Failure modes and failures mitigation in GPGPUs: A reference model and its application

Francesco Terrosi
francesco.terrosi@unifi.it

Andrea Ceccarelli
andrea.ceccarelli@unifi.it

Andrea Bondavalli
andrea.bondavalli@unifi.it

University of Florence – Italy

RCL
RESILIENT COMPUTING LAB

UNIVERSITÀ DEGLI STUDI FIRENZE
DIMAI
DIPARTIMENTO DI MATEMATICA E INFORMATICA "ULISSE DINI"

UNIVERSITÀ
DEGLI STUDI
FIRENZE
DIMAI
DIPARTIMENTO DI
MATEMATICA E INFORMATICA
"ULISSE DINI"

Introduction

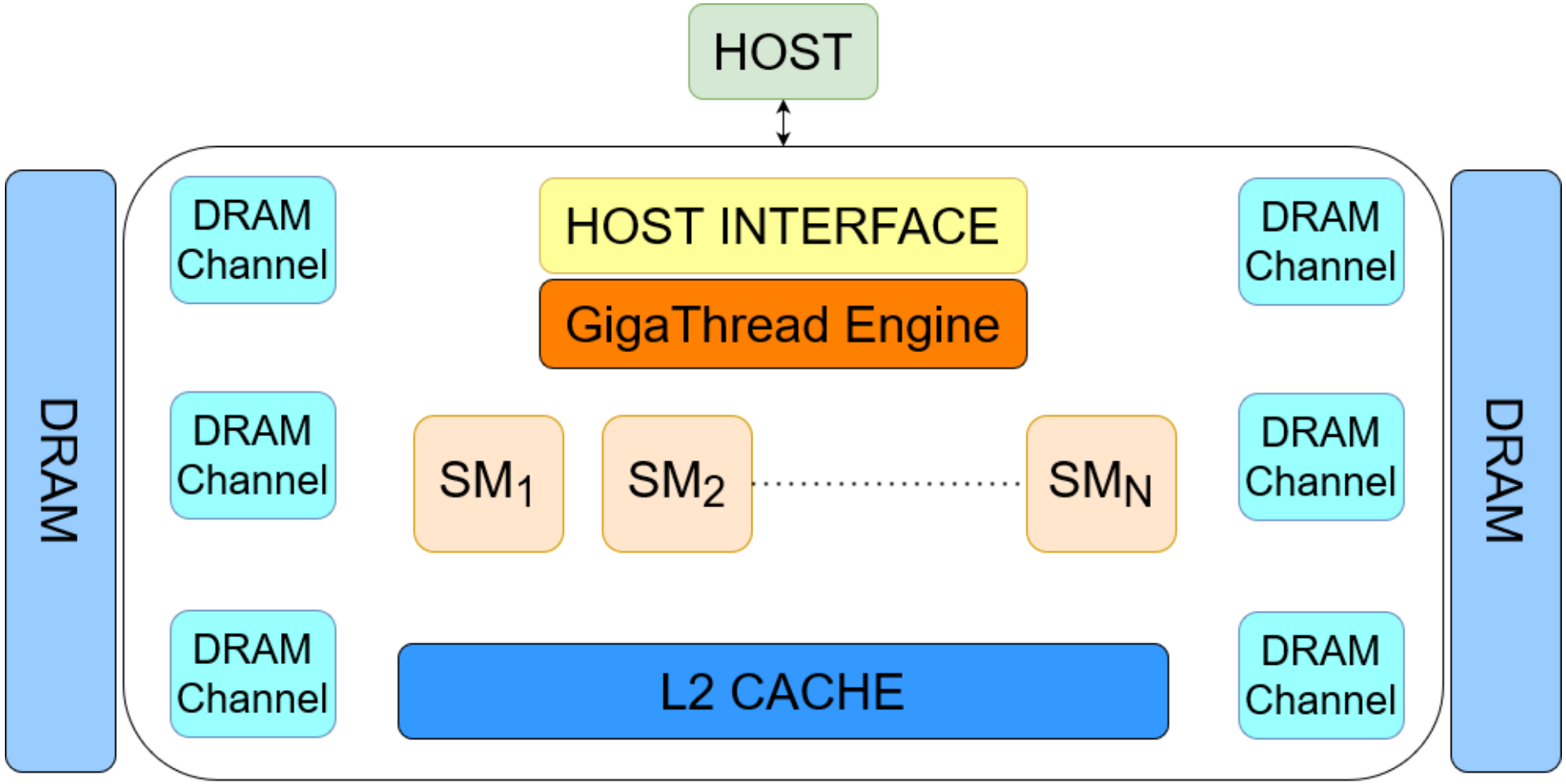
- General-Purpose Graphics Processing Units (GPGPUs) are now used in fields such as High-Performance Computing and Machine Learning applications
- This led researchers and practitioners to focus on the fault-tolerance properties of these devices
- We propose a reference model to characterize the failure modes of a GPGPU at different layers

