bulk_move_lru_tail);
318
```

Bulk Moving Use Case in AMDGPU

- Legacy approach
  - AMDGPU driver will move all PD/PT and Per-VM BOs into idle list. Then move all of them on the end of LRU list one by one. The result of this is that many BOs are moved to the end of the LRU again and again, which has a serious impact on performance.
- Bulk Moving
  - Collect all PD/PT and Per-VM BOs and bulk move them to the end of LRU list one time instead of one by one. This will reduce cost during the buffer moving.

Performance Improvement Data

GPU: Radeon™ RX Vega

Video Memory: 8G

System Memory: 16G

OS: Ubuntu 18.04 LTS

	The Talos Principle (Vulkan)	Clpeak (OpenCL™)	BusSpeedReadback (OpenCL™)
Original	147.7 FPS	76.86 us	0.319 ms(1k) 0.314 ms(2K) 0.308 ms(4K) 0.307 ms(8K) 0.310 ms(16K)
Original + WA (don't move PT BOs on LRU)	162.1 FPS	42.15 us	0.254 ms(1K) 0.241 ms(2K) 0.230 ms(4K) 0.223 ms(8K) 0.204 ms(16K)
Bulk Move	163.1 FPS	40.52 us	0.244 ms(1K) 0.252 ms(2K) 0.213 ms(4K) 0.214 ms(8K) 0.225 ms(16K)

Bulk move will get the highest FPS and lowest latency

Contribution

- Christian König <Christian.Koenig@amd.com>
  - He raised the original idea of bulk moving for the optimization of buffer migration.
  - I worked with him to deliver the completed solution in the kernel driver.

- Alex Deucher <alexander.deucher@amd.com>
  - Maintain and lead AMDGPU kernel driver to support the solution upstream.
  - Actively review and refine the quality of Linux® AMDGPU driver stack.

Thank You and Q&A

DISCLAIMER AND ATTRIBUTIONS

DISCLAIMER

The information contained herein is for informational purposes only, and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD’s products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale. GD-18

AMD