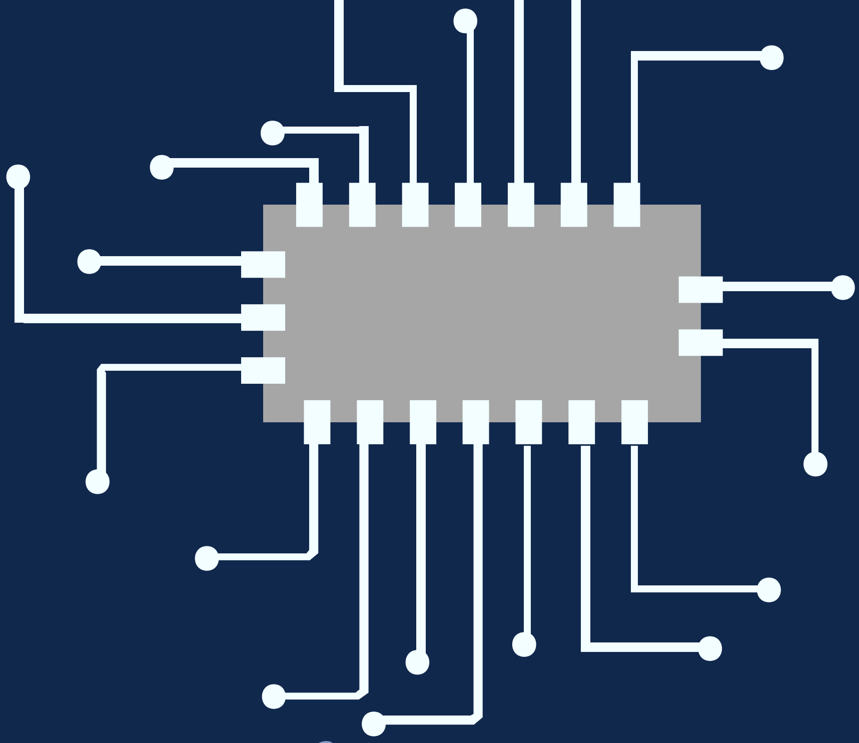
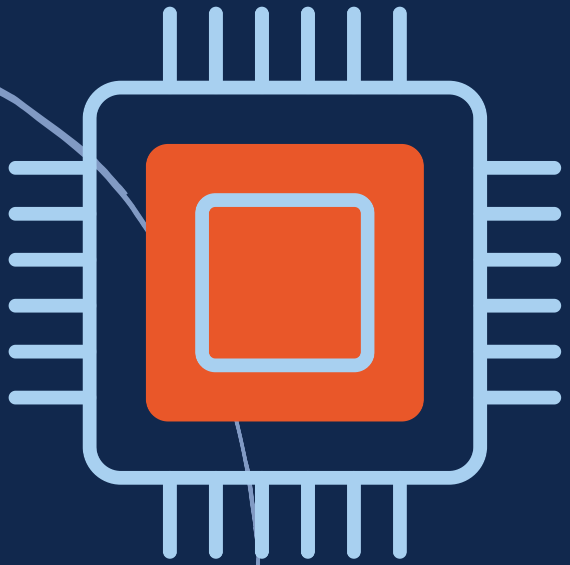
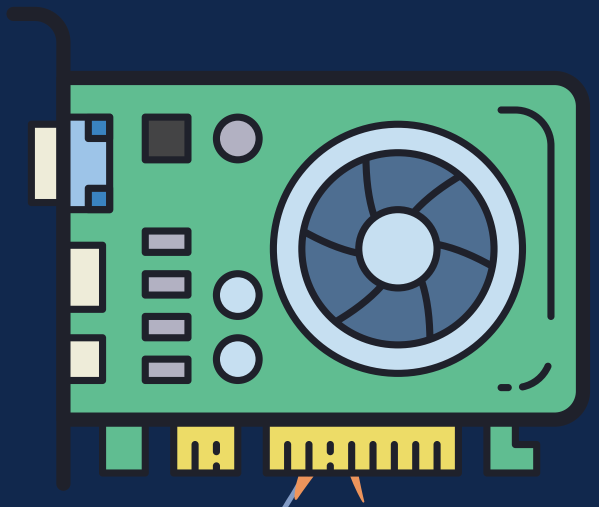


Maximizing the OS throughput with GPU abstractions

Prakhar Varshney





Problem Statement

Modern Operating systems treat GPUs as an I/O device rather than a shared processing unit

The GPUs are hidden behind an ioctl (Input/Output Control) interface. For OS to access the GPUs an external driver is required for example OpenGL or CUDA.