GPGPU Architecture

UNIVERSITÀ DEGLI STUDI FIRENZE
DIMAI
DIPARTIMENTO DI MATEMATICA E INFORMATICA "ULISSE DINI"

GPU Architecture – GPU

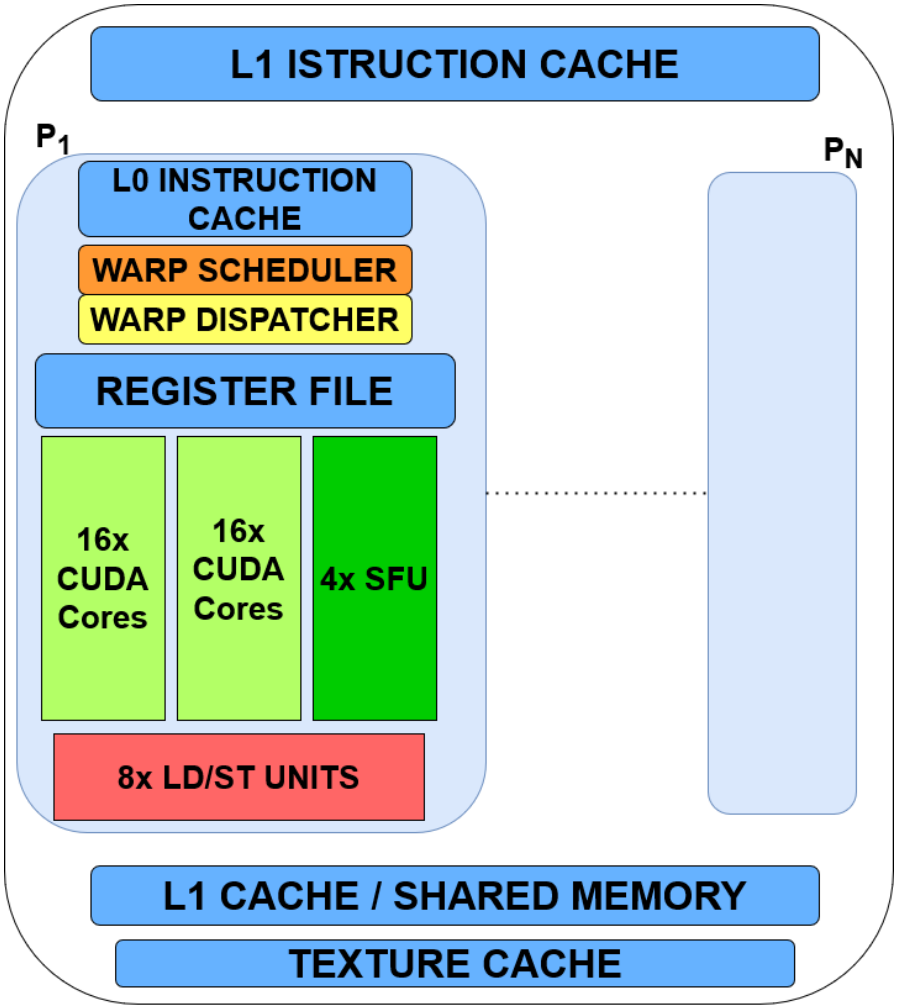
- GPU – TOP LEVEL:
  - global memory (L2 Cache + possible off-chip DRAM), GigaThread Engine (the Global Scheduler), Host Interface, DRAM Channels and the Streaming Multiprocessors (SM).



