The introduction of GPGPU and some implementations on model checking

Zhimin Wu
Nanyang Technological University,
Singapore

Outline

- What is GPGPU
- The Architecture and Features of latest NVIDIA Kepler GK110
- The reference to current researches on the implementation of GPGPU in model checking
- Brief introduction of my current work

General-purpose computing on graphics processing units

- the utilization of aÄgraphic processing units (GPU), which typically handles computation only forÄcomputer graphic, to perform computation in applications traditionally handled by theÄcentral processing unit (CPU)

- CPU: ILP and TLP GPU: DLP

- High Parallelism, High Memory Bandwidth, High computational Power

- Compute Framework: CUDA

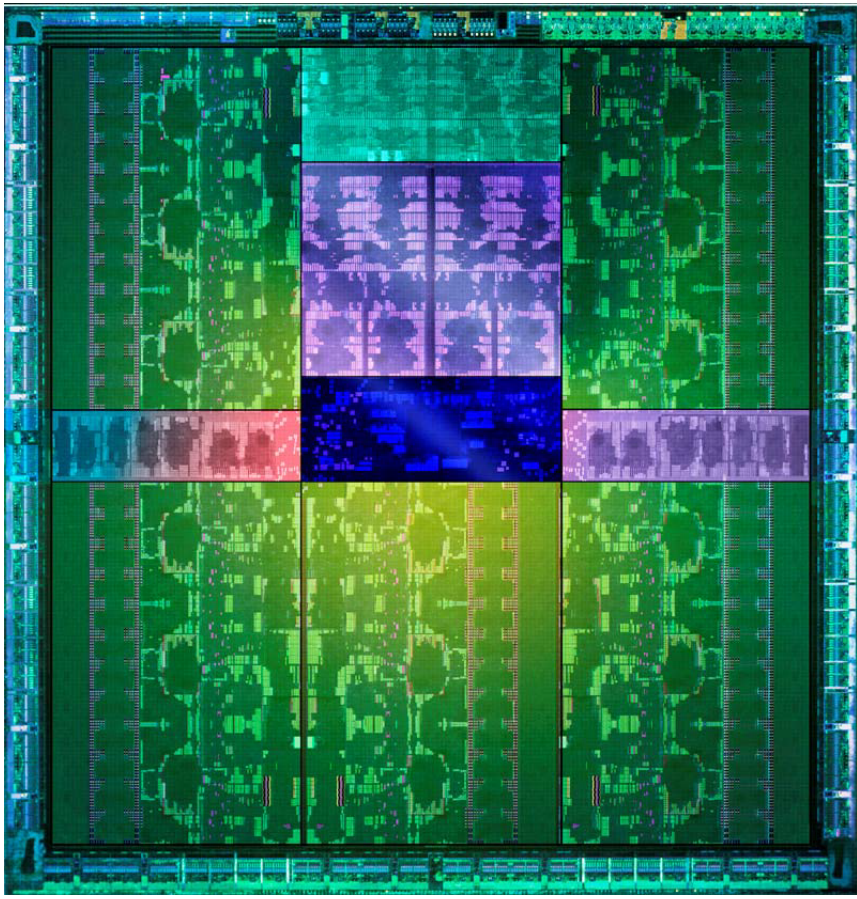
Kepler GK110 Chip

Designed for performance and power efficiency

7.1 billion transistors

Over 1 TFlop of double precision throughput
3x performance per watt of Fermi

New features in Kepler GK110:
- **Dynamic Parallelism**
- **Hyper-Q** with GK110 Grid Management Unit (GMU)
- **NVIDIA GPUDirect® RDMA (United State Space)**



Kelper GK110 Full chip block diagram

Kepler GK110 supports the new CUDA Compute Capability 3.5

	FERMI GF100	FERMI GF104	KEPLER GK104	KEPLER GK110
Compute Capability	2.0	2.1	3.0	3.5
Threads / Warp	32	32	32	32
Max Warps / Multiprocessor	48	48	64	64
Max Threads / Multiprocessor	1536	1536	2048	2048
Max Thread Blocks / Multiprocessor	8	8	16	16
32-bit Registers / Multiprocessor	32768	32768	65536	65536
Max Registers / Thread	63	63	63	255
Max Threads / Thread Block	1024	1024	1024	1024
Shared Memory Size Configurations (bytes)	16K 48K	16K 48K	16K 32K 48K	16K 32K 48K
Max X Grid Dimension	2^16-1	2^16-1	2^32-1	2^32-1
Hyper-Q	No	No	No	Yes
Dynamic Parallelism	No	No	No	Yes

Compute Capability of Fermi and Kepler GPUs

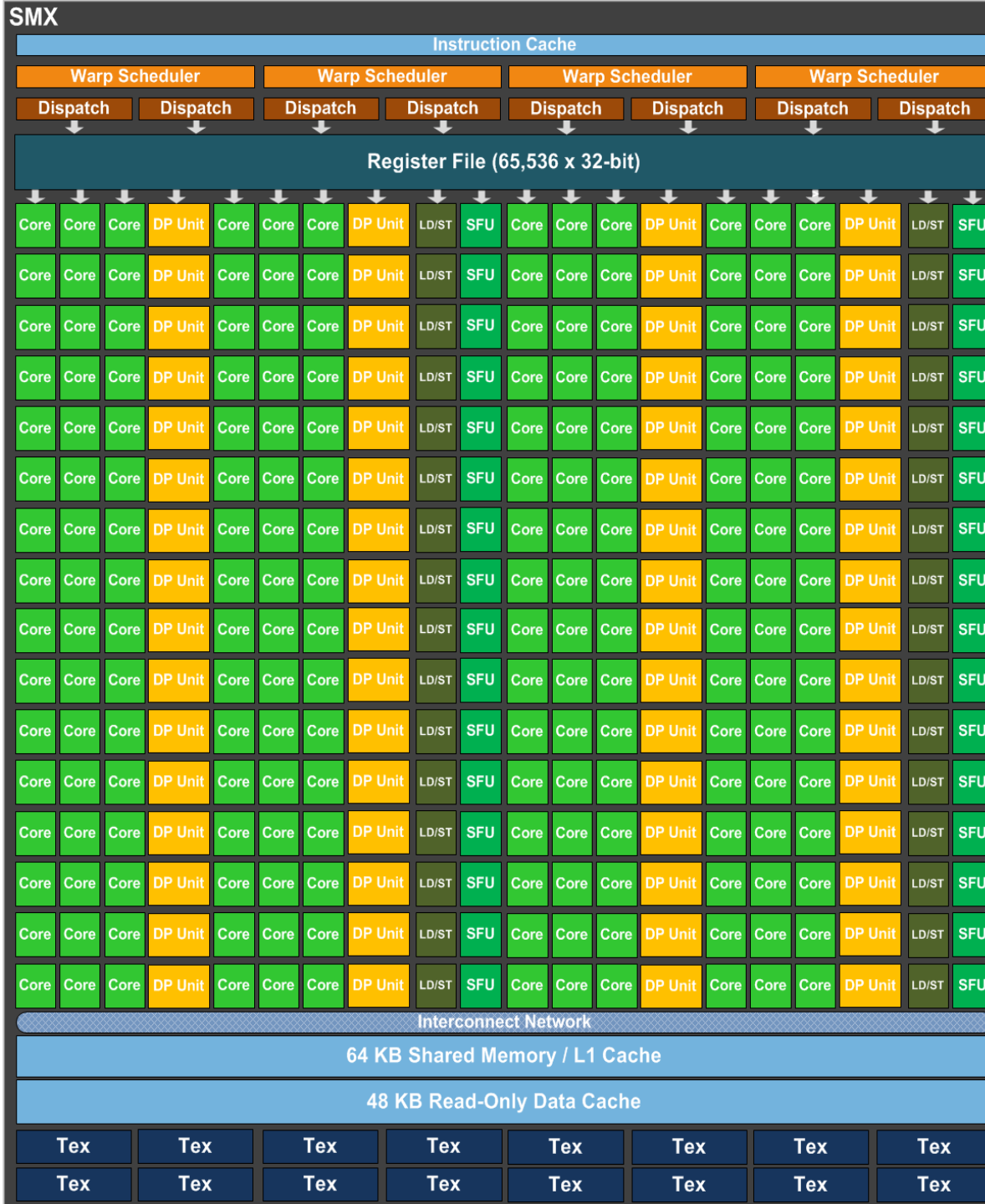
GTX 470/480s have GT100s
C2050s on grid06 and grid07 are compute cap 2.0

