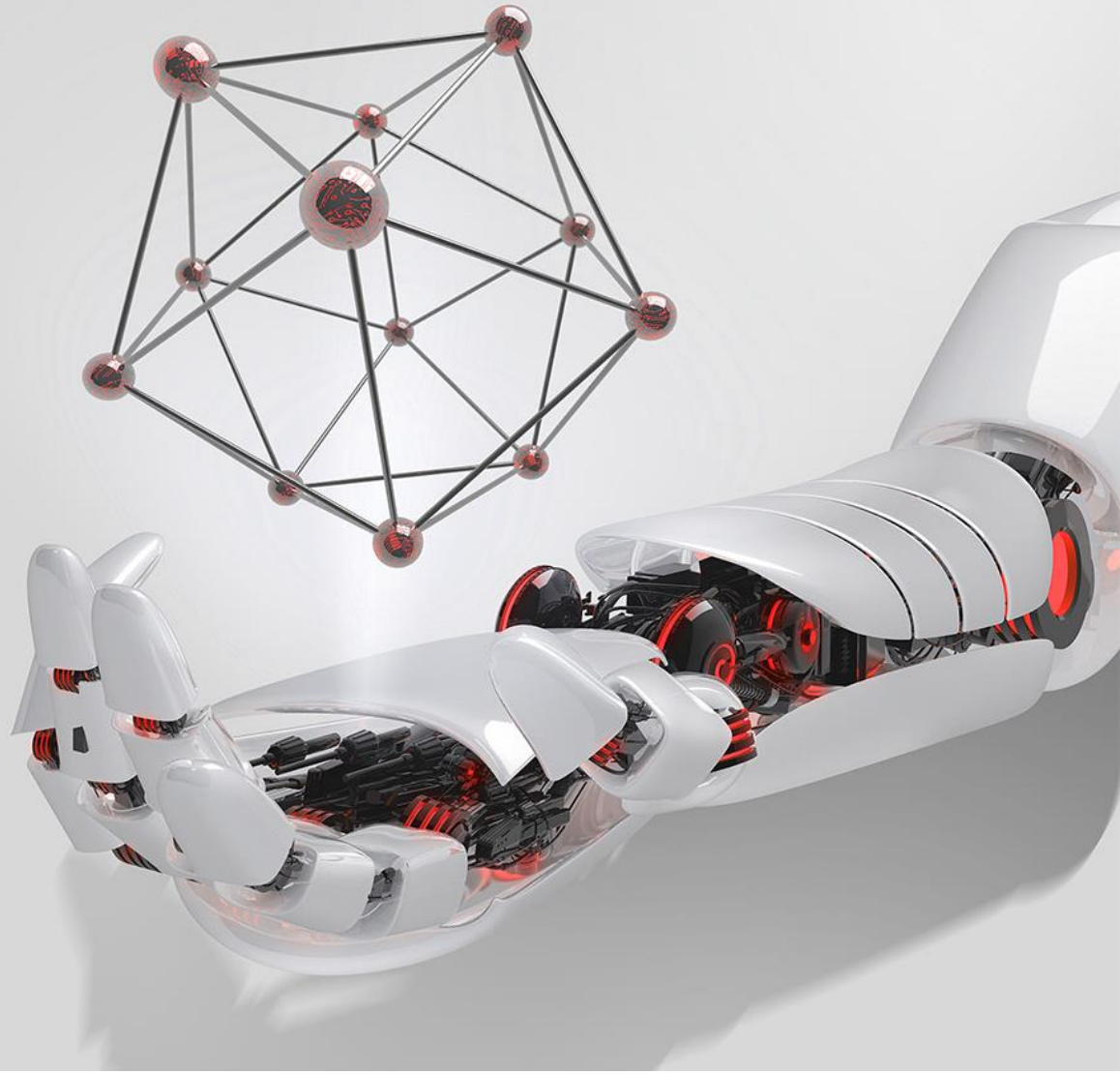
华为昇腾AI处理器及应用初识课程



Chapter 2：昇腾AI处理器

HUAWEI

目录

昇腾AI处理器全场景布局及现状

已发布

MindSpore

AI应用

HiAI Service | General APIs | Advanced APIs | Pre-integrated Solutions

HiAI Engine | ModelArts

应用使能

MindSpore | TensorFlow | PyTorch | Paddlepaddle | …

框架

CANN

(Compute Architecture for Neural Networks)

