Learning the Architecture of NVIDIA GPUs:

We will look at the Microarchitecture **Maxwell** found in the **GTX 900** family.

Graphical user interface

Description automatically generated

Graphical user interface

Description automatically generatedThe ***GPU architecture*** is ***built around*** **a *scalable array of streaming*** **multiprocessor** or ***SMs.* GPU hardware parallelism** is ***achieved through by replication of this architectural building block***. In this diagram, the SMs the image at the left.

Table

Description automatically generatedWe can count **16** such ***streaming multiprocessors(SMs)*** in this GPU. Those streaming multiprocessors **have access to device inbuilt global memory** and **L2 cache, memory controls** and **cache**.

Graphical user interface

Description automatically generatedThe key components of streaming multiprocessor are listed in this diagram. (We’ll use not the ***SM*** of the **Fermi microarchitecture** for this example.)

***Modern SM*** ***units*** are somewhat different than this, but we can clearly identify basic components from this diagram. We have:

* **CUDA cores**: like a CPU core and it will execute instructions fetch into that core.
* **Shared Memory & L1 Cache**: facilitate faster memory access
* **Registers**: limited amount of on-chip memory to store on the fly values for execution
* **Load and store Units**: perform memory load and store requests
* **Warp Schedulers**: determine which ***warp*** or which ***set of threads*** are going to execute next
* **Special Function Units**

We have to especially careful how we use resources like registers, ***shared memory and L1 cache***, as those are ***very limited resources on SM***.

***Warp schedulers*** will take resource consumption factor when it deciding which ***warp is going to execute next***.

***Modern SM are more complicated.***

Here this diagram shows you SM for the Volta microarchitecture.

A screenshot of a computer

Description automatically generated with medium confidenceAll these green squares are cores. But these SM have different types of cores like:

Graphical user interface, chart

Description automatically generated

* Tensor cores (TC)
* Integer operation execution cores (Int)
* FP32 cores (optimized to execute 32-bit floating-point operations faster)
* FP64 cores (optimized to execute 64-bit floating-point operations faster)

Computer architectures can be classified into **four categories** according to **Flynn’s taxonomy** (a classification of Computer Architectures):

Diagram

Description automatically generatedDiagram

Description automatically generatedSISD (Single Instructions Single Data): Represents **sequential computers** which exploits **no parallelism in either the instruction or data streams**. Here a ***single control unit*** will ***fetch a single instruction stream from memory***. The ***control unit*** then ***generates appropriate controls signal*** to ***direct the signal processing element*** to **operate on single data stream** *that is* ***on operation at a time***.

SIMD (Single Instruction Multiple Data): Represents the **organization** of **single computer** containing **control unit**, **processor unit** and **memory unit**. The *instructions* are going to *execute sequentially*. This can be **achieved** by ***pipelining or multiple function units***.

Diagram, timeline

Description automatically generated

Diagram

Description automatically generatedMIMD (Multiple Instruction Single Data): Here ***multiple instruction operates*** ***on one data stream***. This is an **uncommon architecture** which is generally **used for fault tolerance** (the *ability of a system* to ***continue operating without interruption*** when *one or more of its components fail*). Here heterogeneous systems will operate on same data stream and ***must agree on the results***. Examples: Space Shuttle Flight Control Computers.

MIMD (Multiple Instruction Multiple Data): Represents **multiple autonomous processors** ***simultaneously executing different instruction on different data***. *MIMD architectures* includes **multicore super scalar processors** and it *distribute its system* *using* either **one shared memory** **space** or **distributed memory space**.

Cuda follows a version of computer architecture called SIMT.

SIMT is an **extension of SIMD model** with **multithreading** (**single instruction** is going to **run on multiple threads**)

In CUDA, **thread block** is going to *execute* in ***single streaming multiprocessor***. **Multiple** **thread blocks** can be **executed simultaneously** on **same streaming multiprocessor** *depending on the resource limitation* of that ***SM***.

But **one** **thread block** **CANNOT** be executing in ***multiple SMs***. If the *device* ***cannot run single thread block*** in ***one SM*** (**EX:** If the needed **shared memory** **count** for a *particular block* is ***greater than*** the ***available shared memory*** in ***SM***) then **an error will return for that kernel launch**. When a block is schedule to execute on ***SM***, it is ***executed so a single instruction is run by multiple threads***.

Graphical user interface, application

Description automatically generatedIn previous sections, we arrange a ***set of threads*** to ***set of blocks*** on a ***grid***. ***That's what we able to observe from software point of view***.

Chart, diagram

Description automatically generated with medium confidenceOn the hardware level, we **can map these execution as shown in the diagram**.

* The ***grid*** of the ***kernel launch*** is going **to map to device**.
* ***Thread blocks*** in this grid can be divide across **SM** in the device.
* A ***single thread*** in a ***thread block*** is going to be executed by **one single core**.
* And since **we have multiple cores** in a **single SM,** the *same instruction* can execute using ***multiple threads*** in the **SM** (to adhere the **SIMT** convention).

**DIFFERENT CUDA MICROARCHITECTURES CONTAIN DIFFERENT AND ENHANCED CAPABILITIES.**