An SMX

192 single-precision CUDA cores, 64 double-precision units, 32 special function units (SFU), and 32 load/store units (LD/ST).

Full Kepler GK110 has 15 SMXs

Some products may have 13 or



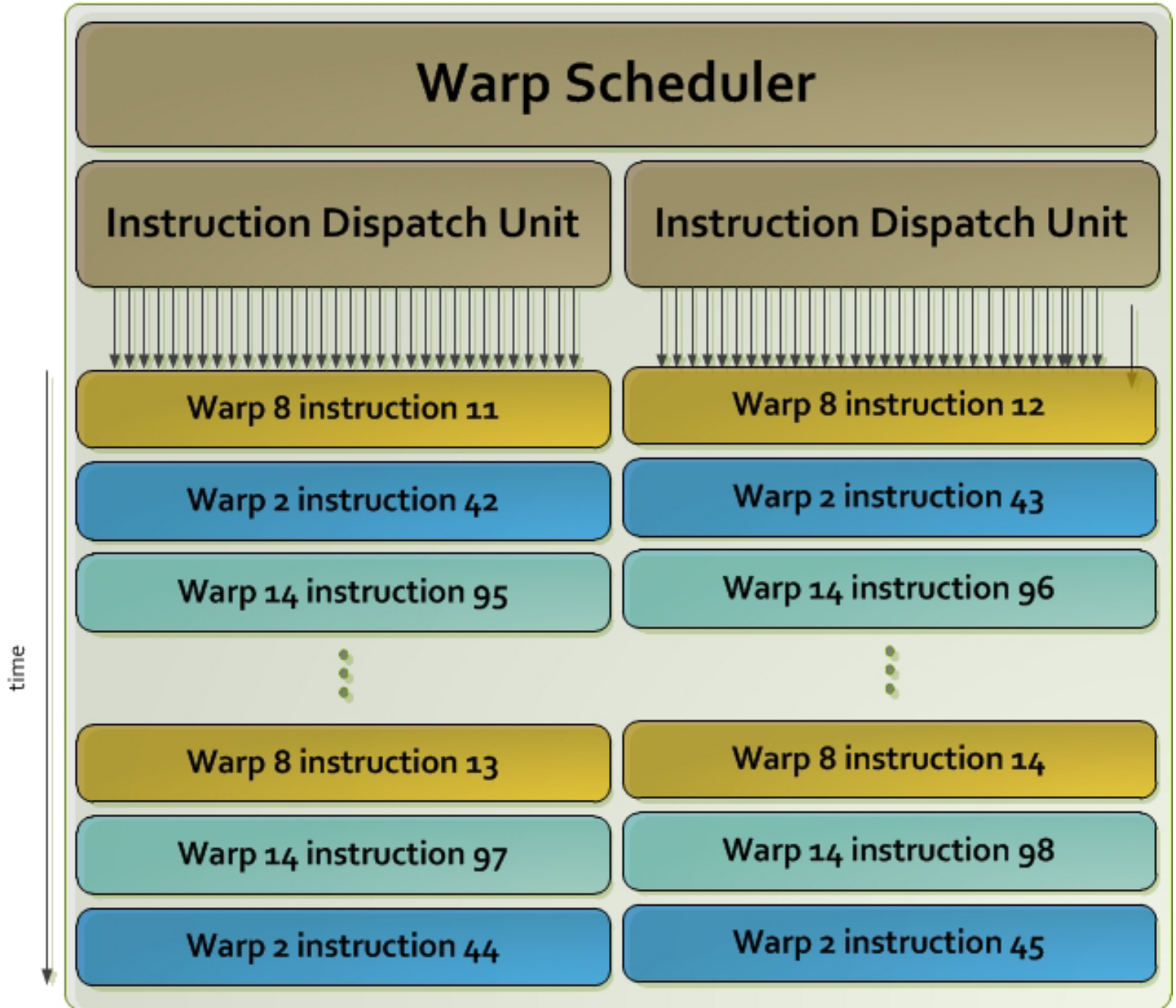
Quad Warp Scheduler

The SMX schedules threads in groups of 32 parallel threads called warps .