CPU vs GPU

CPU

- Low Compute density.
- Optimized for serial operations.
- Reduces Latency

GPU

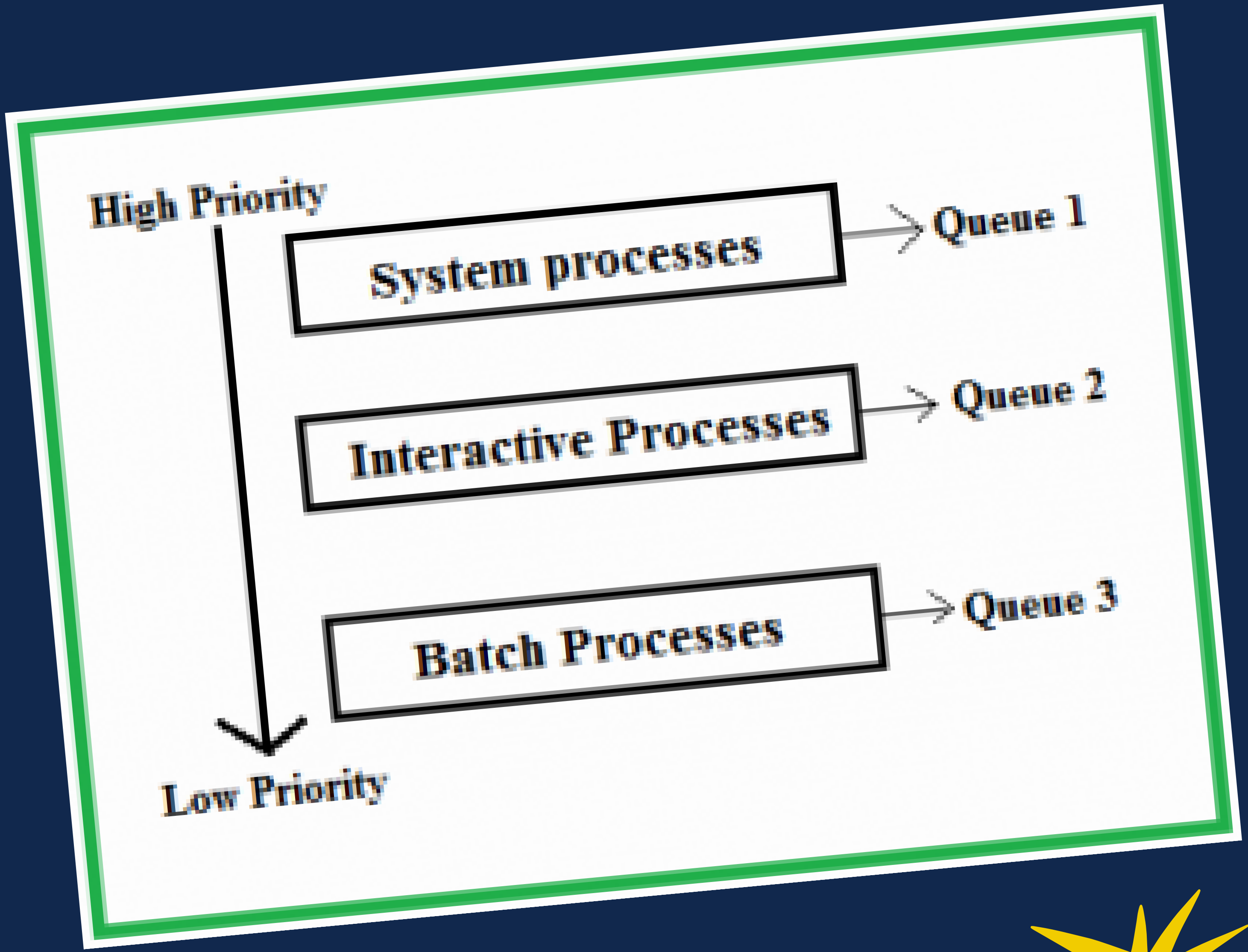
- High Compute Density.
- Built for Parallel operations.
- Maximizes the throughput.

