HERIOT WATT UNIVERSITY

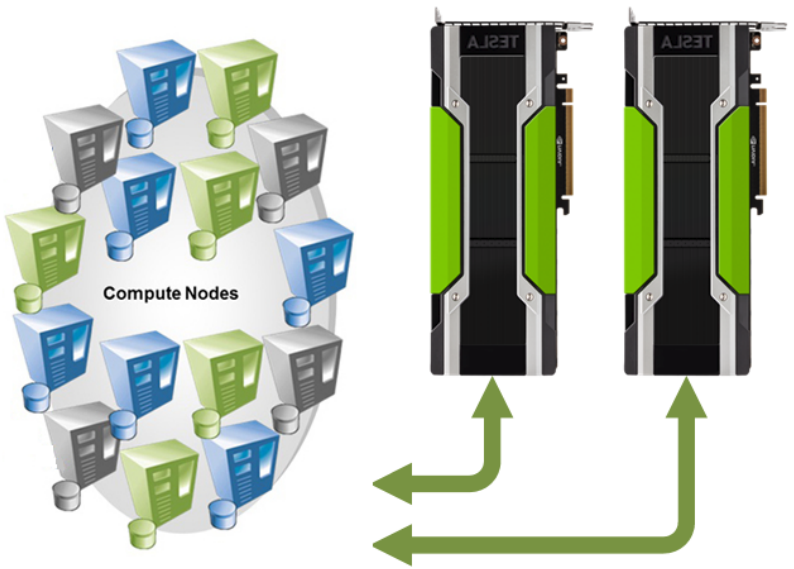
A soft implementation for interrupting GPU kernels

Max M. Baird, Christian Fensch, Sven-Bodo Scholz and Artjoms Šinkarovs

LEADERS IN IDEAS AND SOLUTIONS

HERIOT WATT UNIVERSITY

Checkpoint/Restart with GPUs



- GPUs run asynchronously
- Synchronize with the GPU
- Wait until GPU finishes
- **PROBLEM: Cannot interrupt GPU**

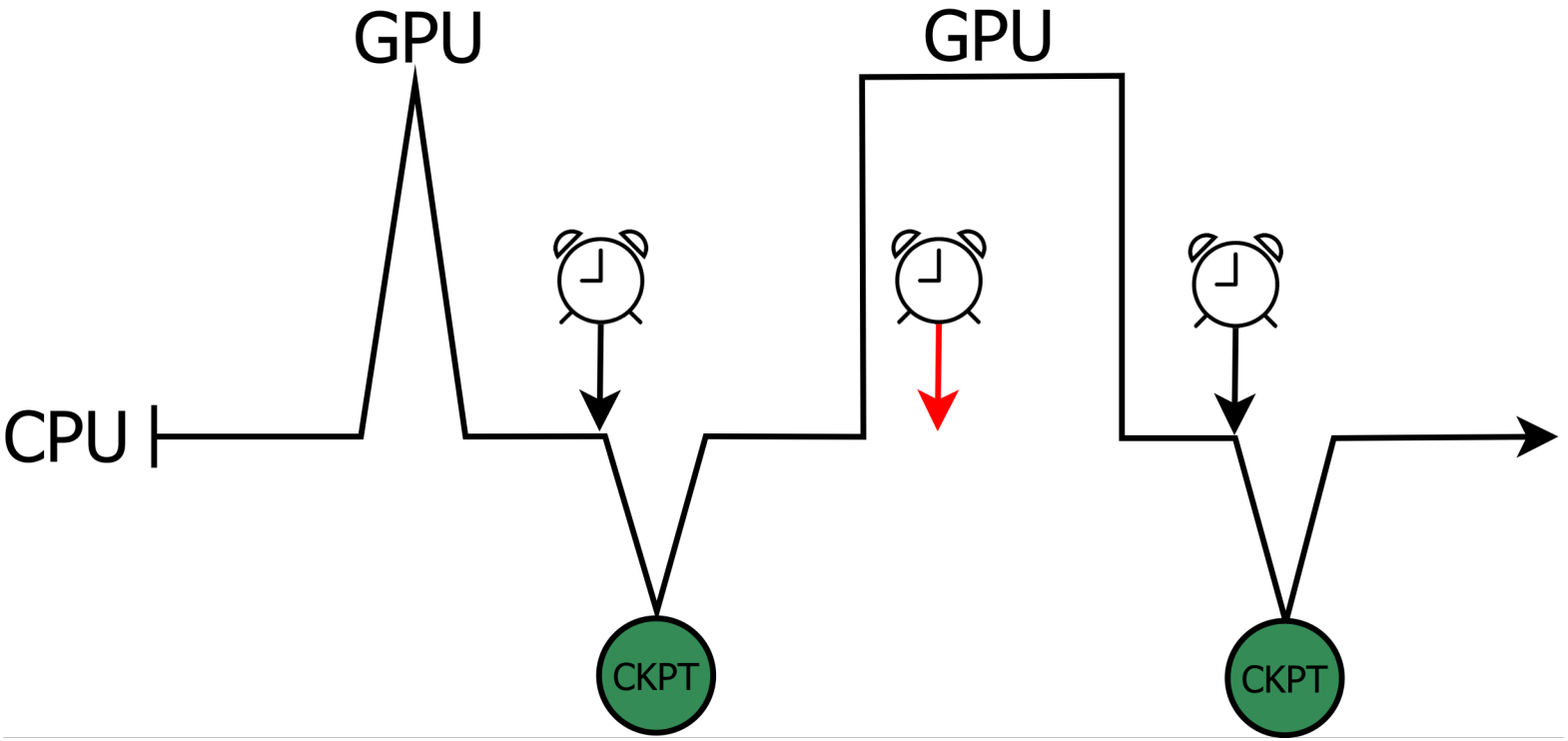
HERIOT WATT UNIVERSITY

Checkpoint/Restart Libraries

- 2002. BLCR – No Support for GPUs
- 2009. CheCUDA – Compatible with BLCR
- 2011. NVCR – Similar to CheCUDA
- 2011. FTI – No support for GPUs

