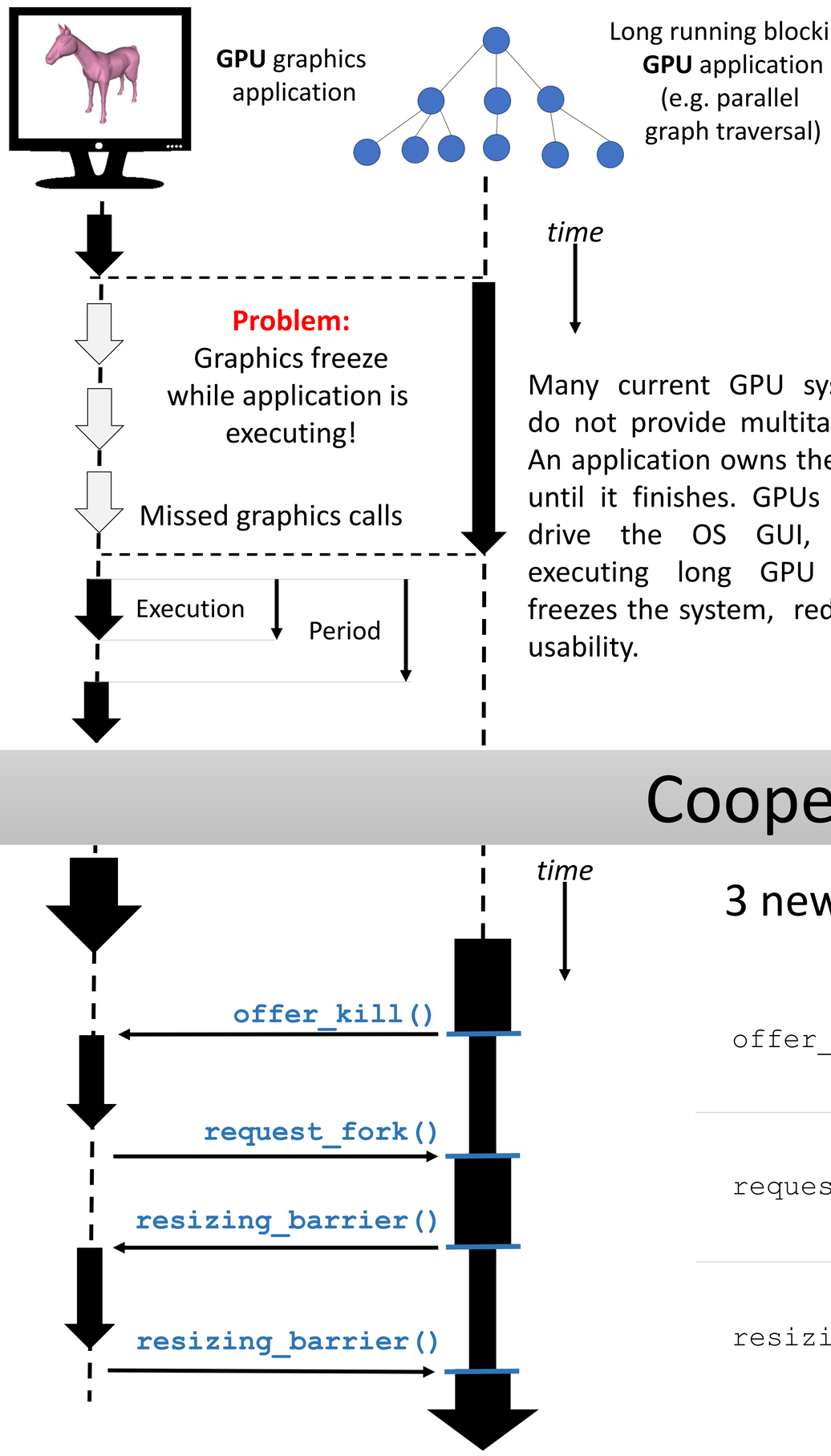
Cooperative Kernels: GPU Multitasking for Blocking Algorithms

Tyler Sorensen, Hugues Evrard, and Alastair F. Donaldson
Imperial College London

Distinguished paper award

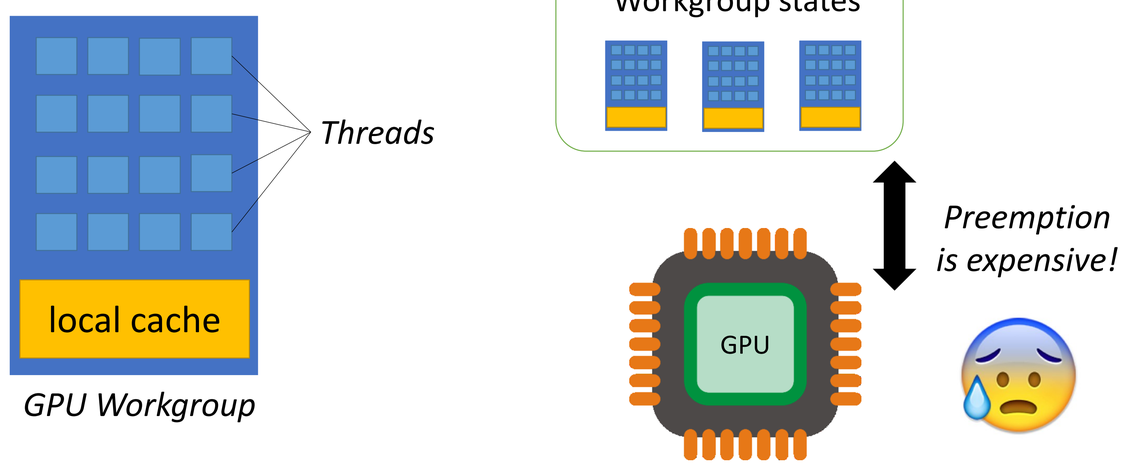
Motivation



Many current GPU systems do not provide multitasking. An application owns the GPU until it finishes. GPUs often drive the OS GUI, thus, executing long GPU apps freezes the system, reducing usability.