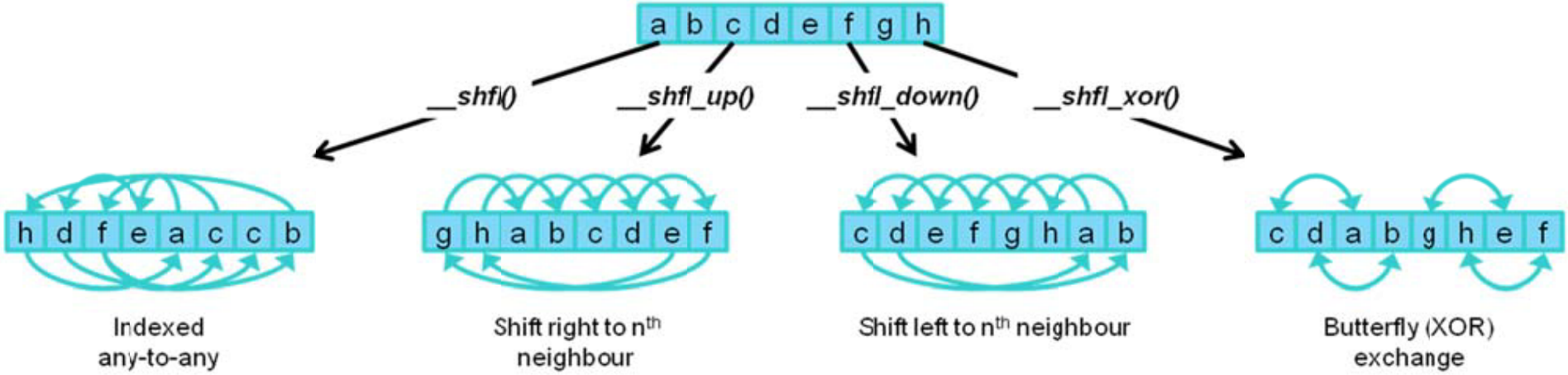
Each SMX features four warp schedulers and eight instruction dispatch units, allowing four warps to be issued and executed concurrently. (128 threads)

Kepler GK110 allows double precision instructions to be paired with other instructions.

One Warp Scheduler Unit

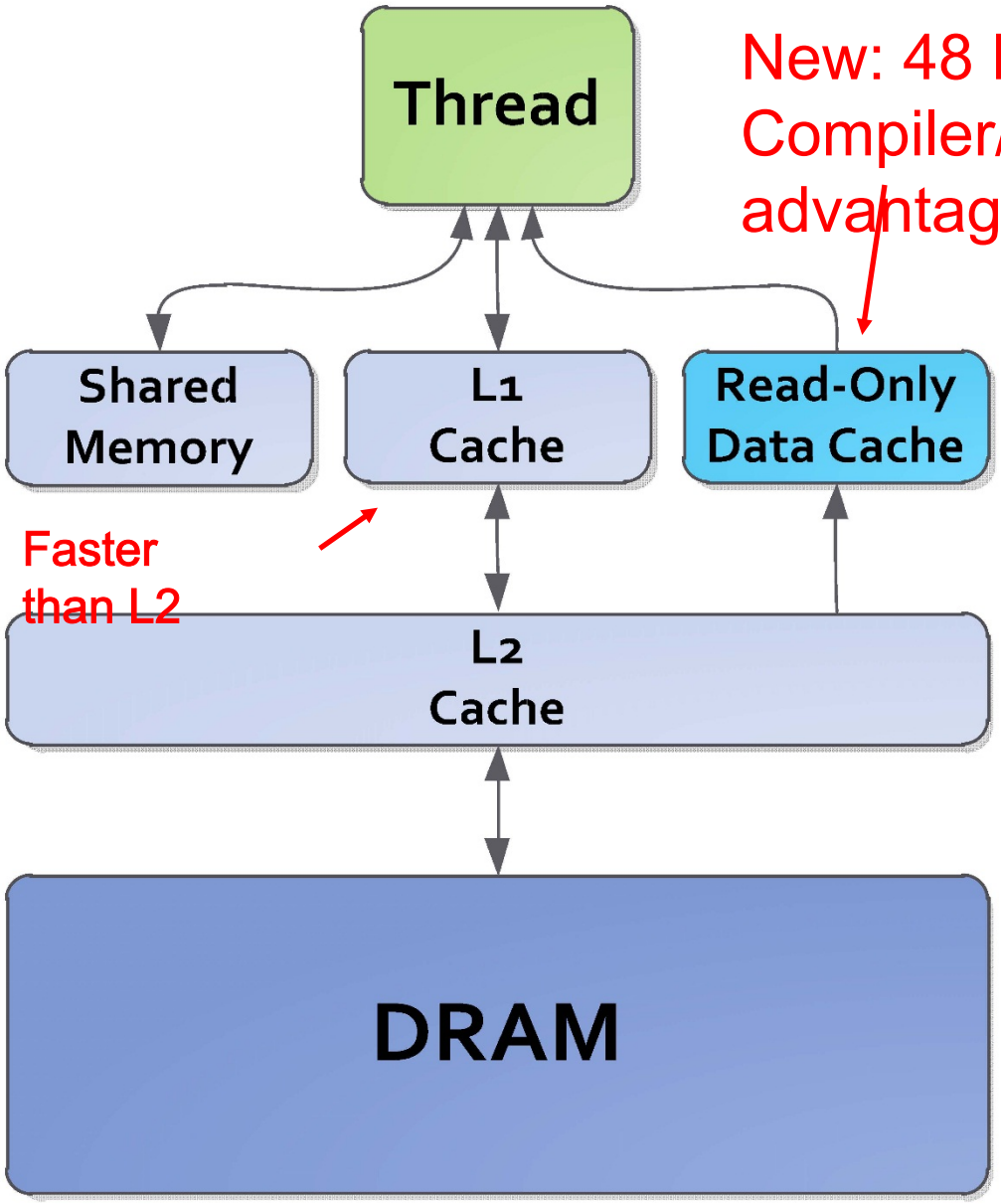


- Each thread can access up to 255 registers (x4 of Fermi)
- New Shuffle instruction which allows threads within a warp to share data without passing data through shared memory:



- Atomic operations: Improved by 9x to one operation per clock ã fast enough to use frequently with kernel inner loops

Kepler Memory Hierarchy



New: 48 KB Read-only memory cache
Compiler/programmer can use to advantage

Shared memory/L1 cache split:
Each SMX has 64 KB onợ chip memory, that can be configured as:

