# THE GPU COMPUTING ERA

wbg231

January 2023

## 1 Introduction

- this article is on the history and evolution of GPUs
- GPU's are really good for doing things in parallel make certain tasks much more computationally passable

### GPU computing evolution

- GPU's evolved to render graphics
- rendering graphics are inherently a parallel problem
- a graphics programmer writes a single threaded program that draws one pixel and the GPU runs multiple instances of that program in parallel
- these types of machines scale really well in parallel

### GPU technology development

- they were pushed by gaming more or less

### early GPUs

- first was made in 1999
- became more easily programable and flexible as time went on

### unified computing and graphics GPUs

- Cuda allowed programming in C on GPUs allowing for unified graphics and computing in the same language

## GPU computing systems

- at first single machine super computers with many GPUs were used
- this then moved to servers

## GPU computing eco systems

- the PGU computing eco system is expanding quickly CUDA is the language interface

## CUDA scalable and parallel architecture

- CUDA is a hardware adn software coprocesing architecture
- Enables GPU's with cuda to execute code written in common programming languages
- extends a single threaded model to work in parallel with a limited set of abstractions
- CUDA allows for the easy development of large highly scalable parallel programs
- CUDA programs are organized into a host program consisting of one or more sequential threads running on the host CPU and and one or more parallel kernels running on the GPU
- a kernel executes a sequential program on a set of light weight parallel threads
- threads are organized into blocks and these blocks are organized in a gird
- threads within a block can synchronize and communicate with each other quickly via block share memory
- threads from different blocks in teh same grid can coordinate via atomic operations in a global memory space.
- kernel grids can synchronize via global barriers if they are sequential
- thread blocks must be independent within the program allowing scalability for the gpu
- ideally want small simple programs that can be run in parallel across many different threads