GPU preemption

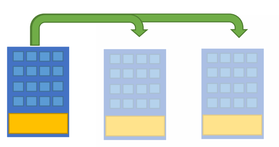
For CPU multicore systems, preemption solves the multitasking problem. Preemption is the ability to save the state of a programs thread and remove it from a hardware resource to return later.



On GPUs, preemption is difficult due to the large state that needs to be saved. A GPU workgroup's state contains up to **256 threads** and **a local cache**. Efficiently saving and restoring is non-trivial.

Cooperative kernels

3 new programming instructions for stateless multitasking

`offer_kill()`	A calling workgroup is ready to be killed ***if*** the system needs the resource for another task	
`request_fork()`	A calling workgroup may be forked (copied), ***if*** the system has available resources.	
`resizing_barrier()`	Synchronizes all workgroups in the program. At this point workgroups may be killed or forked ***depending*** on resource availability or contention.	*Either!* *+ sync Barrier*

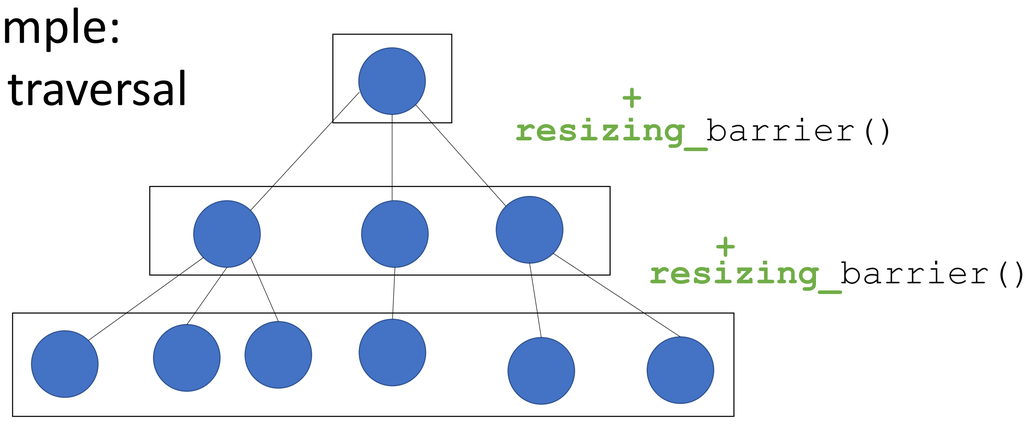
Programming model

Currently programmers are responsible for understanding and adding cooperative instructions correctly.

Cooperative kernels are **backwards compatible**. New instructions can be treated as no-ops on existing GPUs.

Ported **8** existing applications with **minimal** changes following simple guidelines.

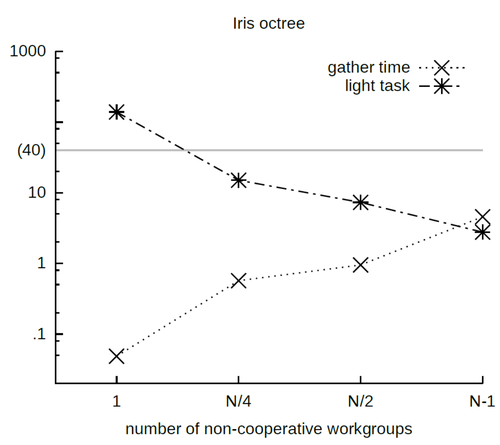
Example:
graph traversal

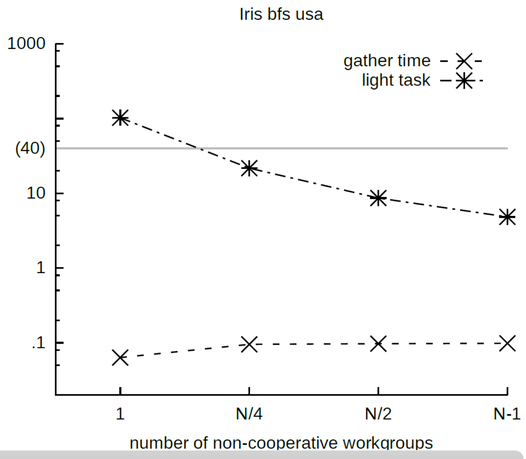


Results

Prototype framework implemented for Intel GPUs. Models two tasks (graphics and long-running). Graphics tasks of three levels of intensity tested. We maintain smooth GUI on all graphics tasks with reasonable overhead on the long-running application.

Workload	Period	Execution	Overhead	Workgroups
Light	70 ms	3 ms	1.00x	25%
Medium	40 ms	3 ms	1.03x	25%
Heavy	40 ms	10 ms	1.28x	50%





Support: EPSRC Fellowship EP/N026314, and a gift from Intel Corporation