- 48 KB of Shared memory with 16 KB of L1 cache,

or

- 16 KB of shared memory with 48 KB of L1 cache

or

- (new) a 32KB / 32KB split between shared

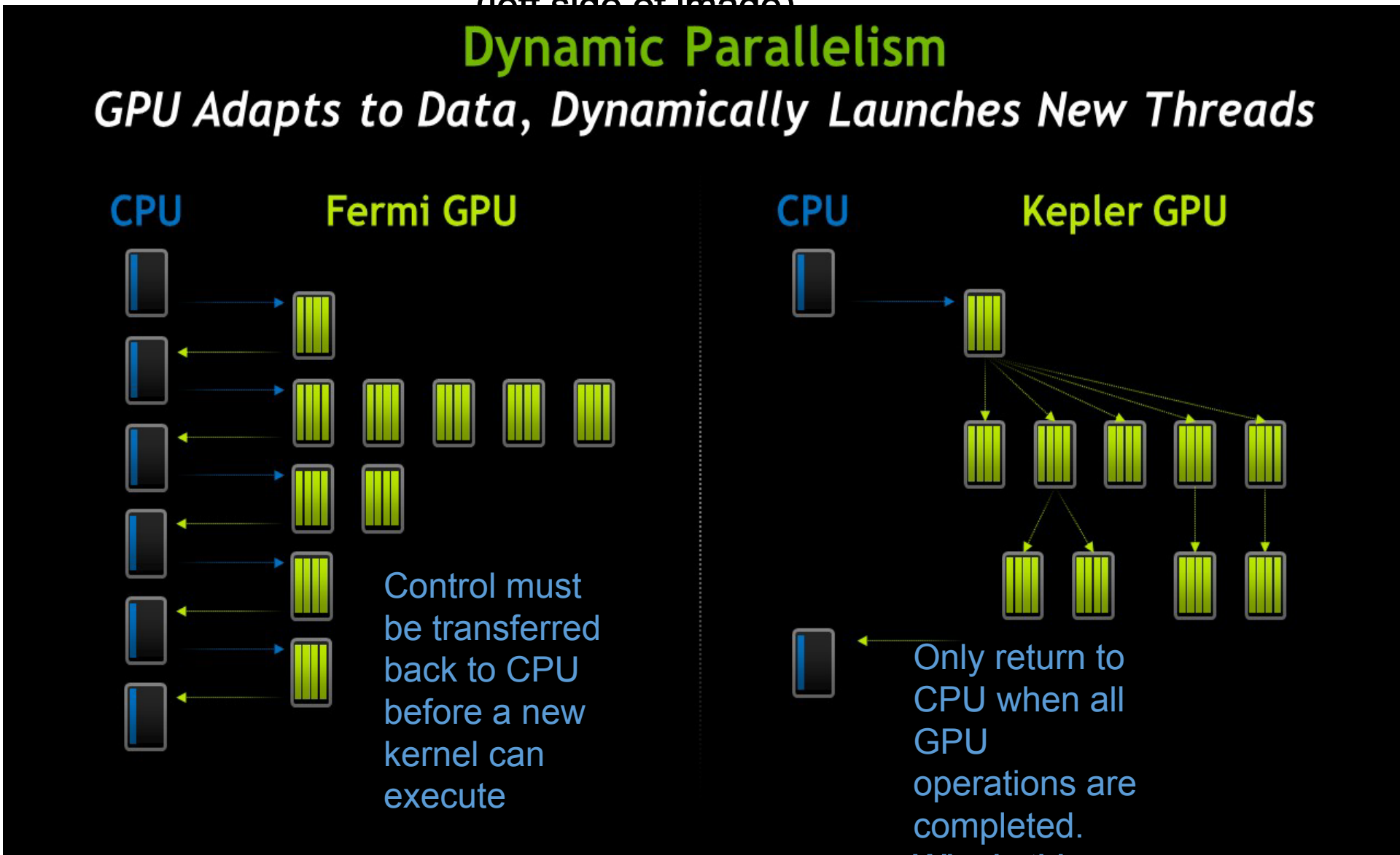
Dynamic Parallelism

- Fermi could only launch one kernel at a time on a single device. Kernel had to complete before calling for another GPU task.

- ’In Kepler GK110 any kernel can launch another kernel , and can create the necessary streams, events and manage the dependencies needed to process additional work without the need for host CPU interaction.˜

- ’ .. makes it easier for developers to create and optimize recursive and dataȩdependent

'Dynamic Parallelism allows more parallel code in an application to be launched directly by the GPU onto itself (right side of image) rather than requiring CPU intervention (left side of image)~



’With Dynamic Parallelism, the grid resolution can be determined dynamically at runtime in a data dependent manner. Starting with a coarse grid, the simulation can ’zoom in˜ on areas of interest while avoiding unnecessary calculation in areas with little