芯片使能

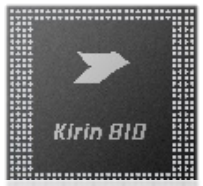
Ascend-Nano | Ascend-Tiny | Ascend-Lite | Ascend | Ascend-Mini | Ascend-Max

IP & Chip

消费终端 | 公有云 | 私有云 | 边缘计算 | 行业终端

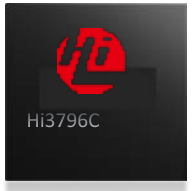
已发布

Ascend910



已商用

(荣耀、NOVA)



即将商用

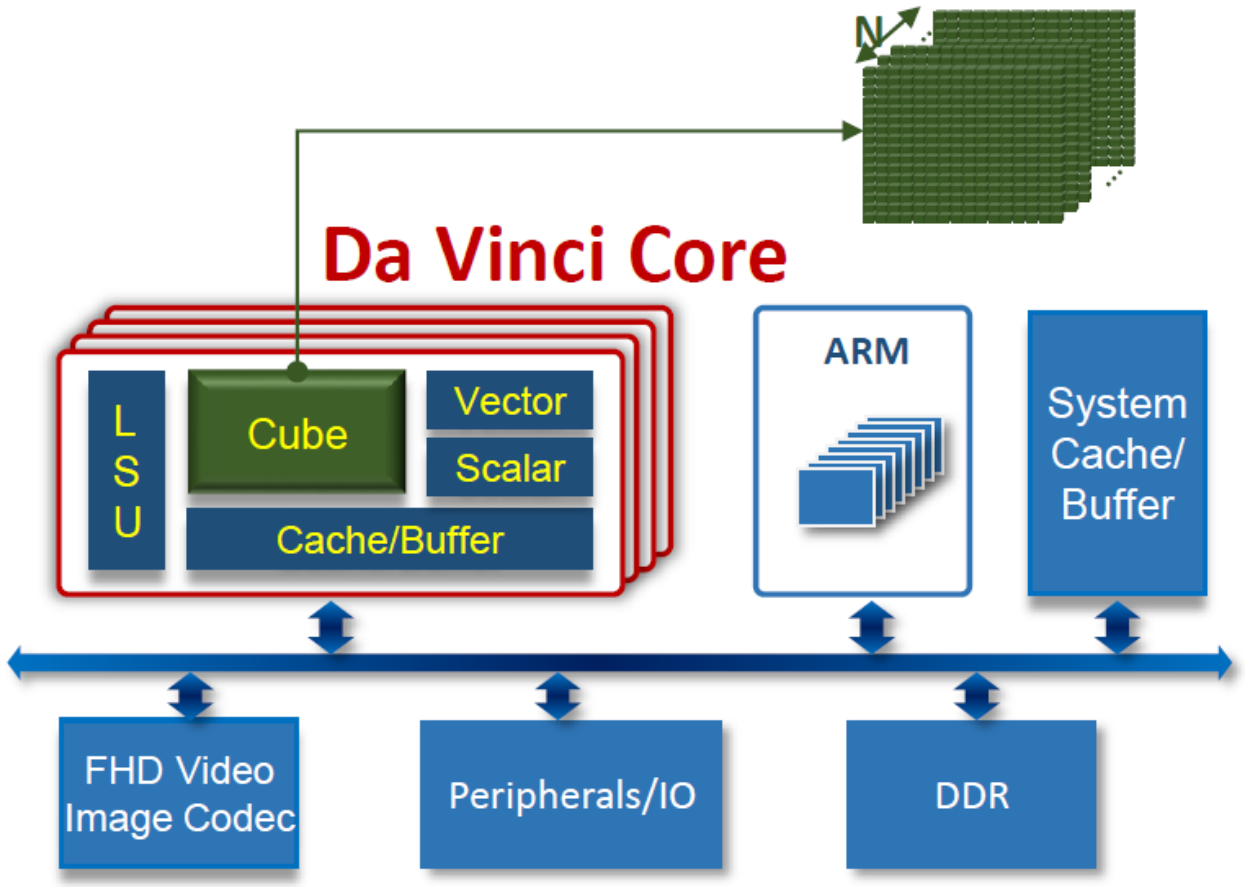
(STB机顶盒，智慧家庭)