HERIOT WATT UNIVERSITY

Checkpoint/Restart with GPU



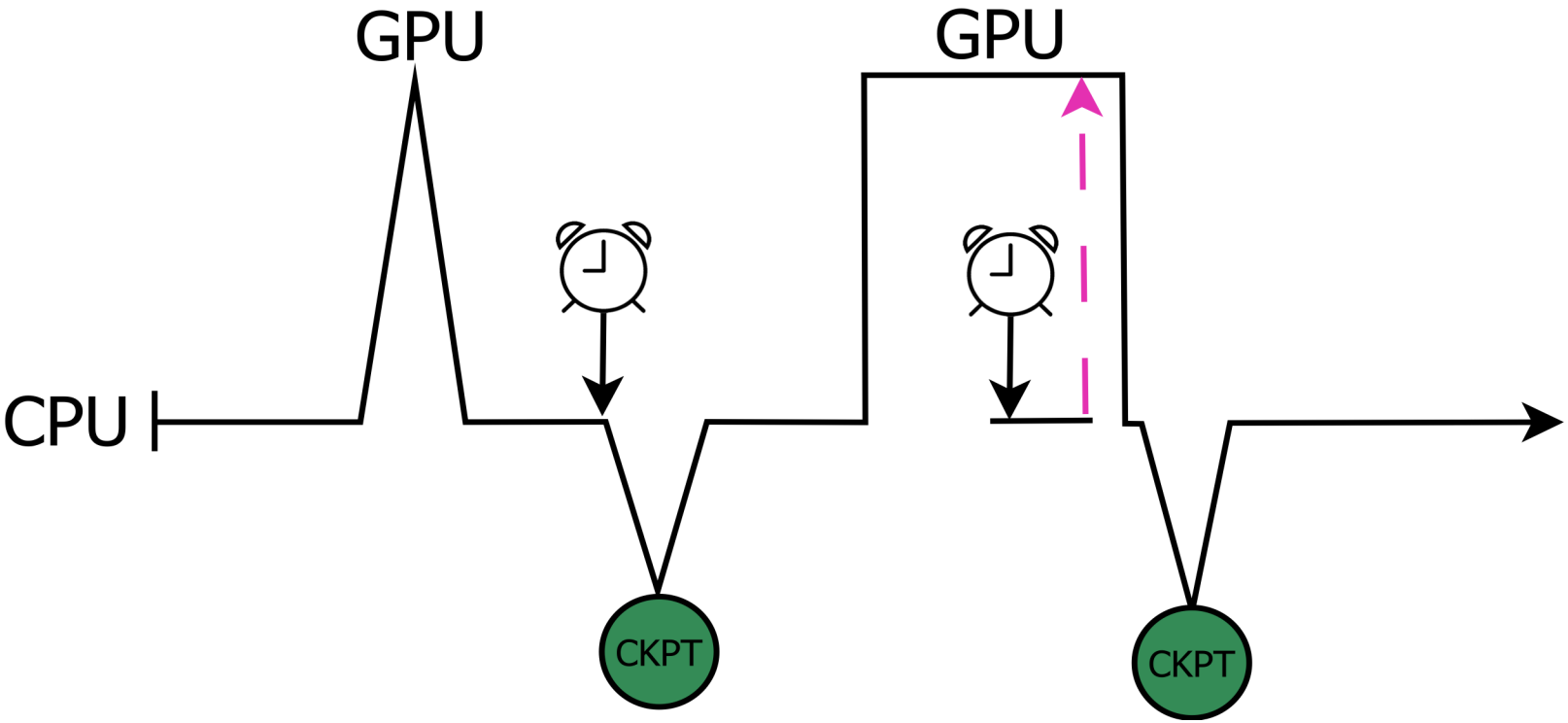
HERIOT WATT UNIVERSITY

Possible Approaches?

- Manually Rewrite kernels
  - Can be tricky
- Automated approach?
  - No interrupt
  - What memory to transfer?