Solution

Add a GPU abstraction for Operating Systems.

The best possible way to maximize the throughput is to provide a load balancing between CPU and GPU by giving OS the full control of the GPU.

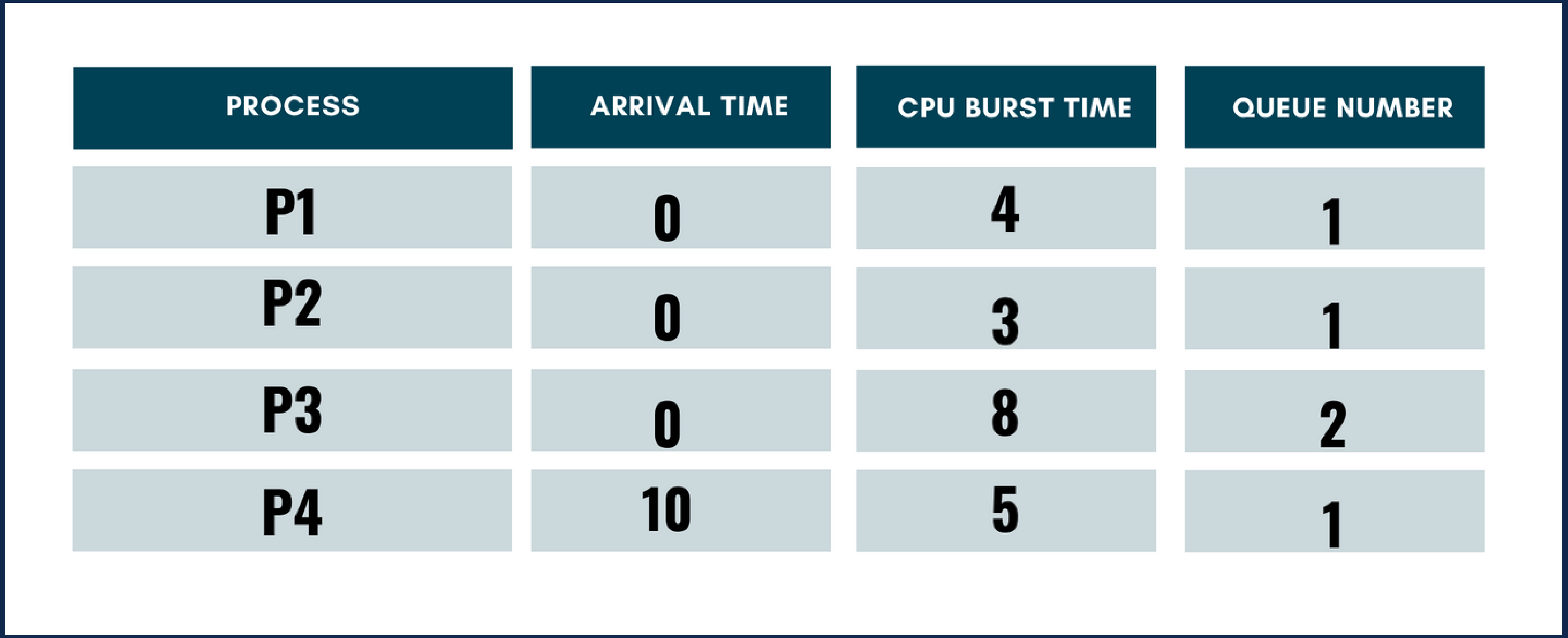
Results



Multilevel Feedback Queue Scheduling with GPU

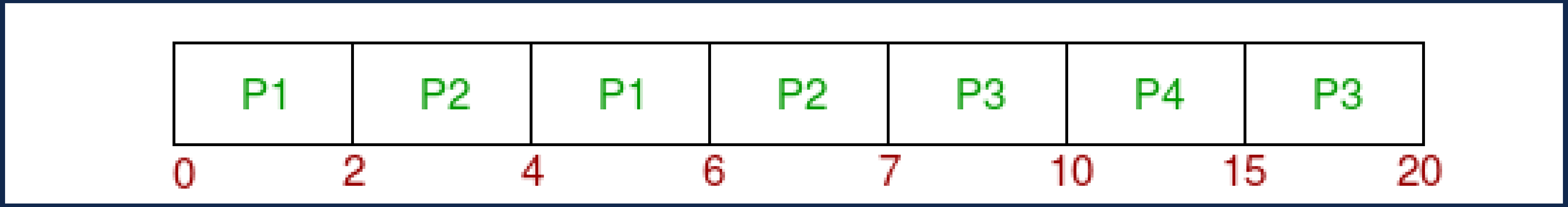
Multilevel Feedback Queue Scheduling can be implemented by introducing a parallel laod balancing for example Queue 3 can be assigned to GPU while the CPU handels Queue 1 and Queue 2.