UNIVERSITÀ DEGLI STUDI FIRENZE
DIMAI
DIPARTIMENTO DI MATEMATICA E INFORMATICA "ULISSE DINI"

GPU Architecture
Streaming Multiprocessor

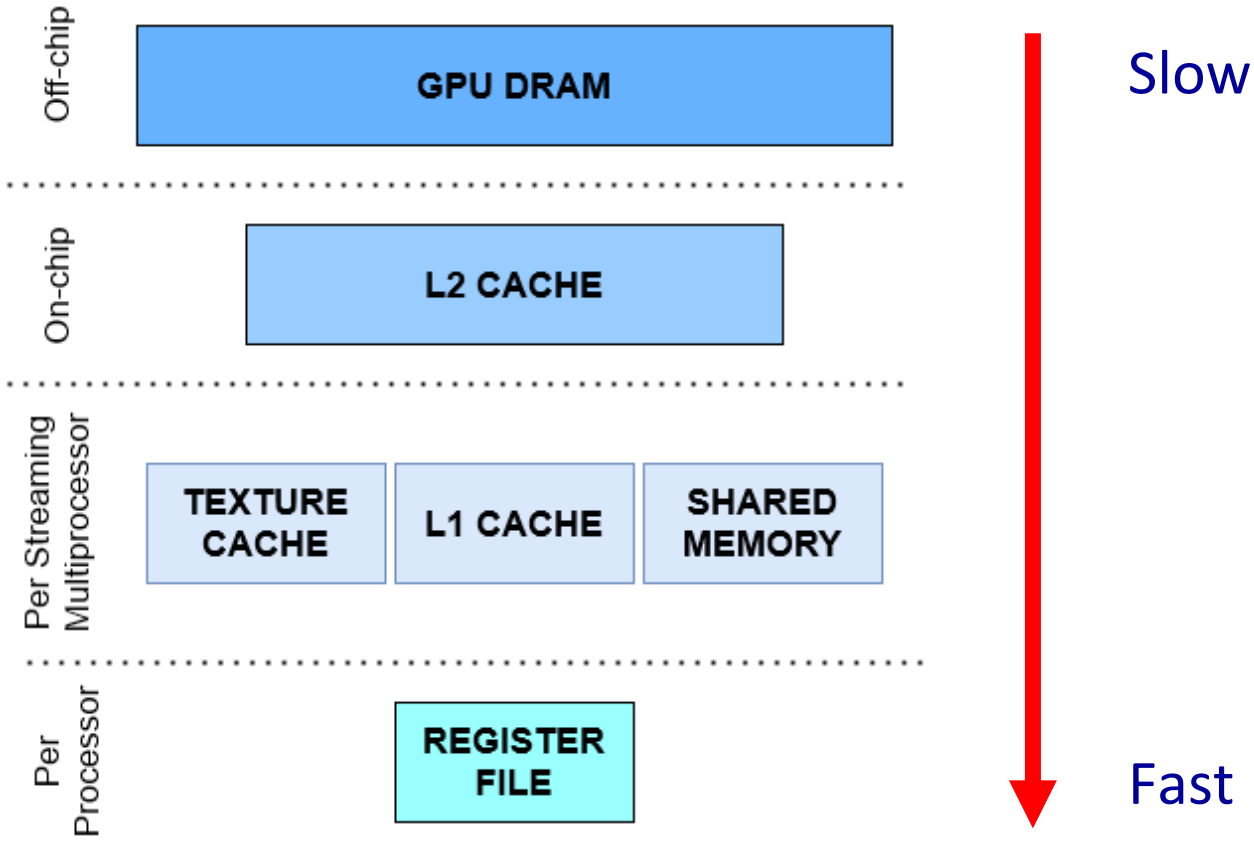
- STREAMING MULTIPROCESSOR: Clusters of processors, each processor has its own resources and shares the same L1 Caches
  - inside an SM, groups of 32 threads (warps) are scheduled per clock cycle



UNIVERSITÀ DEGLI STUDI FIRENZE
DIMAI
DIPARTIMENTO DI MATEMATICA E INFORMATICA "ULISSE DINI"

GPU Architecture
Memory Hierarchy

- The memory hierarchy is designed to *maximize the bandwidth* of data transfers
  - The L2 Cache is designed as a *bandwidth filter* for DRAM



GPU Services and Failure Modes

UNIVERSITÀ DEGLI STUDI FIRENZE
DIMAI
DIPARTIMENTO DI MATEMATICA E INFORMATICA "ULISSE DINI"