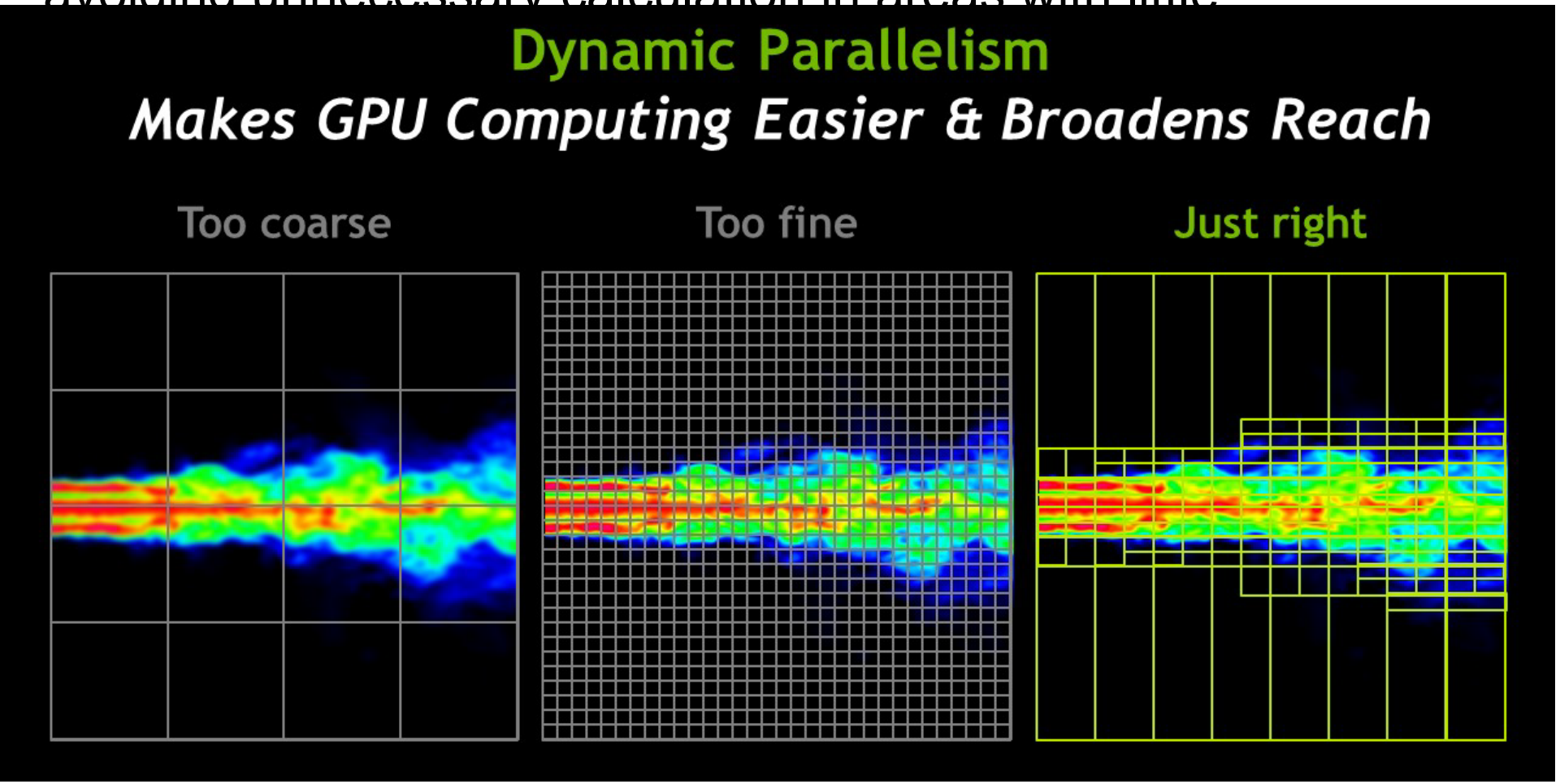


Image attribution Charles Reid

HyperQ

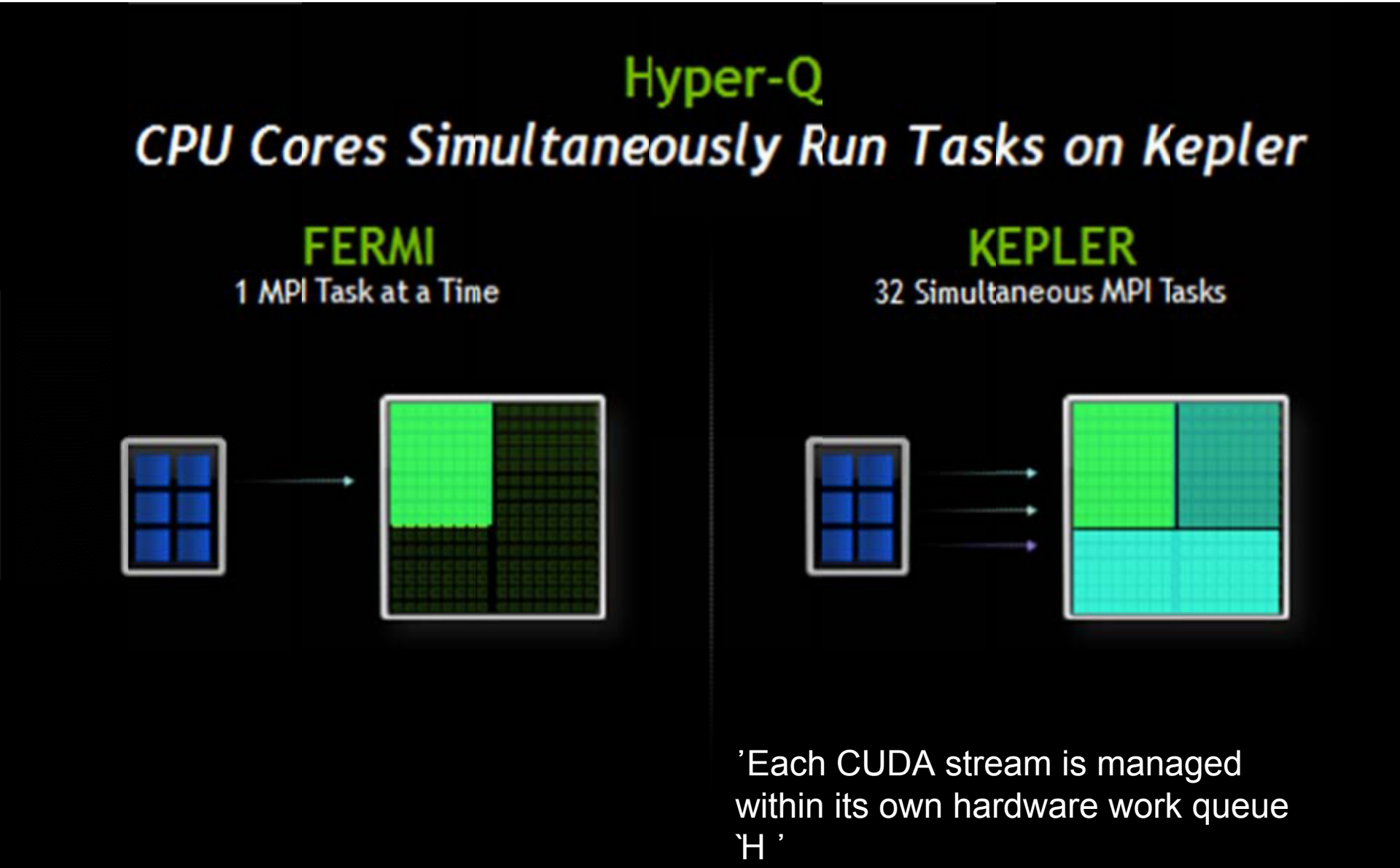
’The Fermi architecture supported 16-way concurrency of kernel launches from separate streams, but ultimately the streams were all multiplexed into the same hardware work queue .˜

’Kepler GK110 `H HyperQ increases the total number of connections (work queues) `H by allowing 32 simultaneous, hardware-managed connections..˜