HERIOT WATT UNIVERSITY

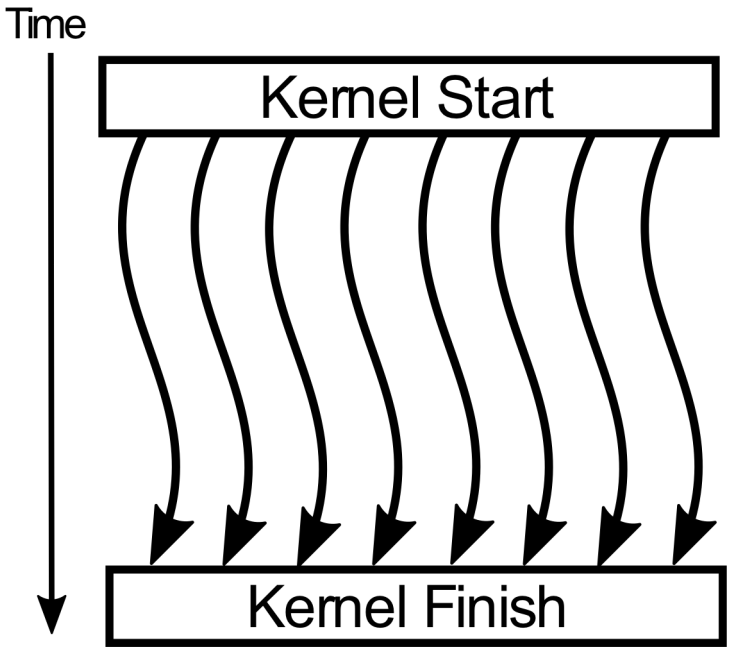
Solution: Soft Interrupt



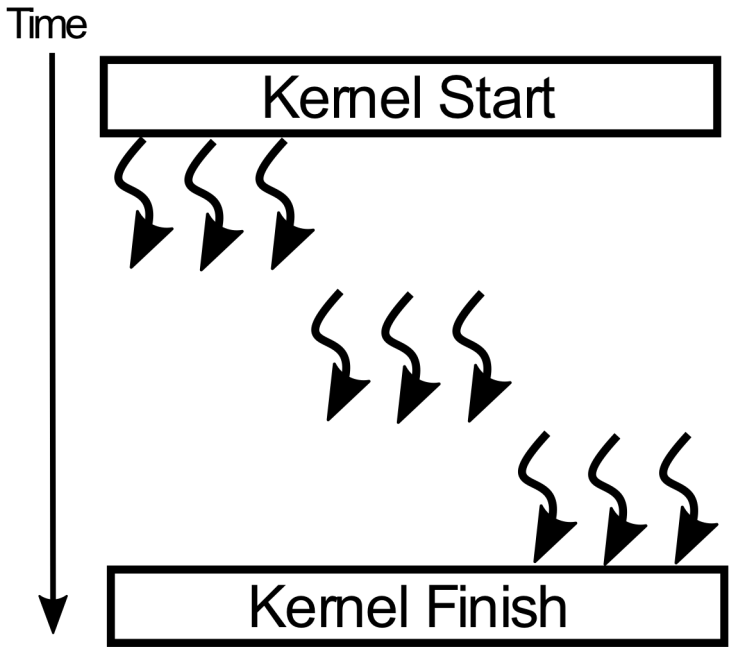
HERIOT WATT UNIVERSITY

Key Idea

Conceptually



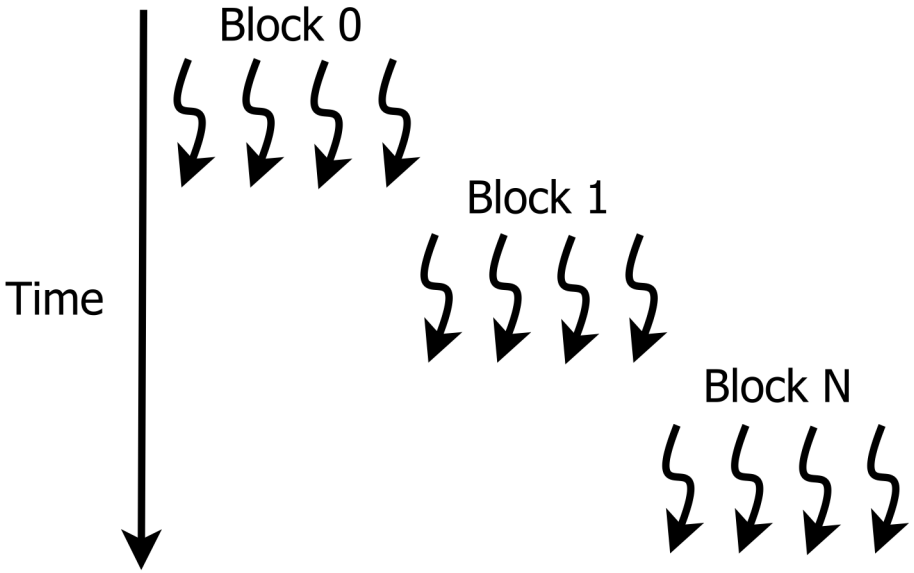
Reality



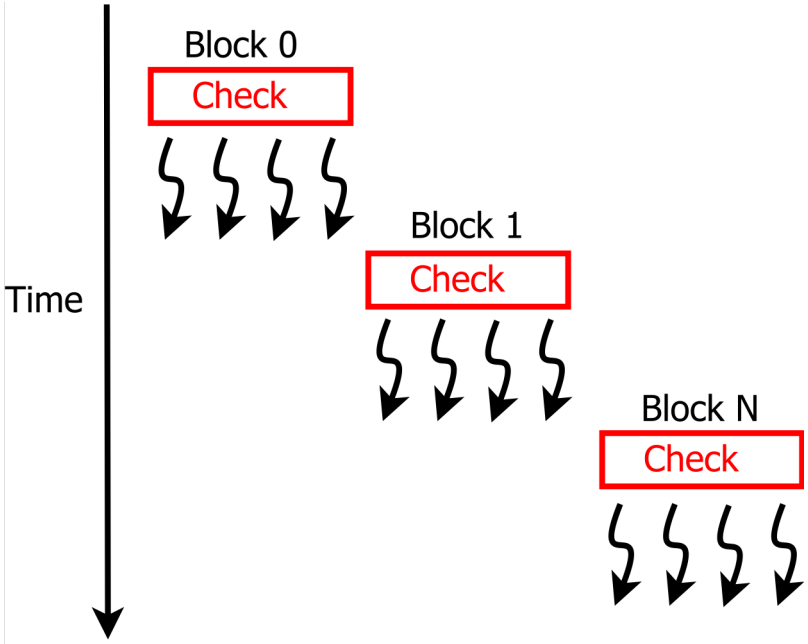
HERIOT WATT UNIVERSITY

Key Idea

CUDA model



Checks can be inserted



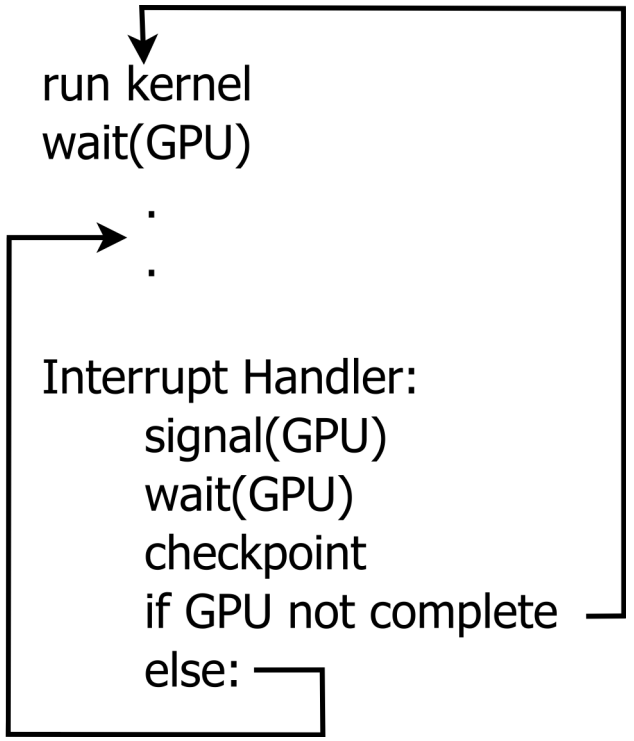
HERIOT WATT UNIVERSITY

Idea of Soft Interrupt

- GPU

```
do{
    If  time to return
    execute
}while(work)
```

