Upstreaming ION Features: Issues that remain

Linux Plumbers 2015
ION and its upstreamable features
ION? What??

- Memory Manager written for Android
- Devices have different ‘constraints’ on the memory they can access
- Allocates from different pools or types of memory
  - Cached or uncached buffers
- Manages Caching and Shares via dma-buf
- Primarily for Graphics, but other users as well
- Generally Allocator, and dma_buf exporter
ION Features Upstreaming:

- Allocation
- ‘correct’ way of mapping & coherency management
  - Cached v/s uncached?
- Sharing
  - done with dma-buf; ION only as an exporter though, doesn’t support importing
Allocation
ION Allocation

- ‘what’s not right with ION allocation’
  - Userspace is required to understand device buffer constraints, and available heaps
- Upstream friendly way:
  - is to have allocation ‘sorted out’ in a way that applications don’t have to know exactly the hardware underneath
Allocation: Upstreaming ION needs

- Same buffer, different devices, different constraints
  - need a way of defining and sharing device constraints in kernel
  - use device constraints to help with choosing allocator
Constraint-aware Allocation: Possibilities

- Two ways discussed
  - Migration of pages
  - Delayed Allocation
Allocation: Prerequisites

- For both migration and delayed allocation:
  - Sharing of device constraints
  - All importers must follow the sequence
    - ‘attach()’
      - -> map() -> {USE} -> unmap() and repeat
    - detach()
Allocation: Constraint Sharing

- Share constraints, and match
  - Constraint sharing is done via attach->dev->dma_parms at each device attach
    - Rob posted [1] for adding some information in dma_parms, and I took them as part of my constraint-sharing patches [2]
  - Constraint matching for devices can be done in multiple ways

[1]: https://lists.linaro.org/pipermail/linaro-mm-sig/2012-July/002250.html
Allocation: Migration of pages

- Migration of pages [1]
  - suggested at ELCE 2013 as ‘transparent backing store migration’
    - Migrate backing storage at attach time if required
    - Seems a pretty decent Idea!
    - Aggressive caching, and/or outright pinning by exporters probably makes it quite difficult to implement
  - Unfortunately, no patches seem to have been submitted :(

Video of the talk at: https://www.youtube.com/watch?v=w_1jP1CSfqM
Allocation: Delayed Allocation

- Delay Allocation to as late as possible
  - Allocate at the *first* map() call after _ALL_ interested importers have attached to the dma_buf
- It’s problematic for Android
  - buffers may be passed around between applications, each sharing the buffer with one device
  - *Possible workaround*: have dummy-devices, correlate to Android gralloc most common usage types
Allocation: Recap of cenalloc

- 2 Dimensions
  - ‘Generic’ exporter interface
  - Delayed Allocation based on constraint-sharing

- Generic Exporter interface
  - miscdevice /dev/cenalloc created, which acts as dma-buf exporter
  - allocators registered with this device at boot time, corresponding to memory constraints that they can allocate for
  - create_buffer, and share with importers
Allocation: Recap of cenalloc

- **Delayed Allocation**
  - each importer sets the required constraints in its dev-
    >dma_parms, and calls attach()
  - actual allocation happens at first map_attachment() call to the
    buffer, based on the current constraints of the buffer
Allocation: Recap of cenalloc

- Cenalloc - RFC review received
  - Initially 4 patch series, with 2 about constraint sharing and 2 adding cenalloc
    - Split into constraint-sharing and cenalloc series separately
    - Constraint-sharing seems to be stuck :(  
  - miscdevice /dev/cenalloc not liked by many
    - should be easy to convert into dma-buf helpers
Coherency, Mapping... and other issues
Other issues: ION dma layer abuse

- ION Uses dma sync APIs without map first
  - cached buffers shouldn’t need to sync at alloc time.
    - no guaranteed enforcement
  - can uncached be replaced with CMA totally?
Coherency: Issues

- “ION stops trying to do anything special with coherency”

- “Same-device coherency management”
  - devices can use both coherent and non-coherent transactions.
  - When mapping same memory multiple times, don’t want to flush cpu caches multiple times
    - no mechanism in dma api to handle this, so devices have caches and stuff to avoid the cpu cache flush
Discussion Topics
Discussion Topics

- Allocation and Constraint-sharing based approaches
- Mapping
- Coherency
Constraint Sharing and Allocation: Discussion

- Are we ok with this idea of constraint-sharing based allocation (delayed and/or migration)?
  - **YES:**
    - I’ll re-submit constraint-sharing patches
    - changes on drm and v4l for following correct attach / map / unmap sequence, and support for delayed allocation
    - rework cenalloc patch as dma-buf helpers
  - **NO:** <begin speculation>
    - keep the ION way of userspace allocation decision, and help to address other issues to get ION moved to mainline?
    - <end speculation>
Map: suggestion

- “Map sg-table without copying data”
  - with this, exporter can test if attachments work by test-attaching stuff
  - Bit inefficient, but exporters can cache the mapped sgttable, if they wanted.
Coherency: Discussion

- “ION stops trying to do anything special with coherency”
  - Call standard dma APIs for coherency
  - BIG performance issues for Android, since uncached page pooling won’t be possible

- “Same-device coherency management”
  - Need Core DMA API extensions to allow no cpu-cache-flushing while mapping same memory multiple times
Recent ION patches for discussion

- ION’s interface has gained more users
- DT bindings for ION
  - “staging: ion: Add generic ion-physmem driver” from Andrew Andrianov [https://lkml.org/lkml/2015/6/22/323]
- Add default dev for cma heap
  - to allow to add another device for cma heap allocation
  - “staging: ion: Add a default struct device for cma heap” from Feng Tang [https://lkml.org/lkml/2015/8/6/212]
References

- https://lwn.net/Articles/565469/