GPU Services and Failure Modes

- GPGPUs are made of many similar *replicated* components

- We can group these components in three categories:
  - *Computation components (Streaming Multiprocessors, GigaThread Engine and Warp Schedulers, CUDA Cores, Load/Store Units)*
  - *Memory components (DRAM, Cache memories, Register Files)*
  - *Communication components (Host Interface, DRAM Channels)*

UNIVERSITÀ DEGLI STUDI FIRENZE
DIMAI
DIPARTIMENTO DI MATEMATICA E INFORMATICA "ULISSE DINI"

GPU Services and Failure Modes

- Computation Components: *Compute* service
  - The process of performing arithmetic/logic operations
- Memory Components: *Read, Write* services
  - The *read* service is the process of retrieving (read) for a memory location specified by its *address*
  - The *write* service is the process of storing (write) data in a memory location specified by its *address*
- Communication Components: *Send, Receive* services
  - *Send* is the process of sending data over a channel
  - *Receive* is the process of receiving data being transferred over a channel

UNIVERSITÀ DEGLI STUDI FIRENZE
DIMAI
DIPARTIMENTO DI MATEMATICA E INFORMATICA "ULISSE DINI"

GPU Services and Failure Modes

- The GPU can be seen as a *system* providing a *set of services*
- These services are offered by the GPU thanks to the combination of the services offered by its components
- The outcome of each service may be *correct* or *incorrect*
- To analyze the outcomes of these services, we define two functions:

1. Correct Service: CS(*component*, *service*, *input*, t_i, t_{min}, t_{max}) = (v, t_v)
2. Delivered Service: DS(*component*, *service*, *input*, t_i, t_{min}, t_{max}) = (w, t_w)

GPU Services and Failure Modes

CORRECT SERVICE

- CS(*component*, *service*, *input*, t_i, t_{min}, t_{max}) = (v, t_v)
- DS(*component*, *service*, *input*, t_i, t_{min}, t_{max}) = (w, t_w)

a) The service delivered by a component is said to be *correctly valued* if w = v

b) The service delivered by a component is said to be *correctly timed*, in the presence of a finite timing constraint defined by the lower and upper bounds t_{min} and t_{max} if:

$$t_i + t_{min} \leq t_w \leq t_i + t_{max}$$

UNIVERSITÀ DEGLI STUDI FIRENZE
DIMAI
DIPARTIMENTO DI MATEMATICA E INFORMATICA "ULISSE DINI"

GPU Services and Failure Modes

- CS(GPU, Compute, x, t_i, t_{min}, t_{max}) = (y, t_y) CORRECT

- CONTENT FAILURE
  - DS(GPU, Compute, x, t_i, t_{min}, t_{max}) = (z, t_y) WRONG
  - DS(GPU, Compute, x, t_i, t_{min}, t_{max}) = (*null*, t_y) NOT

- TIMING FAILURE
  - DS(GPU, Compute, x, t_i, t_{min}, t_{max}) = (y, t_z) EARLY $t_z < t_i + t_{min}$
  - DS(GPU, Compute, x, t_i, t_{min}, t_{max}) = (y, t_z) LATE $t_z > t_i + t_{max}$

- CONTENT AND TIMING FAILURE
  - DS(GPU, Compute, x, t_i, t_{min}, t_{max}) = (z, t_z) ERRATIC $t_z < t_i + t_{min}$ or $t_z > t_i + t_{max}$

GPU Failure Detection and Recovery

UNIVERSITÀ
DEGLI STUDI
FIRENZE
DIMAI
DIPARTIMENTO DI
MATEMATICA E INFORMATICA
"ULISSE DINI"

GPU Failure Detection

- The last section of the work consists in a study of the robustness of a GPGPU in three steps:

1. Literature review of the protection techniques developed for GPGPUs (novel techniques or adapted from the CPU domain)

2. What are the *component and/or service* protected by a specific technique

3. What are the *failures* protected by the specific technique

UNIVERSITÀ DEGLI STUDI FIRENZE
DIMAI
DIPARTIMENTO DI MATEMATICA E INFORMATICA "ULISSE DINI"

GPU Compute Failure Detection

- Computation components: replication with comparison at different granularities:
  - Global replication – GPU (adapted by CPU)
  - Intra-thread replication – Thread level (adapted by CPU)
  - Inter-thread replication – Warp level – (leverages inner redundancy of GPUs)