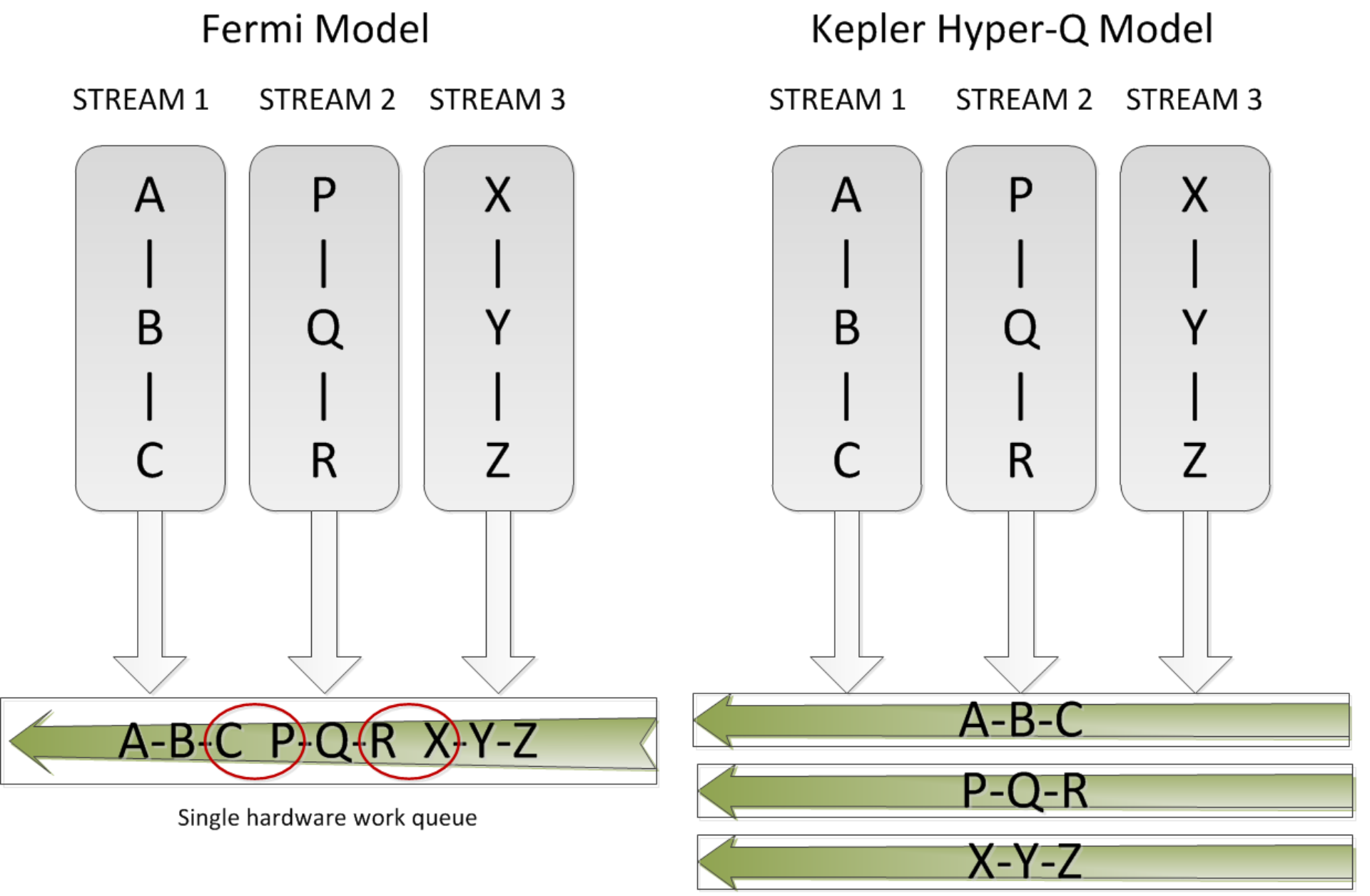
’`H allows connections from multiple CUDA streams, from multiple Message Passing Interface (MPI) processes, or even from multiple threads within a process.