已商用

已商用

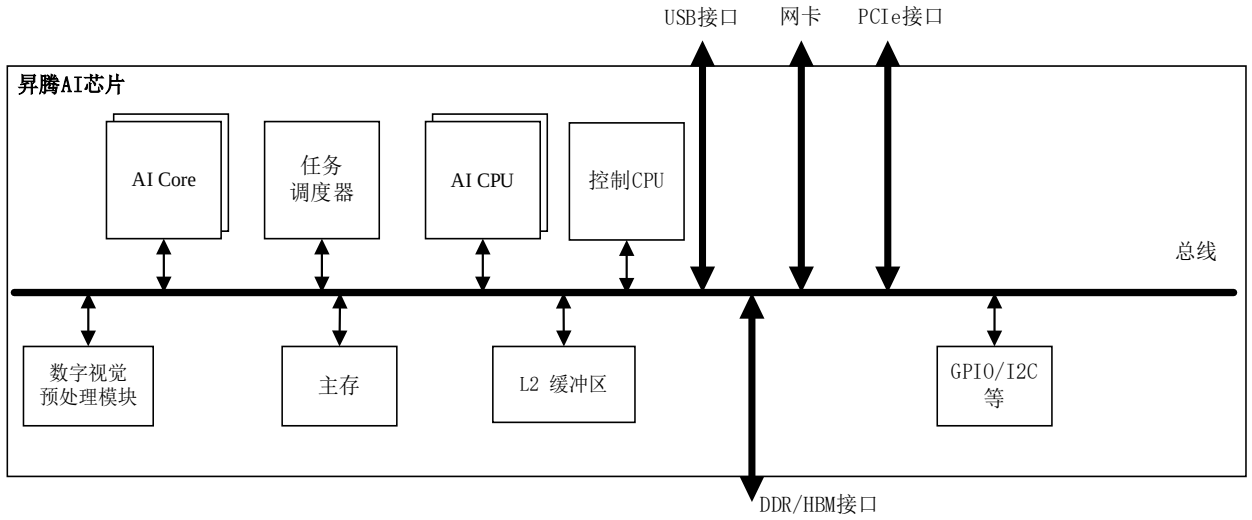
Ascend310 AI处理器规格

SPECIFICATIONS	Description
Architecture	AI co-processor
Performance	Up to 8T @FP16
	Up to 16T@INT8
Codec	16 Channel Decoder – H.264/265 1080P30 1 Channel Encoder
Memory Controller	LPDDR4X
Memory Bandwidth	2*64bit @3733MT/S
System Interface	PCIe3.0 /USB 3.0/GE
Package	15mm*15mm
Max Power	8Tops@4W, 16Tops@8W
Process	12nm FFC



Note: This is typical configuration, high performance and low power sku can be offered based on your requirement.