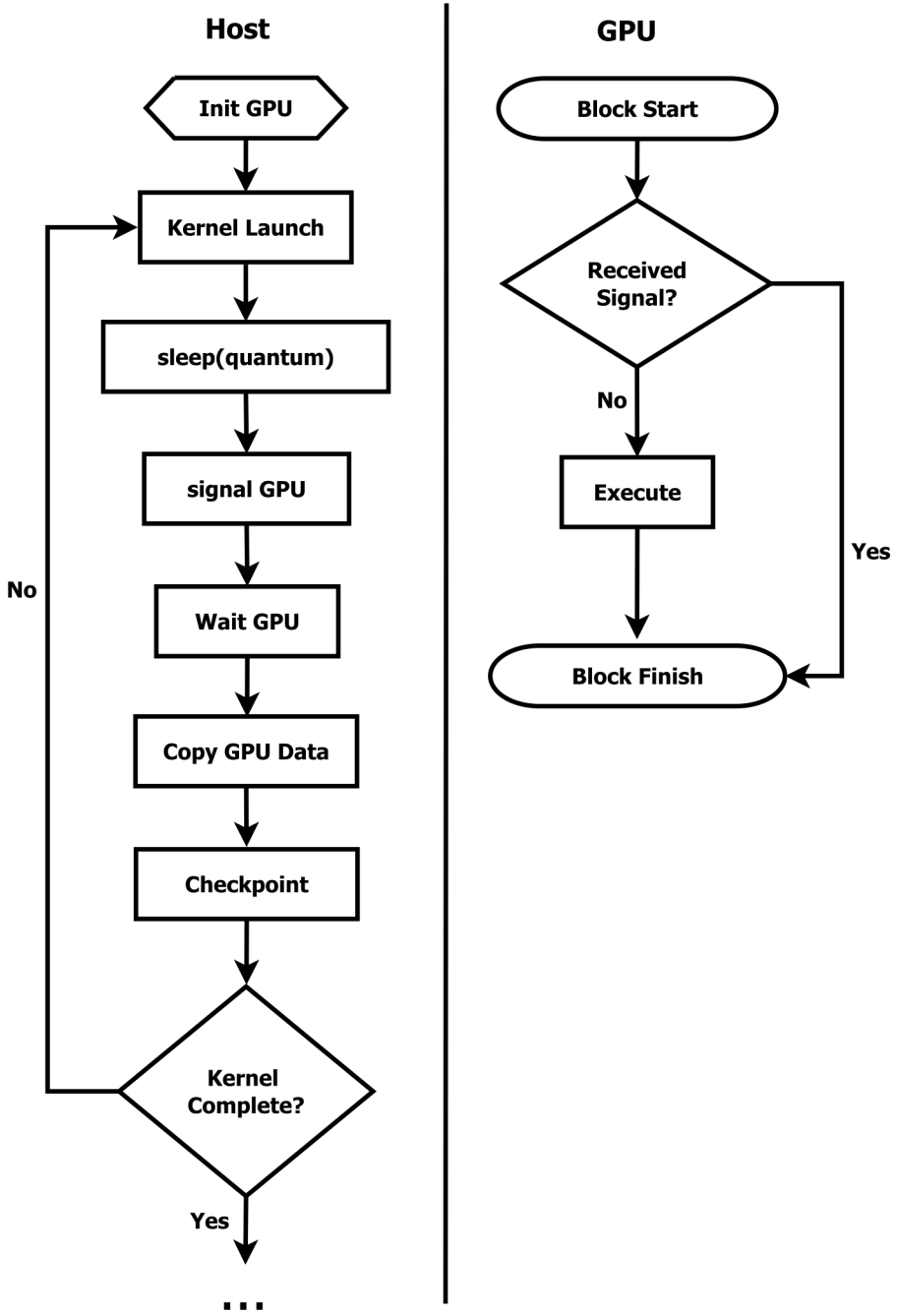
- Host



HERIOT WATT UNIVERSITY

Checkpoint/Restart Extension

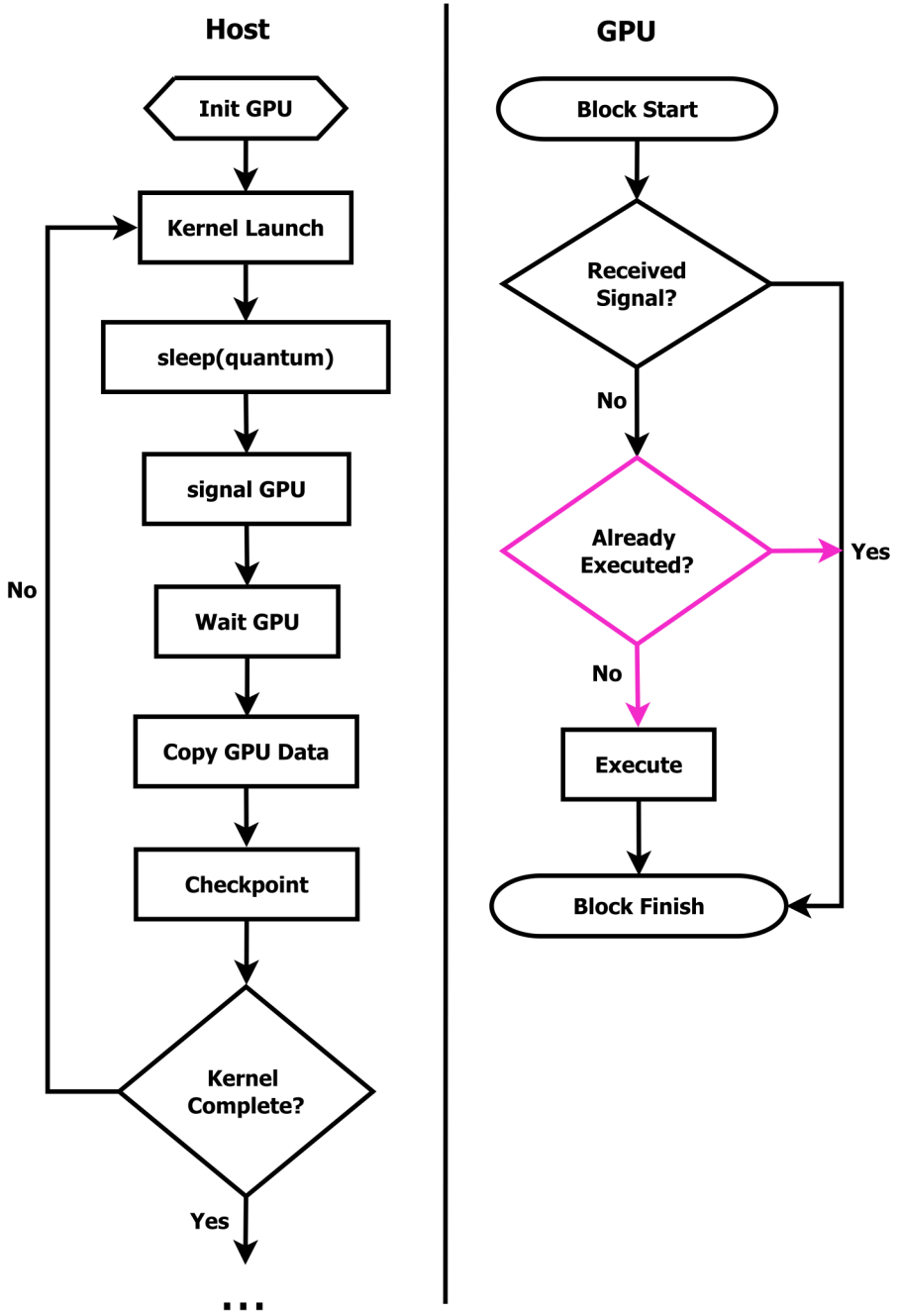
- Kernel executes in loop
- Thread groups check for permission to continue



HERIOT WATT UNIVERSITY

Checkpoint/Restart Extension

- Kernel executes in loop
- Thread groups check for permission to continue



HERIOT WATT UNIVERSITY

A library to automate this

- Concise API
  - 3 Macros
    - Re-write kernel definition
    - Re-write kernel launch
    - Insert check in kernel
  - 2 Wrappers
    - Keep track of allocations on GPU
    - Keep track of frees on GPU

https://bitbucket.org/maxbaird/cuda_backup/

HERIOT WATT UNIVERSITY

Sources of Overhead

- Conditional checks in each thread
- Soft interrupts of a kernel
- Memory transfers

HERIOT WATT UNIVERSITY

Kernel for Evaluating Overhead

```
__global__ void
kernel(unsigned long long n, unsigned long long
*res)
{
 unsigned long long x=0;
 for (unsigned long long i=0 ; i<n; i++){
      x++;
 }
 *res = x ;
}
```