Applications that previously encountered false

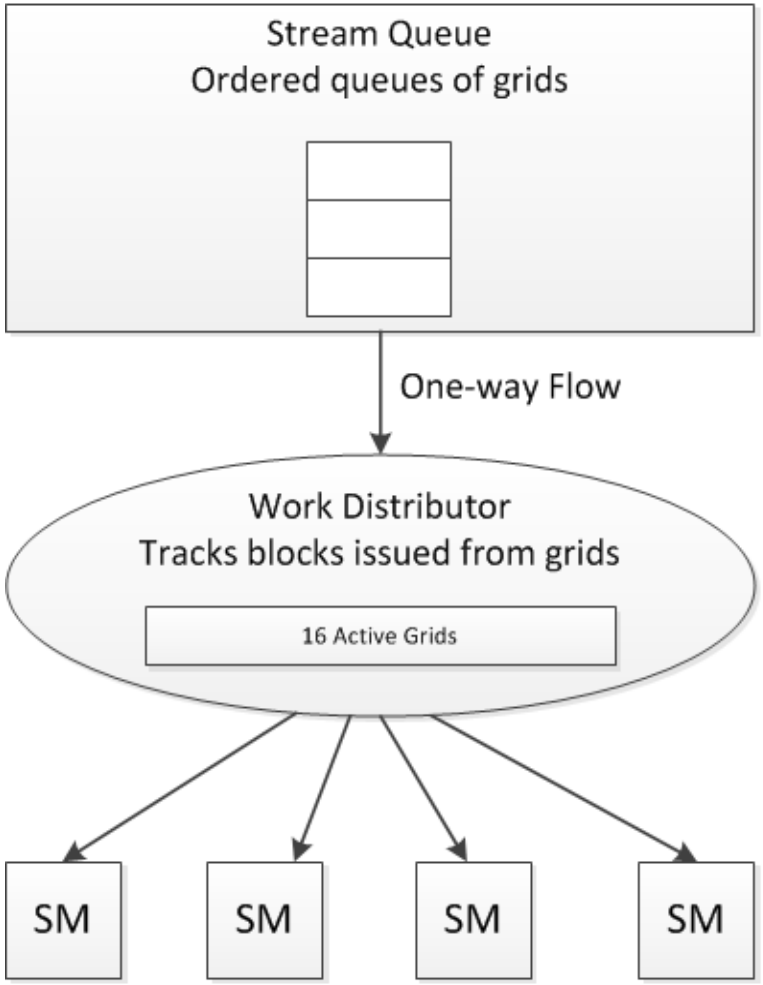
HyperQ



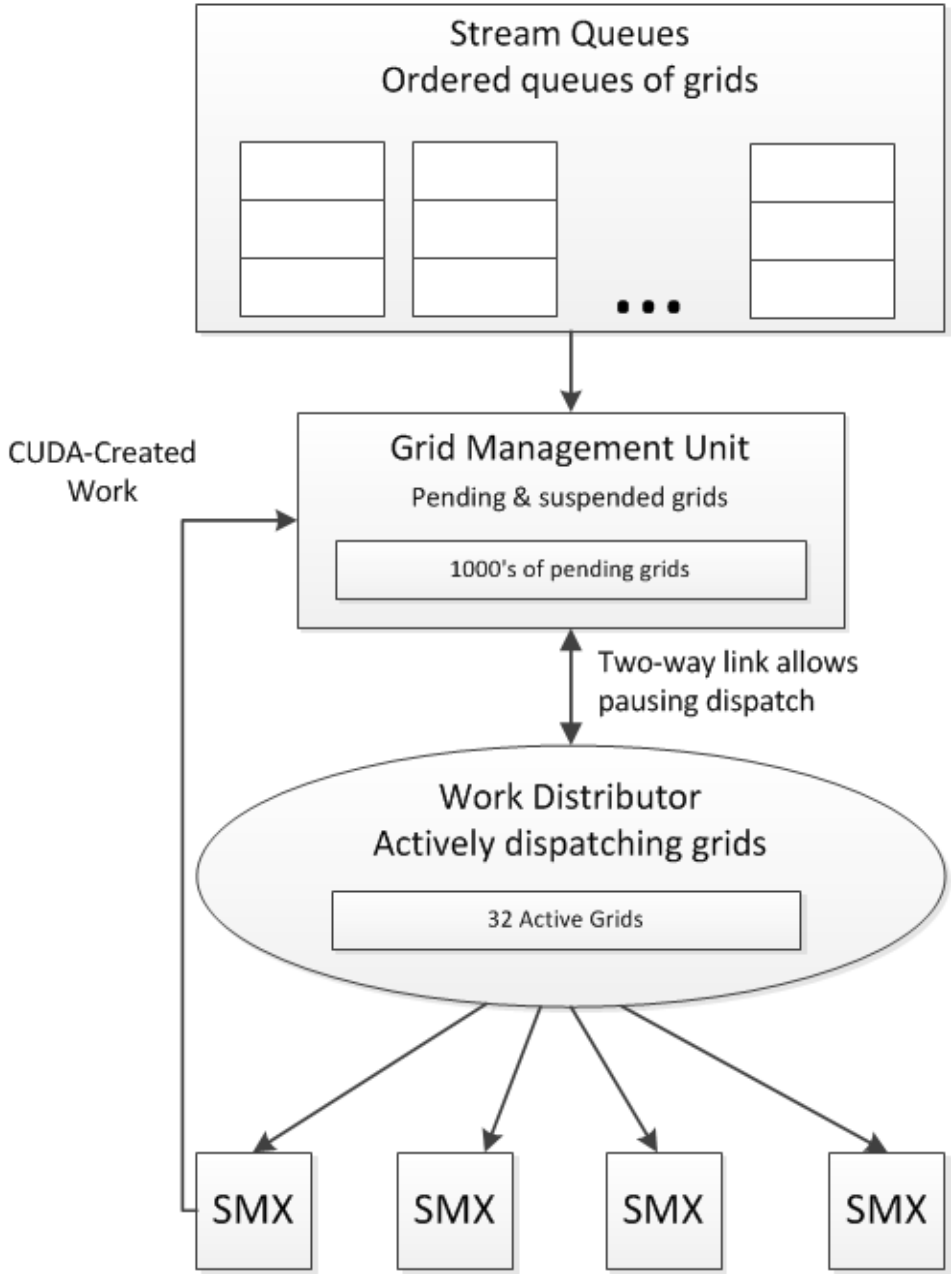
’Each CUDA stream is managed within its own hardware work queue
`H ’


Fermi Model
STREAM 1
STREAM 2
STREAM 3
A
B
C
P
Q
R
X
Y
Z
A-B-C P-Q-R X-Y-Z
Single hardware work queue
Kepler Hyper-Q Model
STREAM 1
STREAM 2
STREAM 3
A
B
C
P
Q
R
X
Y
Z
A-B-C
P-Q-R
X-Y-Z
Each stream receives its own work queue

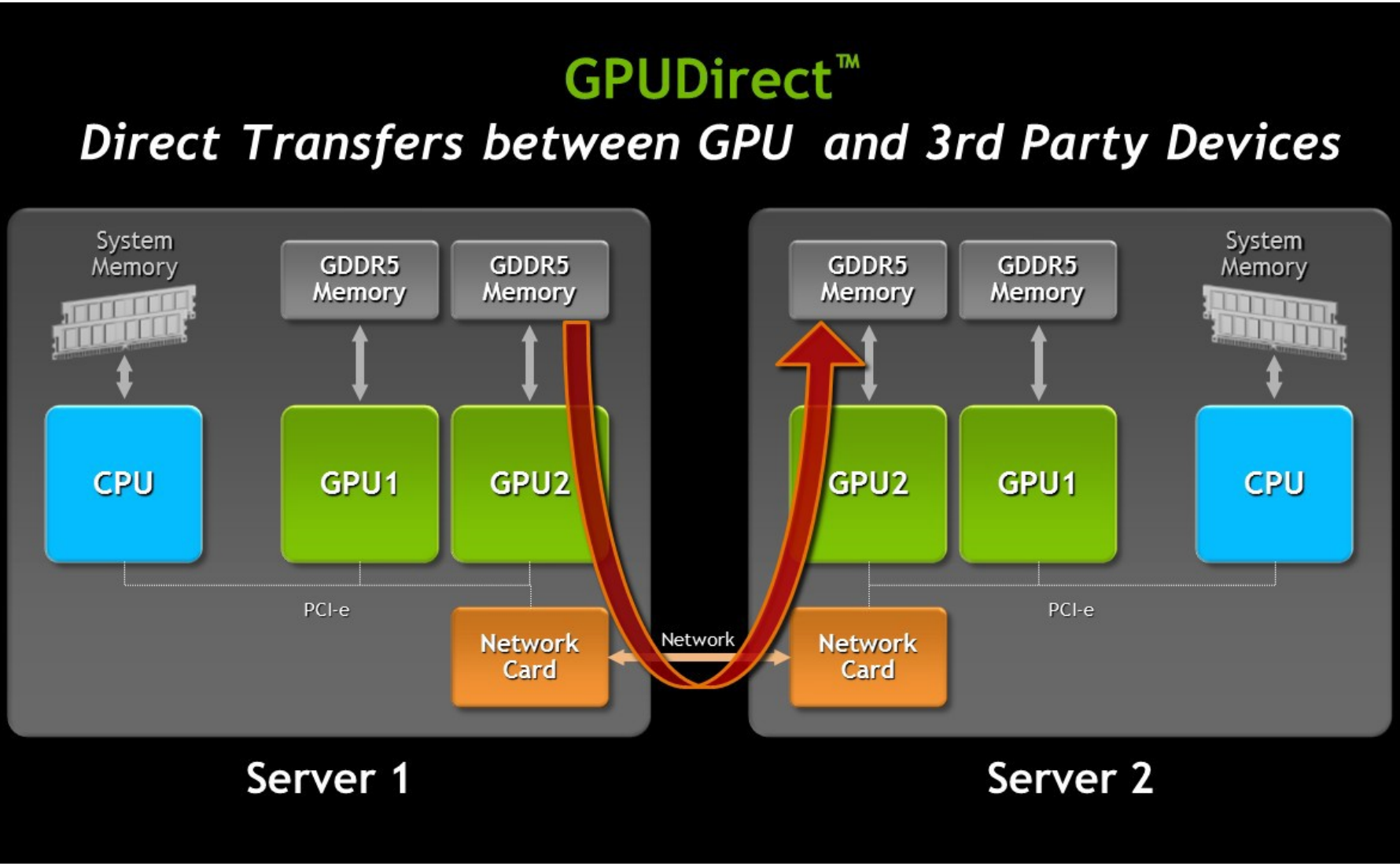
Fermi Workflow



Kepler Workflow



'The redesigned Kepler HOST to GPU workflow shows the new Grid Management Unit, which allows it to manage the actively dispatching grids, pause dispatch and hold