Ascend310 AI处理器逻辑架构

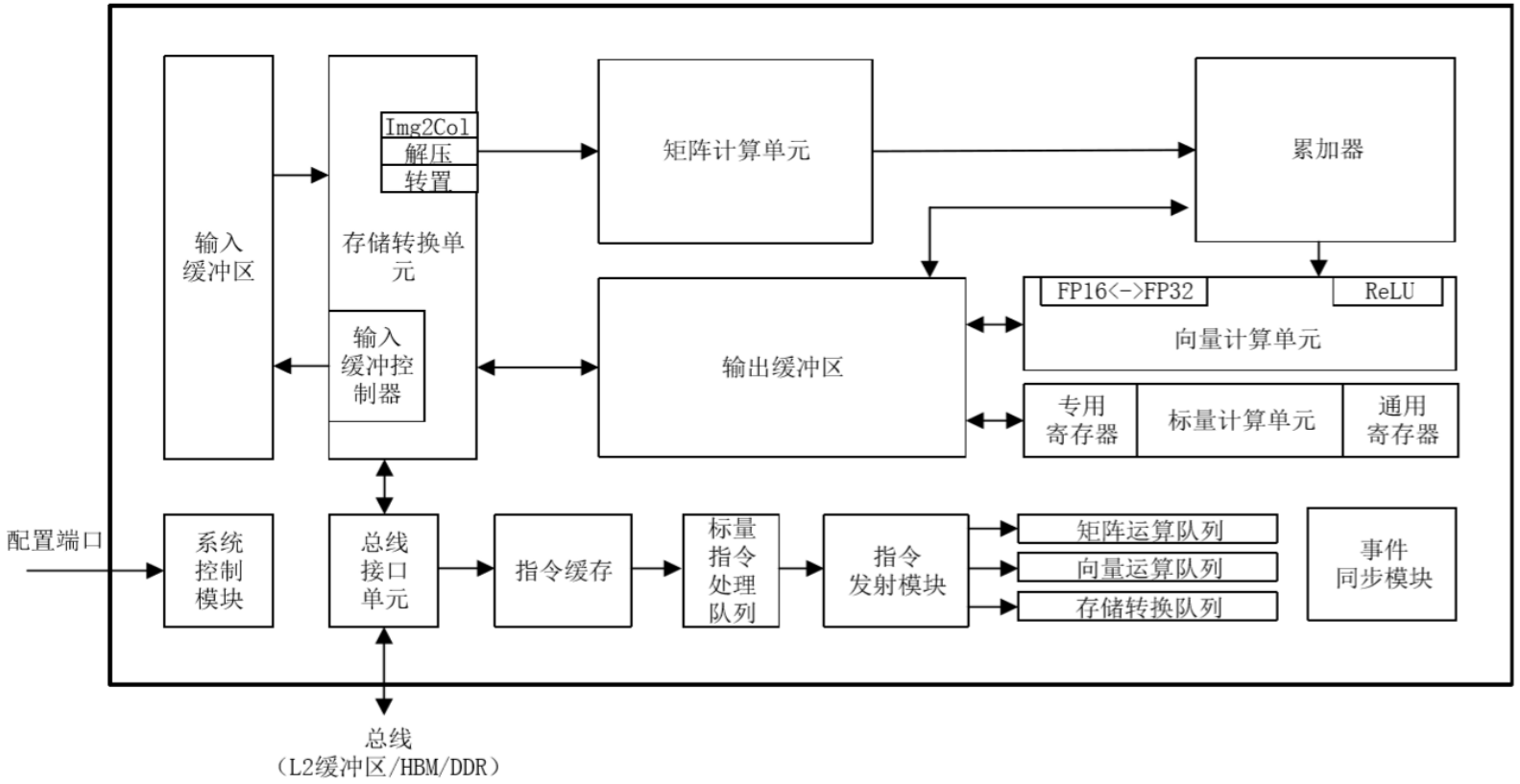


昇腾AI处理器的主要架构组成：

- 芯片系统控制CPU（Control CPU）
- AI计算引擎（包括AI Core和AI CPU）
- 多层级的片上系统缓存（Cache）或缓冲区（Buffer）
- 数字视觉预处理模块（Digital Vision Pre-Processing，DVPP）等