Consider four processes with their queue number assigned such as Priority of Q1 is greater than Priority of Q2

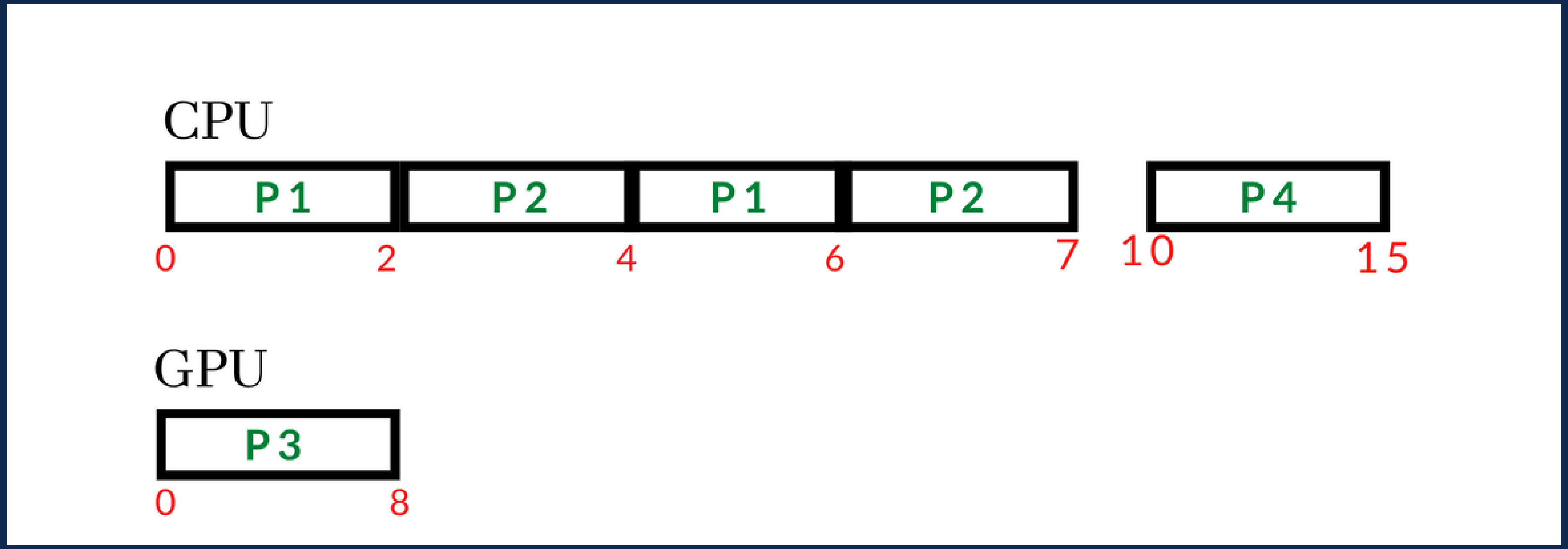


PROCESS	ARRIVAL TIME	CPU BURST TIME	QUEUE NUMBER
P1	0	4	1
P2	0	3	1
P3	0	8	2
P4	10	5	1

With CPU Its Gantt chart would be.



After implementing a load balancing between CPU and GPU



CONCLUSION

1. This research shows how GPUs can be used as a co-processing unit to maximize the system throughput.
2. GPUs should be allocated as a shared computation resource rather than an I/O device.

Thank you.

Prakhar Varshney
Boston University
CS-575 Operating systems