HERIOT WATT UNIVERSITY

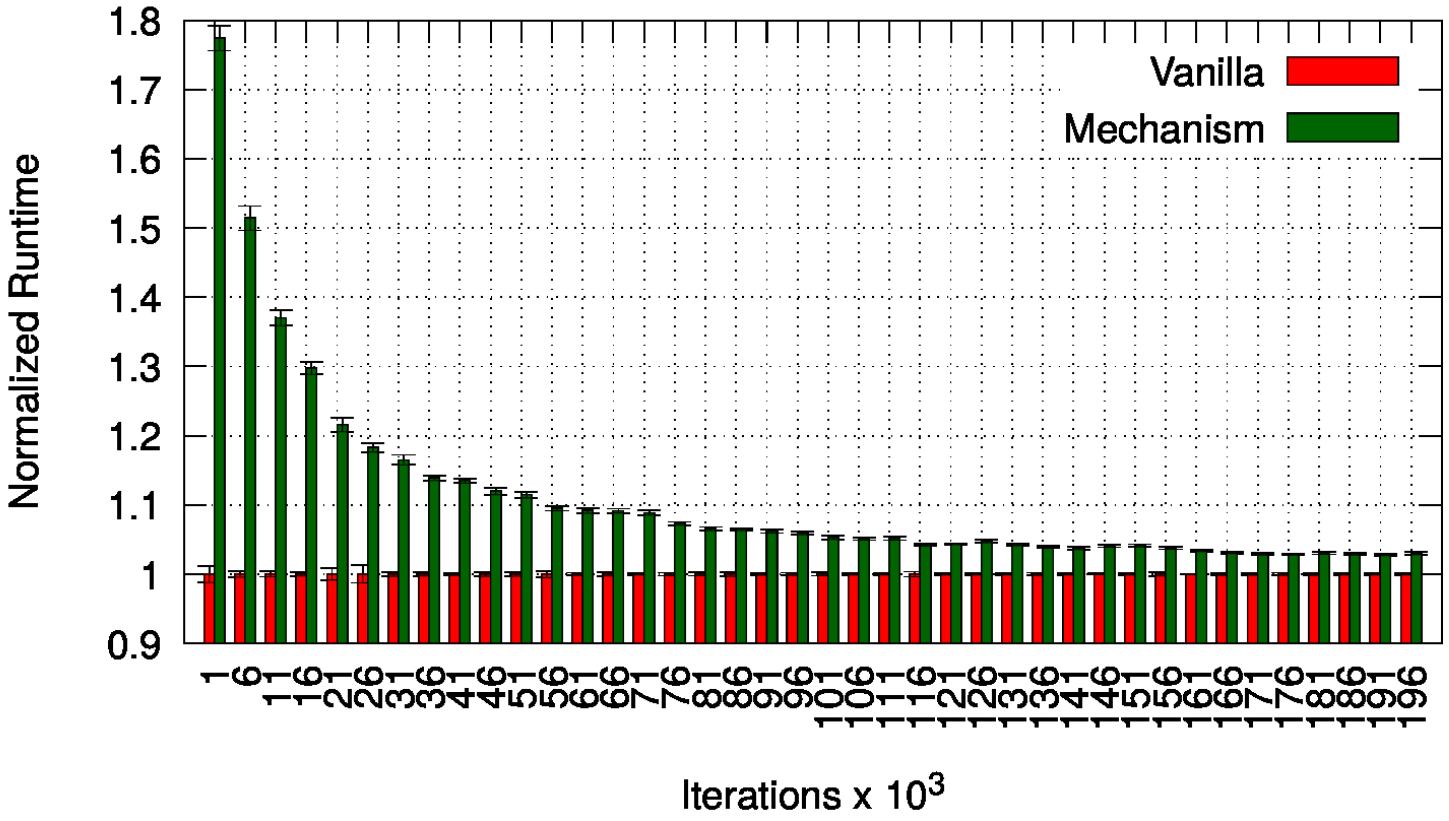
Experiment Environment

CPU	AMD Opteron 6376
GPU	Nvidia Titan-XP
CPU Memory	512 GB
GPU Memory	12 GB
Driver Version	384.81
CUDA Version	9.0
PCIe	x16
Operating System	Scientific Linux Release 7.4

