Real time scheduling for advanced automotive systems

Present: Yiming Gan

GPU and GPGPU

1. Same types of processor
2. GPU: only graphic operations
3. GPGPU: Your own program for anything that can be parallelized

Complete self-driving car has shown up on the road in California

Tesla
8 cameras,
Nvidia Drive PX2,
running 4 Neural
Networks

Different kinds of works that will be scheduled on GPUs

1. Tracking
2. Object Detection
3. Traffic Sign Recognition

Tracking

Frame 1

Frame 2

ref:https://www.youtube.com/watch?v=9DJDe-IwgXQ

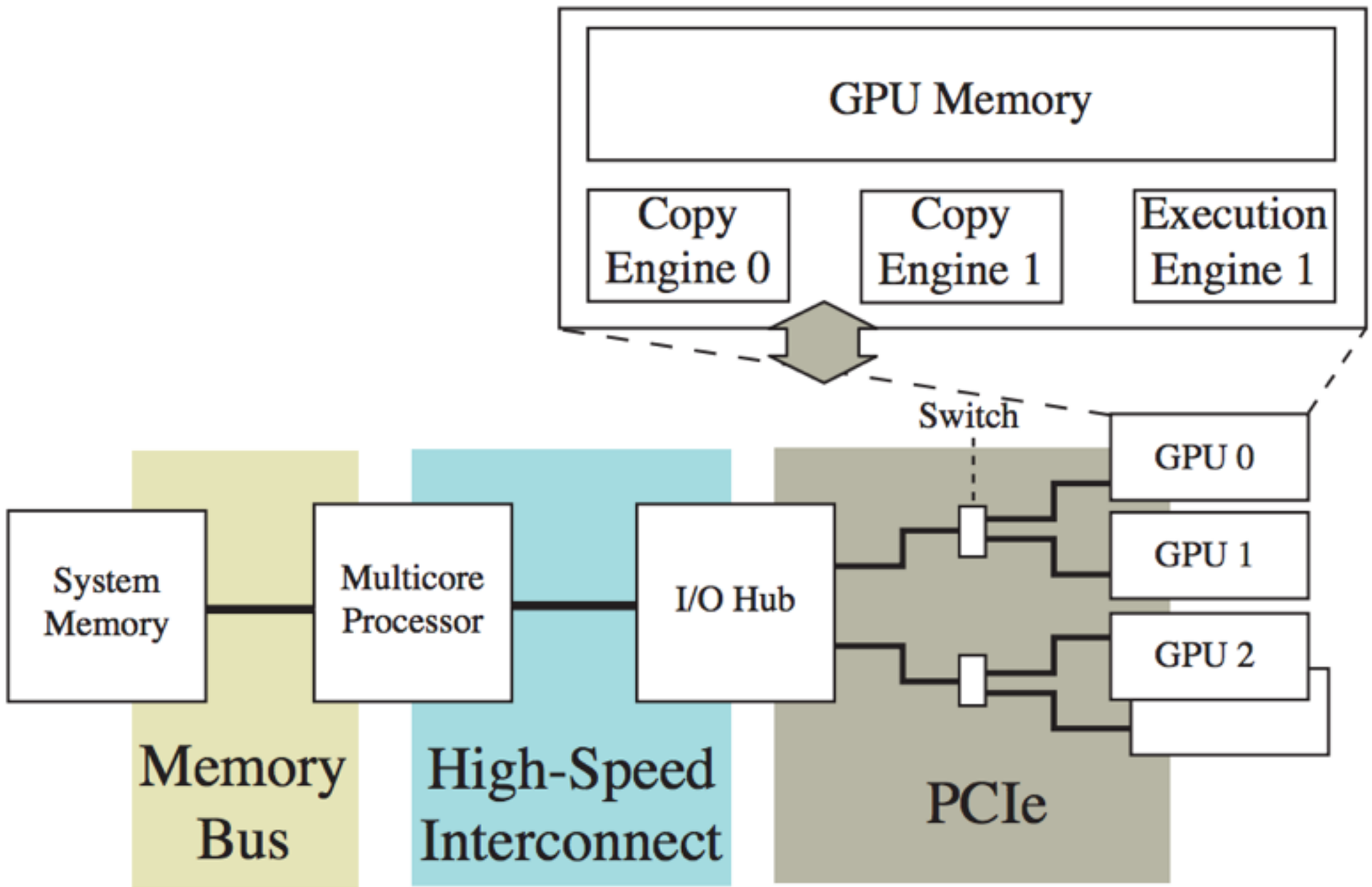
Object detection

ref:https://pjreddie.com/darknet/yolov2/

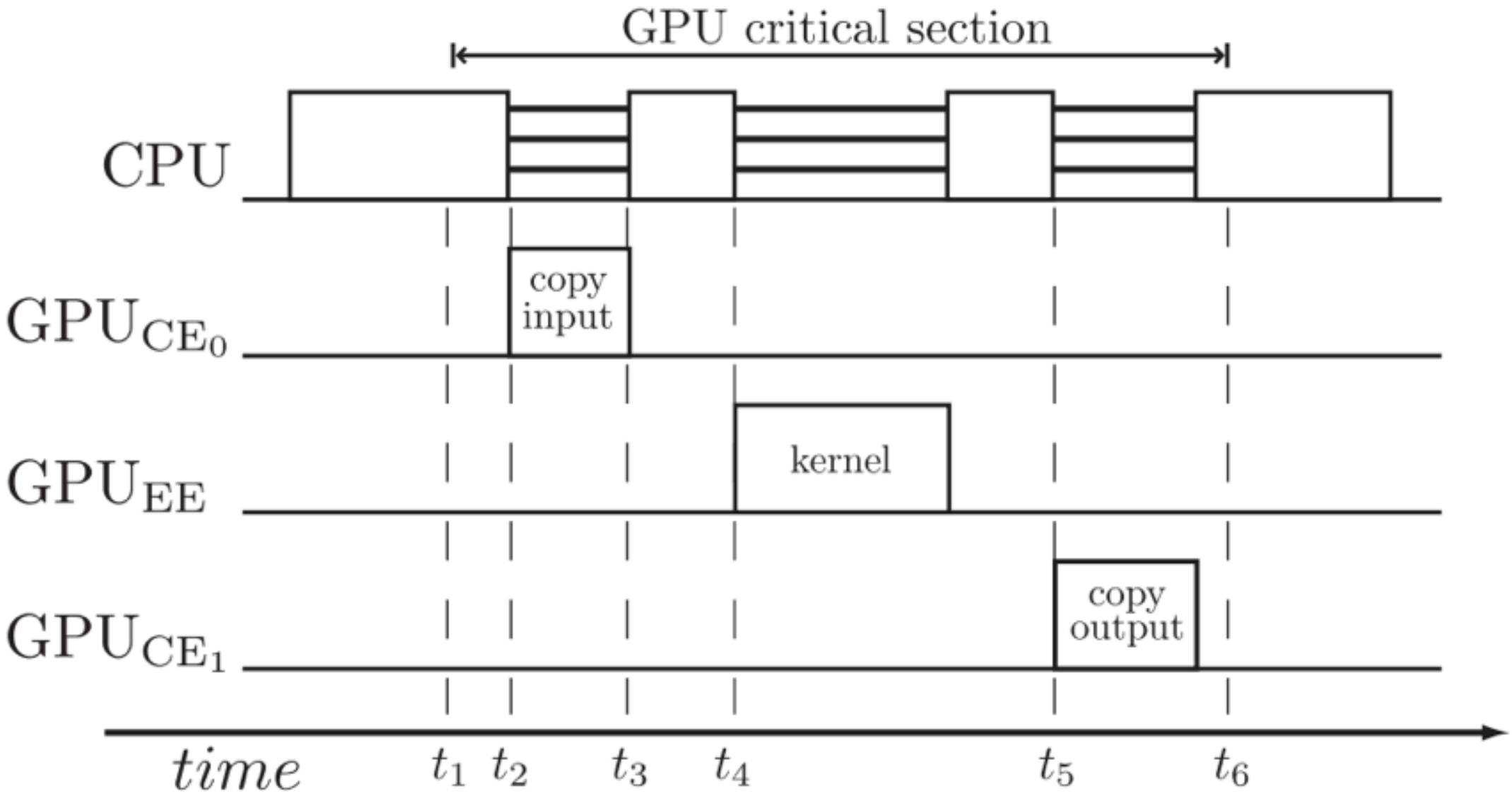
Traffic sign recognition

