Demystifying GPU architecture for AI processing

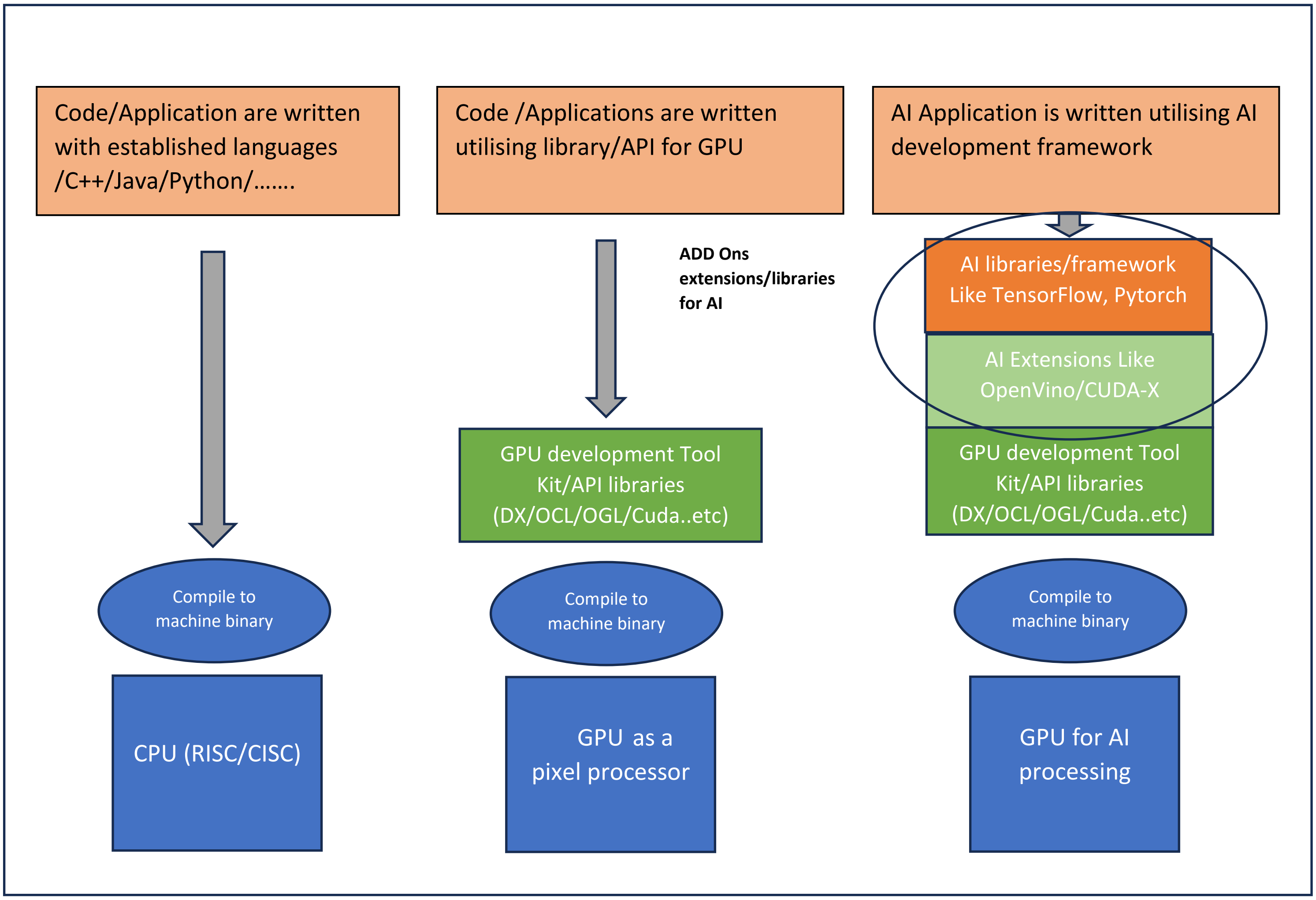
Blog 4 : Leveraging GPUs programming model and Eco system for AI applications

In Blog 3, we explored how GPUs' scalable parallel processing capabilities are well-suited for neural network processing. In this blog, we will examine the programming model and developer ecosystem of GPUs and how these can be leveraged to accelerate AI development.

The GPU supports a generic programming model with matured eco system

New inventions often face adoption challenges because their complexity creates a steep learning curve, requiring a significant investment to understand new architectures and develop appropriate programming models.

