Using process address space on the GPU

Jérôme Glisse

September 2013

Outline

Motivation

Memory management

Hardware solution

Software solution

On the graphic side

The middle man

```
void vec_add(float *vr, float *va, float *vb, unsigned s
{
  gpu_bo *gpu_a, *gpu_b, *gpu_r;


  gpu_a = gpu_alloc(sizeof(float) * size);
  gpu_b = gpu_alloc(sizeof(float) * size);
  gpu_r = gpu_alloc(sizeof(float) * size);
  gpu_upload(gpu_a, va, sizeof(float)*size);
  gpu_upload(gpu_b, vb, sizeof(float)*size);

  gpu_shader_execute(vec_add_shader, gpu_r, gpu_a, gpu_b


  gpu_download(gpu_r, vr, sizeof(float)*size);

}
```

Cut the middle man

```
void vec_add(float *vr, float *va, float *vb, unsigned s
{
  gpu_shader_execute(vec_add_shader, vr, va, vb, size);
}
```

Memory management inherently concurrent

Pagetable concurrent update

- Userspace map/unmap.
- Memory reclaim.
- Deduplication (KSM).
- Migration (NUMA architecture).
- Compaction.
- ...

The more memory pressure the more concurrent update.

Synchronization

Pagetable update

- Concurrency imply no serialization.
- Pagetable is synchronization point.
  - Save pte (pagetable entry).
  - Perform job (swapping, migrating, ...).
  - Check pte is same if so update, otherwise back off.
- Page backing an address might change at any time.
- Pinning would defeat memory management.
- Taking reference defeat memory management too.

Notify me

Virtualization

- Host kernel has global overview.
- Guest kernel has local overview.
- Efficiency needs communication both ways.

Meet the mmu notifier API

- Bracket large pagetable update with range start/end callback.
- Allow proper youngness accounting
- ...
- No serialization, concurrent notification on overlapping range.

Moving target

Catch me if you can

- Pagetable is a moving target.
- Page behind an address might change at any time.
- Playing catchup.

Mirroring process address space requirements

- Both CPU and GPU use same page for same address.
- Concurrent access by both (in most cases at least).
- No pinning.
- No references.
- ...

Road forks

- Hardware solution.
- Software solution.

Meet the new middle man, IOMMU

IOMMU original motivation

- I/O protection and isolation (security).
- Easy remapping and scatter gather.
- Middle man between device and system memory.
- Virtualization and device isolation.
- ...

IOMMU unique position

- Middle man.
- Close CPU (same die since memory controller got merge).
- IOMMU can be tie to specific CPU.
- IOMMU can understand CPU pagetable format.
- IOMMU can walk any process pagetable and relay information.

IOMMU and the PCIE ATS/PASID case

PASID PCIE specification

- PASID = Process address space identifiant.
- Unique ID associated with a process and thus a pagetable.
- Device can tell IOMMU which process they are interested in.
- ...

ATS PCIE specification

- ATS = address translation service.
- IOMMU translate virtual address to physical address.
- Device can cash in TLB IOMMU translation.
- Device must offer TLB flush/invalidation mechanism.
- Carry over protection (read, write, execute, ...)
- ...

How device make use of address translation

Device pagetable case

Device manage its own pagetable, a bit flag in each entry tell if the address should use IOMMU address translation or not.

- Flexible, can mix VRAM and SRAM.
- Flag on any level of the pagetable (TLB cache optimization).
- Complex TLB cache and memory controller.

Device use aperture case

- Address inside (or outside) aperture use ATS/PASID.
- Address outside (or inside) usual device memory controller.
- Not flexible.
- Coarse granularity.
- Simpler memory controller and TLB cache.

The limits of hardware solution

Partial solution

- One way, device asking pagetable controlled by CPU.
- Can not copy temporarily data into VRAM.

IOMMUv2 linux limitations

- IOMMU use empty pagetable during CPU pagetable update :
  - Add latency
  - Can be very frequent with memory pressure.
  - Solvable by adding a flag to CPU pagetable entry

GPUs are VRAM junkies

Never fast enough, never big enough

- GPU crave for big bandwidth and low latency.
- VRAM up to 200GB/s vs 20GB/s for system memory.
- Up to 4 times lower latency with VRAM.
- Such bandwith unlikely to happen soon for system memory.

Software solution

- Use of VRAM require change to linux kernel.
- Kernel must know about VRAM and where are things.
- Not exclusive with hardware solution. Software can handle VRAM object only.

Challenges

Serialize

- Device pagetable update can not be done from CPU.
- Serialization between CPU and device pagetable.
- Pagetable coherency (same page backing same address).
- Serialization might badly hurt performances.

Minimize

- Minimize changes needed to core mm codepath.
- Avoid disruptive changes.
- Do not break linux API (like memory cgroup).
- Add another point of failure for process or file corruption.
- ...

It is happening

- Partnership between NVidia and Red Hat.
- Working prototype.
- Should be soon send as an RFC upstream.

Open and useful to other

Generic device agnostic API. No assumption on what the device do.

Useful for any hardware with :

- Pagetable and support true pagefault.
- Support read only page entry.
- Preemptable workload.
- Page size on the device can be different from the CPU.

Best to have :

- Dirty accounting to avoid over dirtying.
- Fast preemption.
- Fast device page table update.
- ...

Graphic use case

- Texture upload without memcpy.
- Next sparse texture extension automatic load from disk.
- GL using process address for seemless compute shader.
- Cache policy need custom API (UC, WC, ...).