ref:https://www.youtube.com/watch?v=qD-axktFWt0

GPU hardware



GPGPU operations and state migration



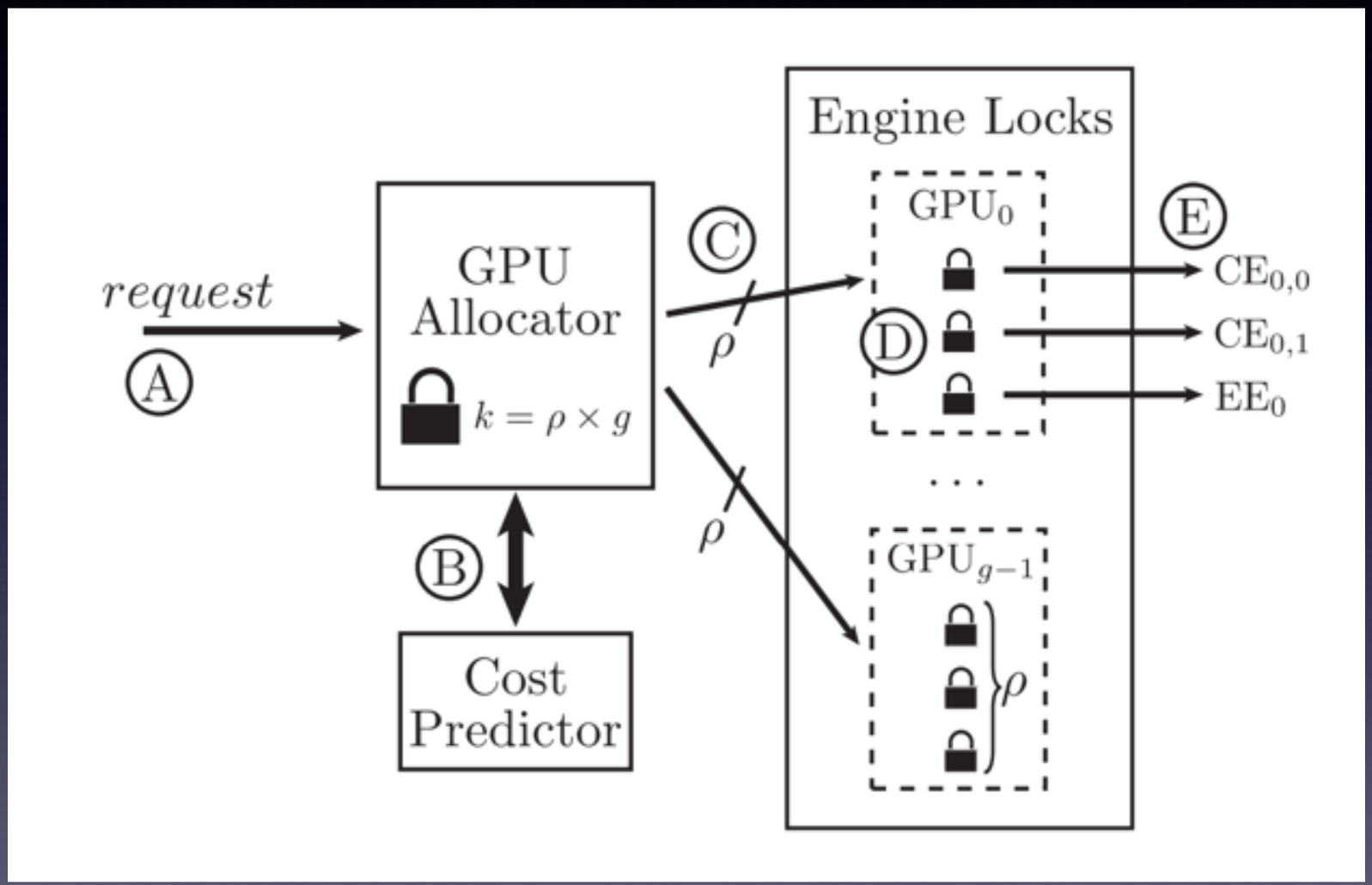
OS Concerns

1. Allocation
2. Budgeting
3. Integration

GPUSync Overview

1. A Synchronization based approach
2. Improve average-case performance while maintaining SRT
3. Take care of all three concerns