- **AI Core**：集成了2个AI Core。昇腾AI芯片的计算核心，主要负责执行矩阵、向量、标量计算密集的算子任务，采用达芬奇架构。
- **ARM CPU核心**：集成了8个A55。其中一部分部署为**AI CPU**，负责执行不适合跑在AI Core上的算子（承担非矩阵类复杂计算）；一部分部署为专用于控制芯片整体运行的**控制CPU**。两类任务占用的CPU核数可由软件根据系统实际运行情况动态分配。此外，还部署了一个专用CPU作为**任务调度器**（Task Scheduler，TS），以实现计算任务在AI Core上的高效分配和调度；该CPU专门服务于AI Core和AI CPU，不承担任何其他的事务和工作。
- **DVPP**：数字视觉预处理子系统，完成图像视频的编解码。用于将从网络或终端设备获得的视觉数据，进行预处理以实现格式和精度转换等要求，之后提供给AI计算引擎。
- **Cache & Buffer**：SOC片内有层次化的memory结构，AI core内部有两级memory buffer，SOC片上还有8MB L2 buffer，专用于AI Core、AI CPU，提供高带宽、低延迟的memory访问。芯片还集成了LPDDR4x控制器，为芯片提供更大容量的DDR内存。
- **对外接口**：支持PCIE3.0、RGMII、USB3.0等高速接口、以及GPIO、UART、I2C、SPI等低速接口。

