

Improving the Linux/DRM GPU scheduler

Google Summer of Code 2018

Nayan Deshmukh

nayan26deshmukh@gmail.com

Mentored by: Christian König



Contents

- About me
- DRM scheduler
- My Project
- Future Work
- Questions



About me

- I did GSoC this year
- I graduated from IIT Kanpur with a bachelors in computer science
- I recently joined Samsung Electronics
- I have previously worked with Mesa community

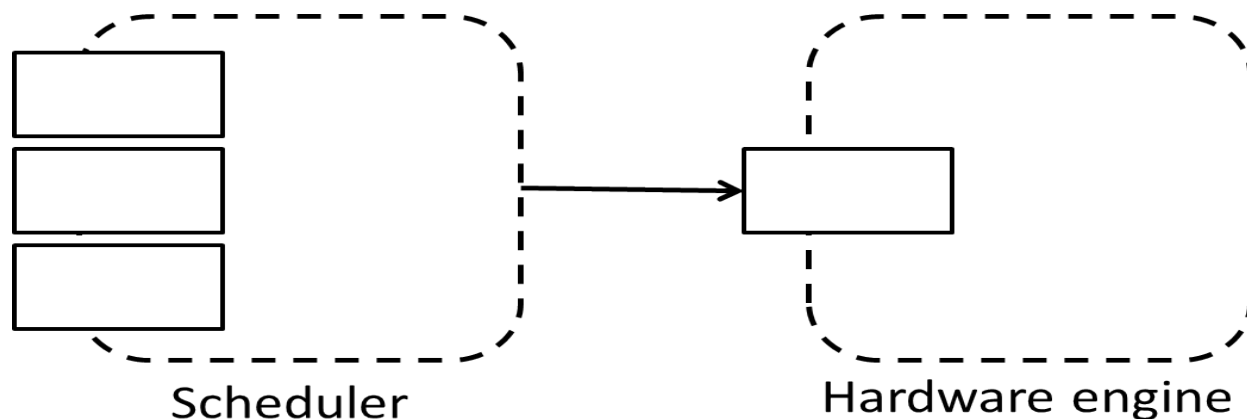


DRM scheduler

- The amdgpu's (AMD's graphics driver) scheduler was shifted to a shared space (now called DRM GPU scheduler) so that the other drivers can reuse the code
- The DRM scheduler is now used by amdgpu, etnaviv (graphics driver for Vivante GPUs), and recently the Broadcom V3D driver.

DRM scheduler

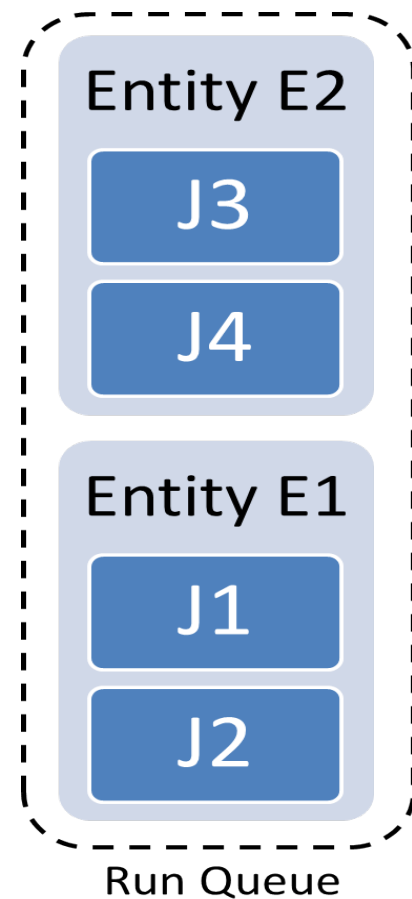
- Job is the basic unit which is executed by the hardware engine



- Scheduler can handle dependencies among jobs
- We can also specify priority among the jobs

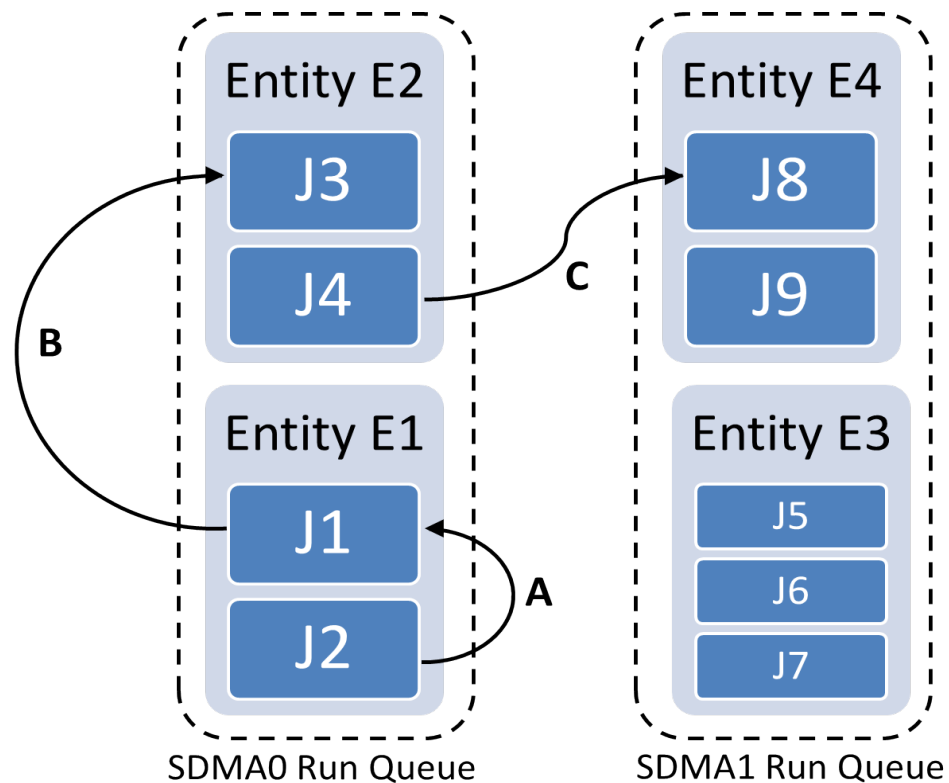
DRM scheduler

- Entity is a dynamic list of jobs
- The jobs in a entity are always scheduled in the order that they were pushed
- Entities are pushed to the run queues



DRM scheduler

- Job depend on other jobs





DRM scheduler

- Each hardware engine has its own scheduler instance
- When an entity is created it is attached to a scheduler and will remain attached to it for the rest of its lifetime
- The jobs from an entity can be scheduled only on one hardware engine
- What if we have multiple copies of the same hardware engine? Can we do load balancing?



My Project

- Implemented shifting of entity from one scheduler to other
- Driver specifies the possible hardware engines during entity init
- We shift the entity when a new job is pushed to it
- Need to identify the cases where shifting is correct and beneficial



My Project

- Three phases
- First phase
 - Understanding the code
 - Added documentation, cleaned up the API
- Second phase
 - Discussed various ideas
 - Decided to go for a small but simple implementation
 - Started writing the code



My Project

- Third phase
 - Completed the code
 - Spent a good amount of time in debugging
 - Upstream the code
- After GSoC
 - Minor improvements to the code
- You can get more details of my project here:
<https://ndesh26.github.io/categories/#gsoc>



Future work

- Have a better criteria for calculating load
- There are more cases where we can shift entities
- We only shift an entity when we push a job to it
- Analyse the performance benefits on real life workloads



Questions?



Thank you