I915 GuC Submission/DRM Scheduler Section

Upstream plan

For upstream the overall plan for landing GuC submission and integrating the i915 with the DRM scheduler is:

- Merge basic GuC submission
    - Basic submission support for all gen11+ platforms
    - Not enabled by default on any current platforms but can be enabled via modparam enable_guc
    - Lots of rework will need to be done to integrate with DRM scheduler so no need to nit pick everything in the code, it just should be functional, no major coding style / layering errors, and not regress execlists
    - Update IGTs / selftests as needed to work with GuC submission
    - Enable CI on supported platforms for a baseline
    - Rework / get CI heathly for GuC submission in place as needed
- Merge new parallel submission uAPI
    - Bonding uAPI completely incompatible with GuC submission, plus it has severe design issues in general, which is why we want to retire it no matter what
    - New uAPI adds I915_CONTEXT_ENGINES_EXT_PARALLEL context setup step which configures a slot with N contexts
    - After I915_CONTEXT_ENGINES_EXT_PARALLEL a user can submit N batches to a slot in a single execbuf IOCTL and the batches run on the GPU in paralllel
    - Initially only for GuC submission but execlists can be supported if needed
- Convert the i915 to use the DRM scheduler
    - GuC submission backend fully integrated with DRM scheduler
        - All request queues removed from backend (e.g. all backpressure handled in DRM scheduler)
        - Resets / cancels hook in DRM scheduler
        - Watchdog hooks into DRM scheduler
        - Lots of complexity of the GuC backend can be pulled out once integrated with DRM scheduler (e.g. state machine gets simplier, locking gets simplier, etc...)
    - Execlists backend will minimum required to hook in the DRM scheduler
        - Legacy interface
        - Features like timeslicing / preemption / virtual engines would be difficult to integrate with the DRM scheduler and these features are not required for GuC submission as the GuC does these things for us
        - ROI low on fully integrating into DRM scheduler
        - Fully integrating would add lots of complexity to DRM scheduler
    - Port i915 priority inheritance / boosting feature in DRM scheduler
        - Used for i915 page flip, may be useful to other DRM drivers as well
        - Will be an optional feature in the DRM scheduler
    - Remove in-order completion assumptions from DRM scheduler
        - Even when using the DRM scheduler the backends will handle preemption, timeslicing, etc... so it is possible for jobs to finish out of order
    - Pull out i915 priority levels and use DRM priority levels
    - Optimize DRM scheduler as needed

TODOs for GuC submission upstream

- Need an update to GuC firmware / i915 to enable error state capture
- Open source tool to decode GuC logs
- Public GuC spec

New uAPI for basic GuC submission

No major changes are required to the uAPI for basic GuC submission. The only change is a new scheduler attribute: I915_SCHEDULER_CAP_STATIC_PRIORITY_MAP. This attribute indicates the 2k i915 user priority levels are statically mapped into 3 levels as follows:

- -1k to -1 Low priority
- 0 Medium priority
- 1 to 1k High priority

This is needed because the GuC only has 4 priority bands. The highest priority band is reserved with the kernel. This aligns with the DRM scheduler priority levels too.

Spec references:

- https://www.khronos.org/registry/EGL/extensions/IMG/EGL_IMG_context_priority.txt
- https://www.khronos.org/registry/vulkan/specs/1.2-extensions/html/chap5.html#devsandqueues-priority
- https://spec.oneapi.com/level-zero/latest/core/api.html#ze-command-queue-priority-t

New parallel submission uAPI

The existing bonding uAPI is completely broken with GuC submission because whether a submission is a single context submit or parallel submit isn't known until execbuf time activated via the I915_SUBMIT_FENCE. To submit multiple contexts in parallel with the GuC the context must be explicitly registered with N contexts and all N contexts must be submitted in a single command to the GuC. The GuC interfaces do not support dynamically changing between N contexts as the bonding uAPI does. Hence the need for a new parallel submission interface. Also the legacy bonding uAPI is quite confusing and not intuitive at all. Furthermore I915_SUBMIT_FENCE is by design a future fence, so not really something we should continue to support.

The new parallel submission uAPI consists of 3 parts:

- Export engines logical mapping
- A 'set_parallel' extension to configure contexts for parallel submission
- Extend execbuf2 IOCTL to support submitting N BBs in a single IOCTL

Export engines logical mapping

Certain use cases require BBs to be placed on engine instances in logical order (e.g. split-frame on gen11+). The logical mapping of engine instances can change based on fusing. Rather than making UMDs be aware of fusing, simply expose the logical mapping with the existing query engine info IOCTL. Also the GuC submission interface currently only supports submitting multiple contexts to engines in logical order which is a new requirement compared to execlists. Lastly, all current platforms have at most 2 engine instances and the logical order is the same as uAPI order. This will change on platforms with more than 2 engine instances.

A single bit will be added to drm_i915_engine_info.flags indicating that the logical instance has been returned and a new field, drm_i915_engine_info.logical_instance, returns the logical instance.

A 'set_parallel' extension to configure contexts for parallel submission

The 'set_parallel' extension configures a slot for parallel submission of N BBs. It is a setup step that must be called before using any of the contexts. See I915_CONTEXT_ENGINES_EXT_LOAD_BALANCE or I915_CONTEXT_ENGINES_EXT_BOND for similar existing examples. Once a slot is configured for parallel submission the execbuf2 IOCTL can be called submitting N BBs in a single IOCTL. Initially only supports GuC submission. Execlists supports can be added later if needed.

Add I915_CONTEXT_ENGINES_EXT_PARALLEL_SUBMIT and drm_i915_context_engines_parallel_submit to the uAPI to implement this extension.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\gpu\rfc\[linux-master][Documentation][gpu][rfc]i915_scheduler.rst, line 138)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: include/uapi/drm/i915_drm.h
        :functions: i915_context_engines_parallel_submit
```

Extend execbuf2 IOCTL to support submitting N BBs in a single IOCTL

Contexts that have been configured with the 'set_parallel' extension can only submit N BBs in a single execbuf2 IOCTL. The BBs are either the last N objects in the drm_i915_gem_exec_object2 list or the first N if I915_EXEC_BATCH_FIRST is set. The number of BBs is implicit based on the slot submitted and how it has been configured by 'set_parallel' or other extensions. No uAPI changes are required to the execbuf2 IOCTL.