High level design for GPUSync

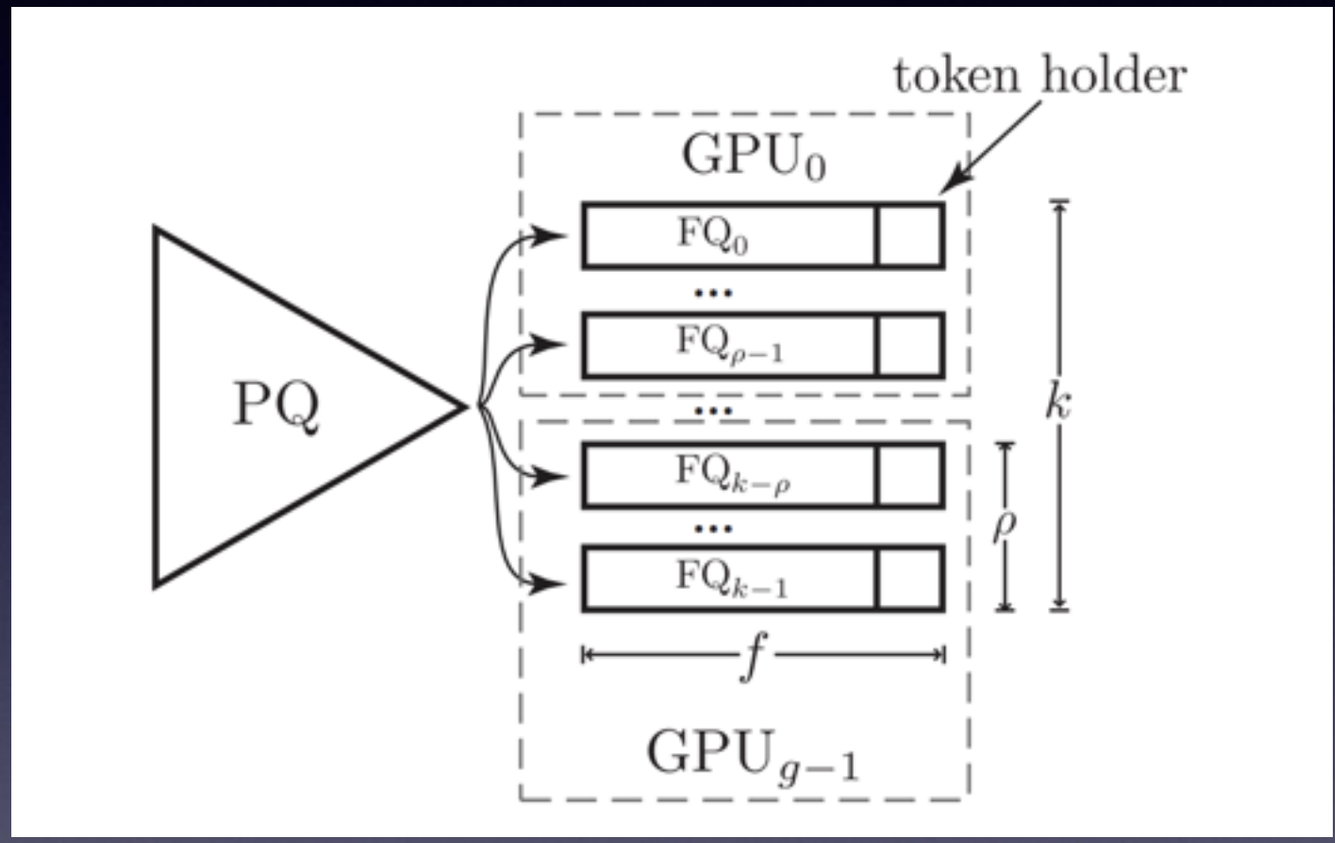


Allocators design

1. Allocators

p=3, g=number of gpus, k= 3* g, f= FIFO length executed on CPU

2. Reconfiguration f is tunable



Allocators design

1. Cost predictor
   Critical section length = execution time + potential migration cost + estimated waiting time.

   Estimated waiting time = sum of estimated execution time and migration cost for all request in FIFO.

   Average and standard deviation over a window of recent several observations.

More about allocators design

1. Engine locks
   A mutex is associated with each CE and EE
   a job is allowed to hold at most one at a time
   should be held as short as possible

More about allocators design

1. Migrations

   Supports both P2P and system memory migration
   a job must hold both copy engine locks
   deadlock avoiding

Budgeting Concern

1. Budgeting

Necessary to ensure a task's resource utilization remains within its provisioned budget.

Budgeting Concern

1. Signaled overruns

Pseudocode 1 Example of budget signal handling.

```
▷ Enabled/disabled on try-block enter/exit.
function BUDGETSIGNALHANDLER()
    throw BudgetException();
end function

procedure DOJOB()
    t ← GetToken();
    gpu ← MapTokenToGpu(t);
    try:
        DoGpuWork(gpu);              ▷ Main job computation.
    catch BudgetException:
        CleanUpGpu(gpu);             ▷ Gracefully cleans up state.
    finally:
        FreeToken(t);
end procedure
```

Budgeting Concern

1. Early releasing

   Continue executing overrunning job when budget exhausts, sacrifices the following job's budget.