- Hardware Checkers – Comparison is perform using specialized hardware (adapted by CPU, hard for GPU)

- Warp Scheduler protection – Software modification technique that allows to detect failures in the warp scheduler units (specific for GPUs)

UNIVERSITÀ DEGLI STUDI FIRENZE
DIMAI
DIPARTIMENTO DI MATEMATICA E INFORMATICA "ULISSE DINI"

GPU Memory Failure Detection

- Memory components are one of the main failure causes in GPGPUs (high temperatures; frequent accesses to memory to read/write *streams* of data)

- This led NVIDIA to introduce a Single Error Correction Double Error Detection Error Correction Code (SECDED ECC) in all the key memory structures of their GPUs

- SECDED ECC can correct 1-bit errors, and detect 2 bits-errors

UNIVERSITÀ DEGLI STUDI FIRENZE
DIMAI
DIPARTIMENTO DI MATEMATICA E INFORMATICA "ULISSE DINI"

GPU Communication Failure Detection

- GPUs use communication components to communicate with the Host system and the off-chip DRAM, namely the Host Interface and the DRAM Channels

- These buffered channels are protected by a Cyclic Redundancy Check (CRC) mechanism with retry

- The Host Interface is implemented with the Peripheral Component Interconnect Express (PCIe) standard, which comes with many built-in error-detection capabilities (and CRC with retry)

UNIVERSITÀ DEGLI STUDI FIRENZE
DIMAI
DIPARTIMENTO DI MATEMATICA E INFORMATICA "ULISSE DINI"

GPU Computation Failure Recovery

- Transient failures in Computation Components can be recovered with *triple replication*
  - Perform 3 execution of the same instruction, if there is agreement on at least 2 outputs, the failure can be masked

- Permanent failures in SMs require to isolate the failed SM. The proposed approach is to redefine the scheduling strategy to avoid using the corrupted unit, at the cost of reducing throughput by 1/N (where N is the amount of SMs)

UNIVERSITÀ
DEGLI STUDI
FIRENZE
DIMAI
DIPARTIMENTO DI
MATEMATICA E INFORMATICA
"ULISSE DINI"

GPU Memory Failure Recovery

- The SECDED ECC mechanism *always correct* 1-bit failures. 2-bits failures are somehow "corrected" with a *page* (i.e., DRAM row) retirement mechanism, preventing the GPU to use the corrupted memory area in future executions
  - *This works both for transient and permanent failures*

- In the case of a permanent failure in one of the memories, continuously applying this mechanism could reduce the overall throughput of the GPU

UNIVERSITÀ
DEGLI STUDI
FIRENZE
DIMAI
DIPARTIMENTO DI
MATEMATICA E INFORMATICA
"ULISSE DINI"

GPU Communication Failure Recovery

- Transient failures in communication components are corrected by the CRC with retry mechanism

- Permanent failures in *communication components* can not be recovered with the present mechanism. However, it was reported that some permanent failures in the Host Interface could be recovered just by cleaning the GPU connectors

UNIVERSITÀ DEGLI STUDI FIRENZE
DIMAI
DIPARTIMENTO DI MATEMATICA E INFORMATICA "ULISSE DINI"

GPU Global Protection

	GPU	Streaming Multiprocessor	Memory Hierarchy	GigaThread Engine	Host Interface	DRAM Channels
WRONG	Global Replication	Inter/Intra-thread Replication Hardware/Software Checkers	SECDED ECC	Not protected	PCI Express Interface CRC with Retry Advanced Error Reporting	CRC With Retry
NOT			Not protected			
EARLY		Inter/Intra-thread Replication	Not protected			
LATE			Not protected			
ERRATIC		Global Replication	SECDED ECC (Value only)			

Summarizing View

UNIVERSITÀ
DEGLI STUDI
FIRENZE
DIMAI
DIPARTIMENTO DI
MATEMATICA E INFORMATICA
"ULISSE DINI"

Summarizing View

- The architecture of a GPU is redundant by design, it can be decomposed at different granularities
- GPU and its components provides a set of services which may be *correct* or *incorrect*
- Currently available fault-tolerance techniques for GPUs were reviewed with respect to the protected component, and the failure mode
- Results showed that some components still need protection