’GPUDirect RDMA allows direct access to GPU memory from 3rdą party devices such as network adapters, which translates into direct transfers between GPUs *across* nodes as well.˜

The implementation of GPGPU in Model Checking

1. LTL Model Checking:
   - J. Barnat, P. Bauch, L. Brim, and M. Ceska. Computing strongly connected components in parallel on cuda. In IPDPS 2011, pages 544ã 555. IEEE, 2011.
   - J. Barnat, P. Bauch, L. Brim, andM. Ceska. Designing fast LTL model checking algorithms for many-core GPUs. Journal of Parallel and Distributed Computing, 72(9):1083ã1097, 2012.
   - Edelkamp, Stefan and Sulewski, Damian. Efficient Explicit-state Model Checking on General Purpose Graphics Processors. In SPIN, pages 106ã 123. Springer, 2010.
2. Probilistic Model :
   - D. Bosnacki, S. Edelkamp, D. Sulewski, and A. Wijs. Parallel probabilistic model checking on general purpose graphics processors. International Journal on Software Tools for Technology Transfer, 13(1): 21ã35, 2011.
   - A. J. Wijs and D. Bosnacki. Improving GPU sparse matrix-vector multiplication for probabilistic model checking. In Model Checking Software, pages 98ã116. Springer, 2012.
3. State Space Exploration
   - S. Edelkamp and D. Sulewski. Parallel state space search on the gpu. In Proceedings of the International Symposium on Combinatorial Search, 2009

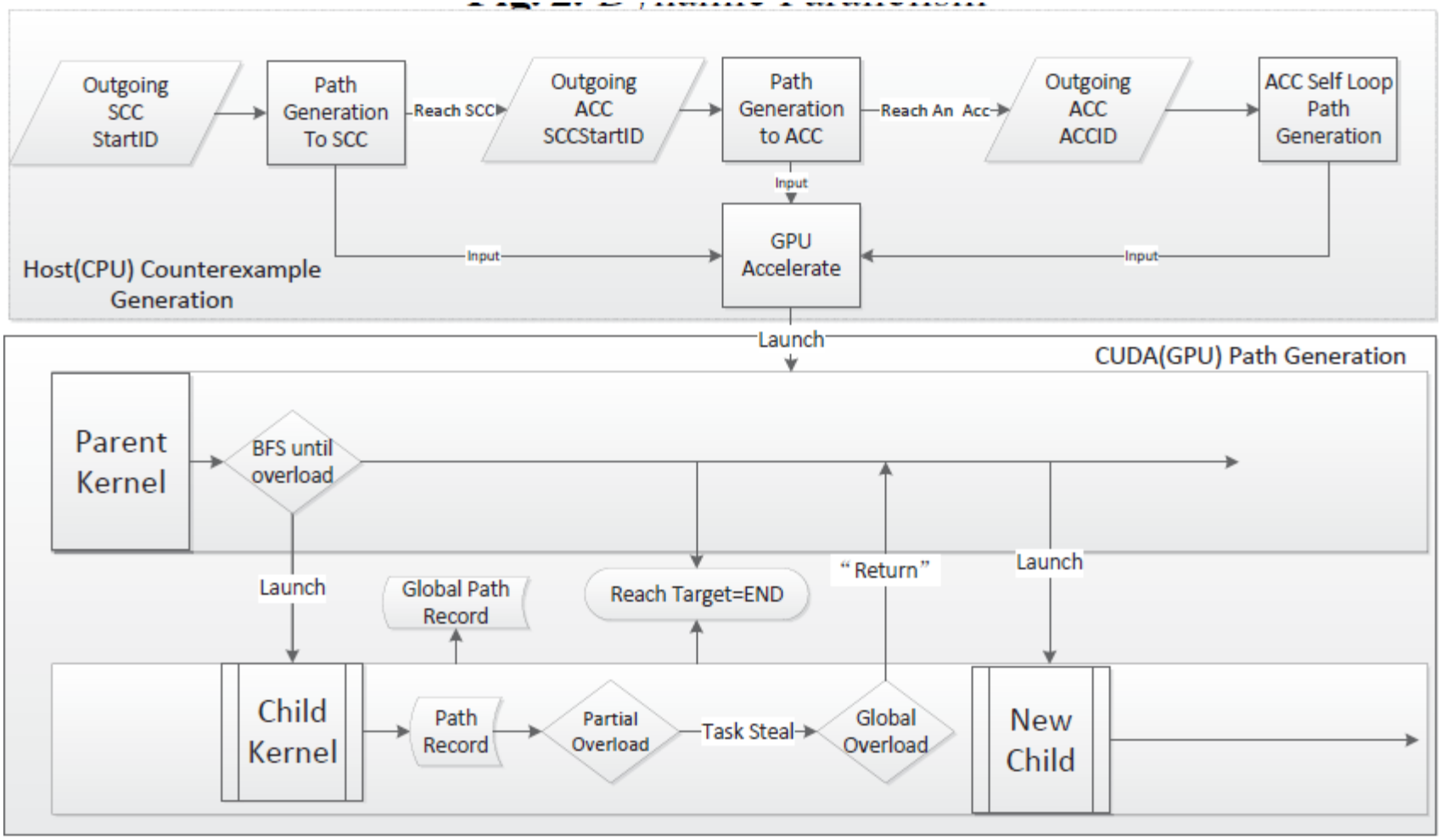
My current work

Implementation of GPU in BFS-related Model Checking Problems:

1. Dynamic Counterexample Generation without CPU involvement

State Space: Known

Abstract: Strongly Connected Component (SCC) based searching is one of the popular LTL model checking algorithms. When the SCCs are huge, the counterexample generation process can be time consuming, especially when dealing with fairness assumptions. In this work, we propose a GPU accelerated counterexample generation algorithm, which improves the performance by paralleling the Breadth First Search (BFS) used in the counterexample generation. BFS work is irregular, which means it is hard to allocate resources and may suffer from im-balance load. We make use of the features of latest CUDA Compute Architecture-NVIDIA Kepler GK110 to achieve the dynamic parallelism and memory hierarchy handle the irregular searching pattern in BFS. We build dynamic queue management, task scheduler and path recording such that the counterexample generation process can be completely taken by GPU without involving CPU. We have implemented the proposed approach in PAT model checker. Our experiments show that our approach is effective and scalable.



2. On-the-fly Deadlock Verification:

State Space: Unknown

Key Ideas: Compact Encoding, state space. Collaborative Synchronization based on SIMD. Hierarchical Hash. Structure for successor generation

Thank you