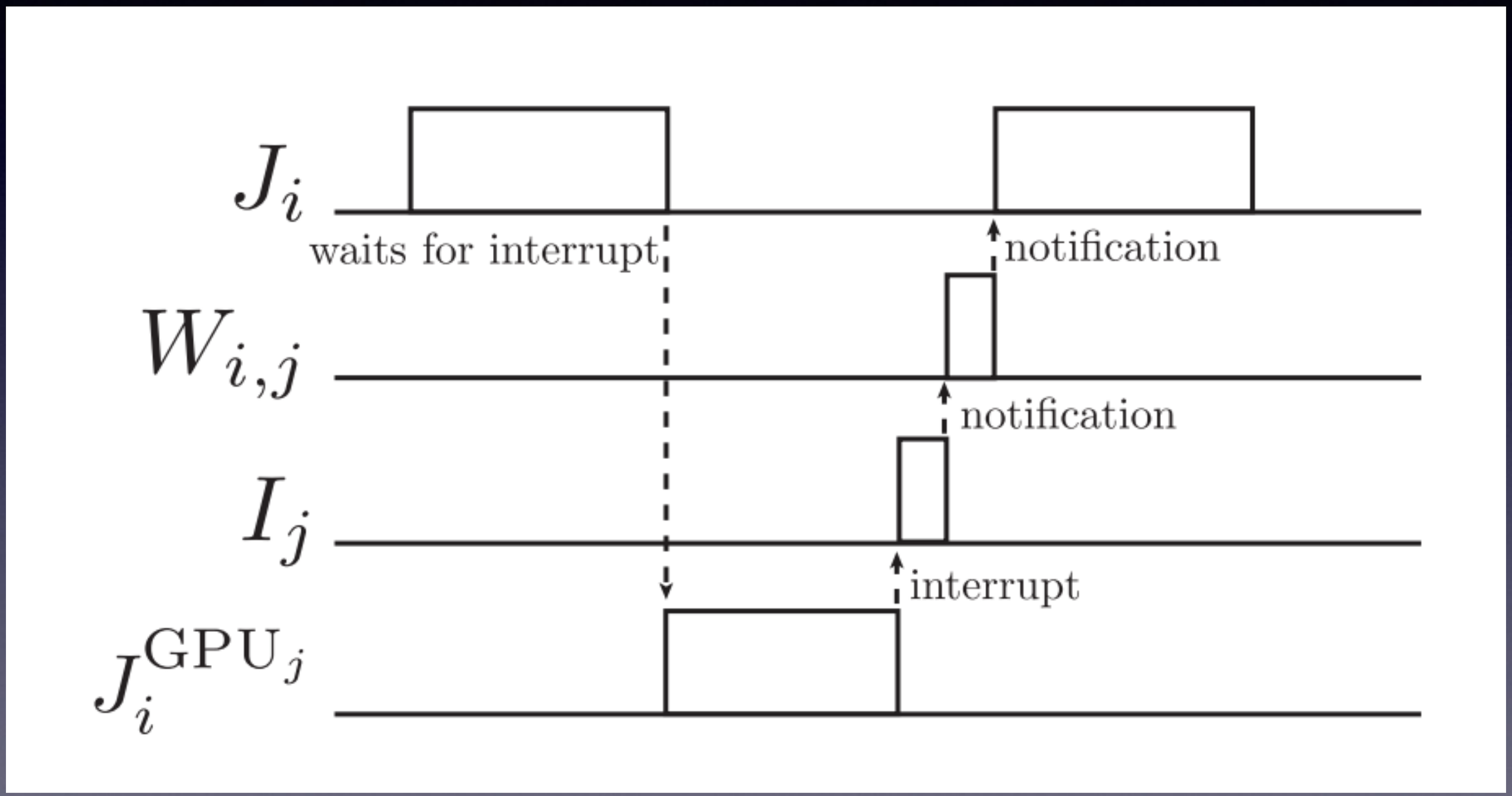
Solving Integration

1. Two types of threads:

   OS kernel-level threads, one per GPU, processes GPU interrupts.

   GPGPU runtime user-level threads, g per task, mediates communication between application code and the GPU driver

Solving Integration



Observations via evaluation

1. A synchronization-based approach to GPU scheduling is effective at supplying tasks with provisioned execution time.
2. Budget enforcement can penalize overrunning tasks.
3. Cost predictor accuracy varies among different situations.