HERIOT WATT UNIVERSITY

Overhead of Conditional Check

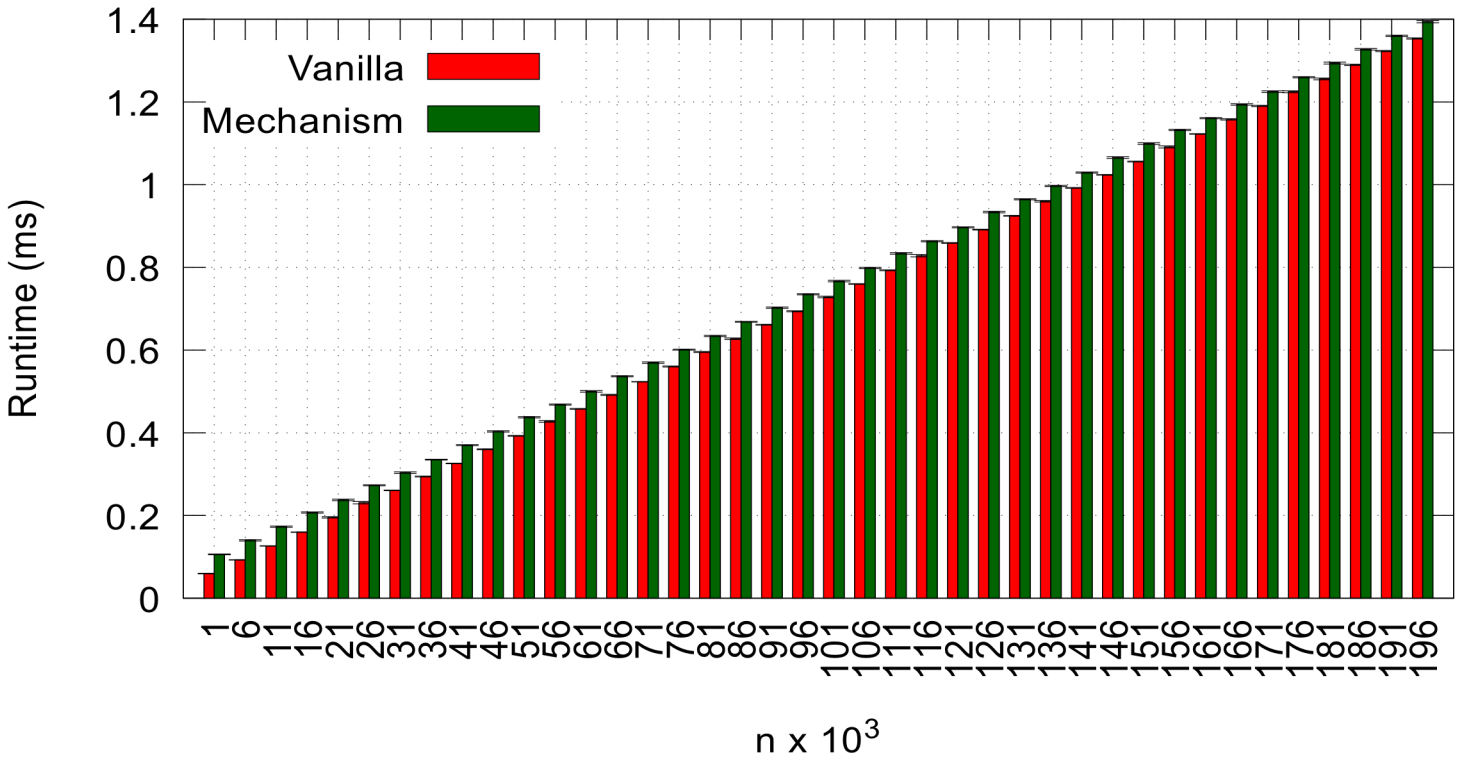
kernel config <<<60, 1024>>>



HERIOT WATT UNIVERSITY

Overhead of Conditional Check

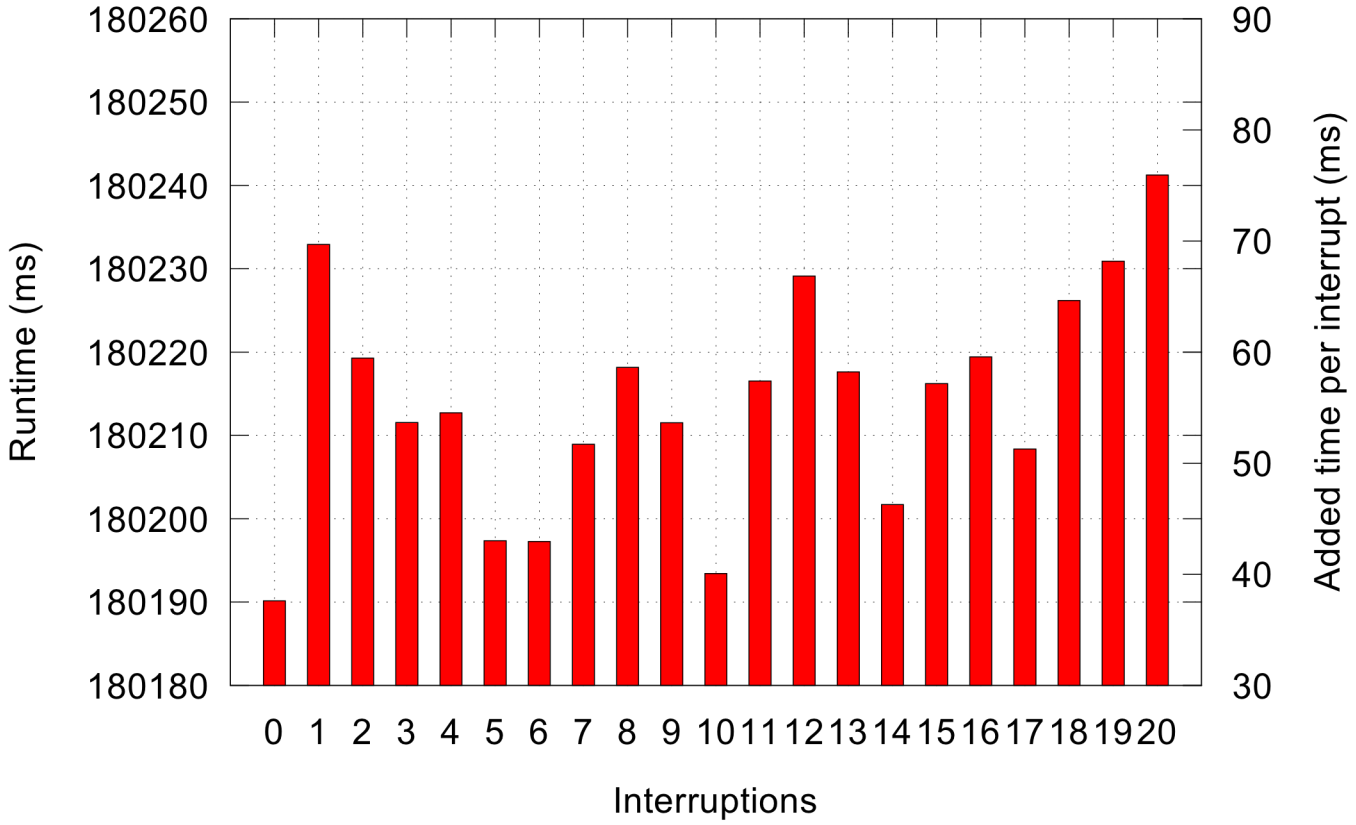
kernel config <<<60, 1024>>>



HERIOT WATT UNIVERSITY

Overhead of Soft Interrupt

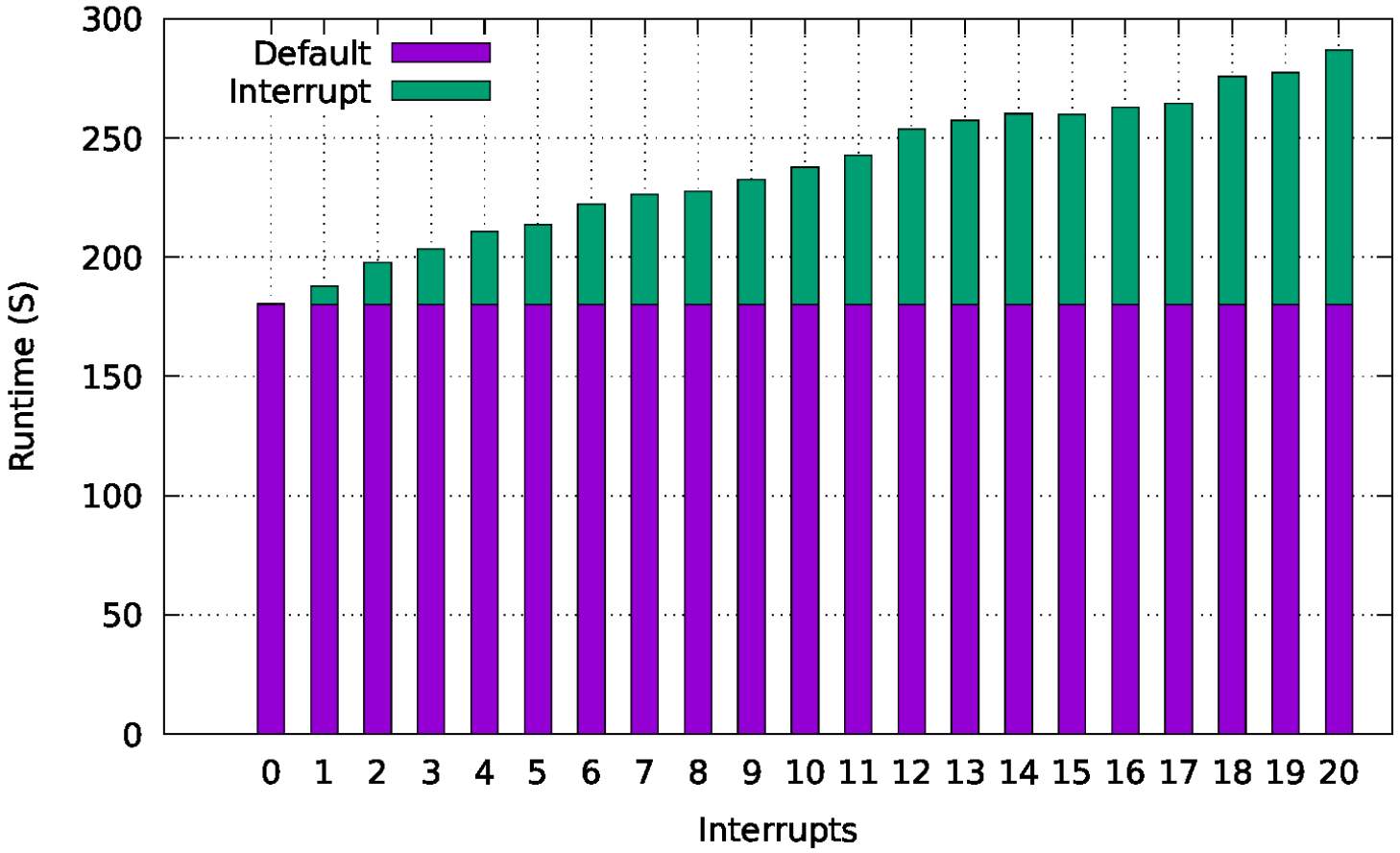
kernel config <<<1320, 1024>>> iterations = 1.6 x 10^9



HERIOT WATT UNIVERSITY

Memory Transfers

98.3% (11.9GB) allocated GPU memory, kernel config <<<1320, 1024>>>
iterations = 1.6×10^9



HERIOT WATT UNIVERSITY

Achievements/Conclusions

- Negligible overhead for the kernel
- Very simple to mechanically transform kernels
- Potential limitations
  - Global synchronization
    - Should be broken into sepaparte kernels
  - Small kernel launch configurations
    - Don’t occur in practice with long running kernels

HERIOT WATT UNIVERSITY

Future Work

- Integrating with the fault tolerance library FTI from BSC
- Leverage the existing MPI and cluster support of FTI
- Compiler integration

HERIOT WATT UNIVERSITY

Questions?