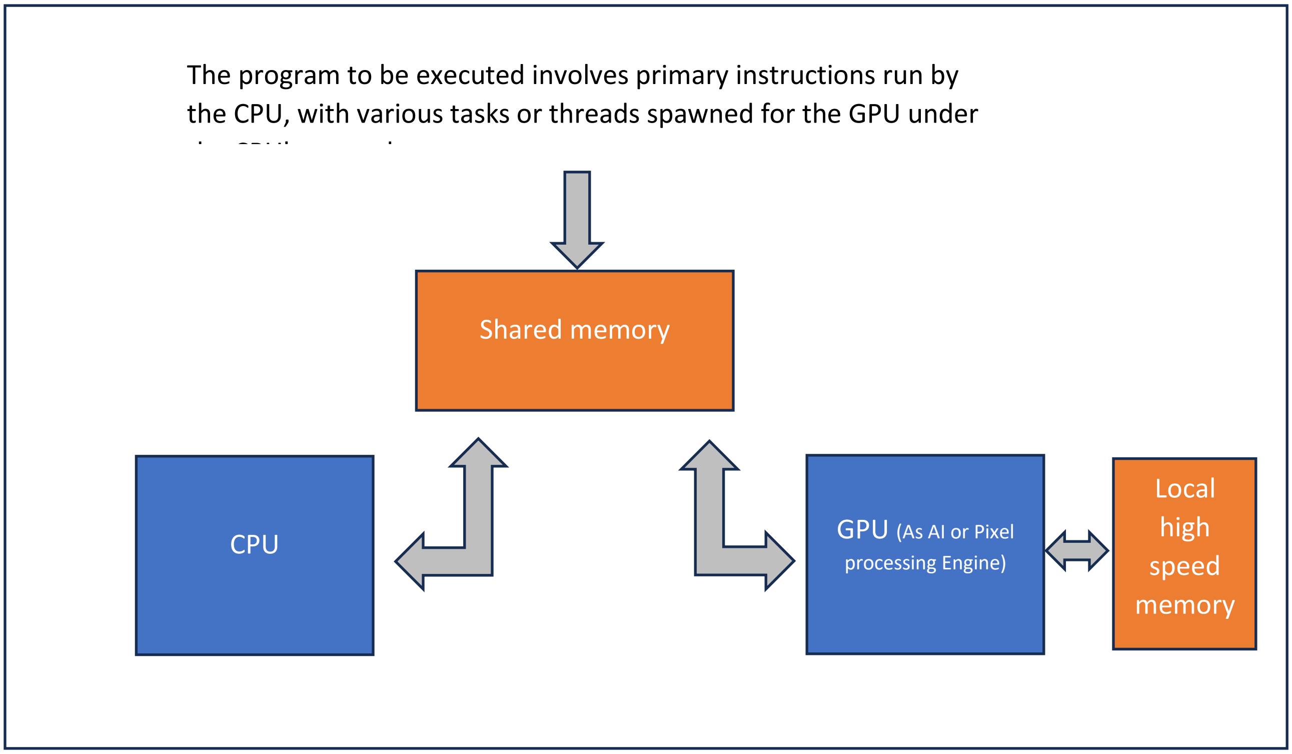
RISC and CISC machines have well-established and relatively straightforward programming models, supported by ecosystems developed over the past four decades. In contrast, GPUs, as parallel vector machines, offer a more complex and less intuitive programming model. Nevertheless, GPUs have cultivated their own community, APIs, and OS support over the years. Besides pixel processing, GPU engines are also used for general-purpose high-performance computing. Leveraging this existing infrastructure for AI programming is generally easier than developing entirely new models from scratch, as required for custom AI processors.


Code/Application are written with established languages /C++/Java/Python/…….
Code /Applications are written utilising library/API for GPU
AI Application is written utilising AI development framework
ADD Ons extensions/libraries for AI
AI libraries/framework Like TensorFlow, Pytorch
AI Extensions Like OpenVino/CUDA-X
GPU development Tool Kit/API libraries (DX/OCL/OGL/Cuda..etc)
GPU development Tool Kit/API libraries (DX/OCL/OGL/Cuda..etc)
Compile to machine binary
Compile to machine binary
Compile to machine binary
CPU (RISC/CISC)
GPU as a pixel processor
GPU for AI processing

Established Programming models of GPU – providing good building block for AI

Data sharing model between CPU and GPU

For many years, GPUs and CPUs have coexisted in platforms, with applications using the CPU as the primary processor to run programs and schedule tasks for the GPU. Data is shared between the CPU and GPU through common memory access. GPUs have two types of memory structures: local memory and shared memory. Shared memory facilitates data transfer between the CPU and GPU. The CPU schedules initial commands and work queues for the GPU, which executes multiple threads in parallel. During processing, the GPU requires fast access to large amounts of transient memory. Finally, results are sent back to the CPU via shared memory.



The program to be executed involves primary instructions run by the CPU, with various tasks or threads spawned for the GPU under

Data Sharing and Hybrid workflow between CPU and GPU

GPU programming models quickly scale up for AI processing due to established workflows, architecture, and ecosystem support. AI workflow models are evolving to leverage both CPUs and GPUs on the same platform, making GPUs a compelling choice for AI processing. As AI applications and development frameworks advance, the generic but well-established GPU ecosystem supports this evolution with lower upfront development costs.