达芬奇架构（AI Core）

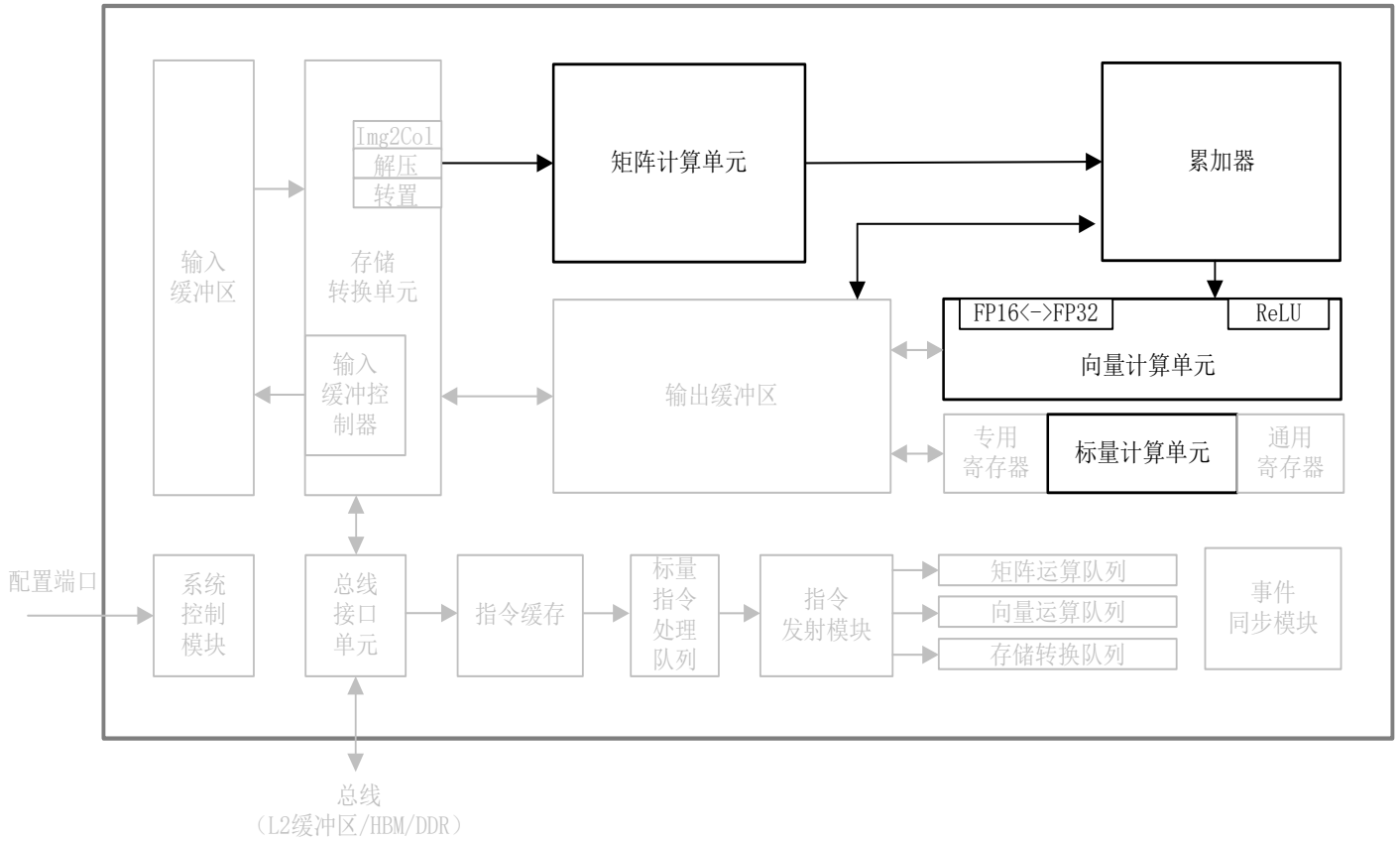


达芬奇架构主要部分：

- **计算单元：** 包含三种基础计算资源（矩阵计算单元、向量计算单元、标量计算单元）
- **存储系统：** AI Core的片上存储单元和相应的数据通路构成了存储系统。
- **控制单元：**整个计算过程提供了指令控制，相当于AI Core的司令部，负责整个AI Core的运行。

达芬奇架构（AI Core）—— 计算单元

三种基础计算资源：Cube Unit、Vector Unit和Scalar Unit，分别对应矩阵、向量和标量三种常见的计算模式。



- **矩阵计算单元（Cube Unit）：**

  矩阵计算单元和累加器主要完成矩阵相关运算。一拍完成一个fp16的 16x16与16x16矩阵乘（4096）； 如果是int8输入，则一拍完成 16*32 与 32*16 矩阵乘（8192）；

- **向量计算单元（Vector Unit）：**

  实现向量和标量，或双向量之间的计算，功能覆盖各种基本的计算类型和许多定制的计算类型，主要包括FP16/FP32/Int32/Int8等数据类型的计算；

- **标量计算单元（Scalar Unit）：**

  相当于一个微型CPU，控制整个AI Core的运行，完成整个程序的循环控制、分支判断，可以为Cube/Vector提供数据地址和相关参数的计算，以及基本的算术运算。

达芬奇架构（AI Core） —— 加速原理

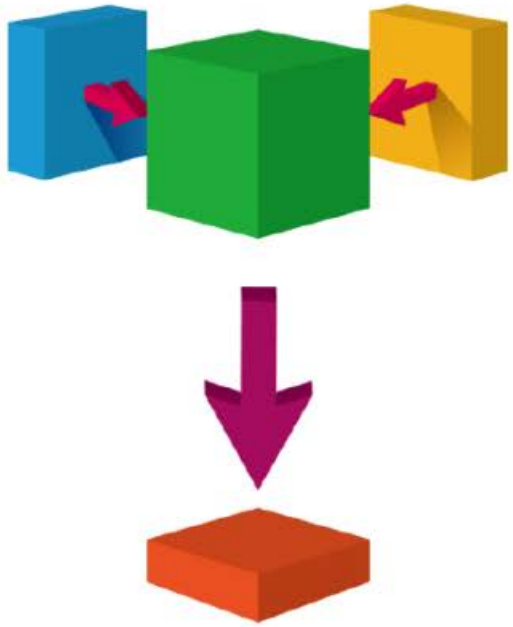
Scalar Unit
Full Flexibility Computation

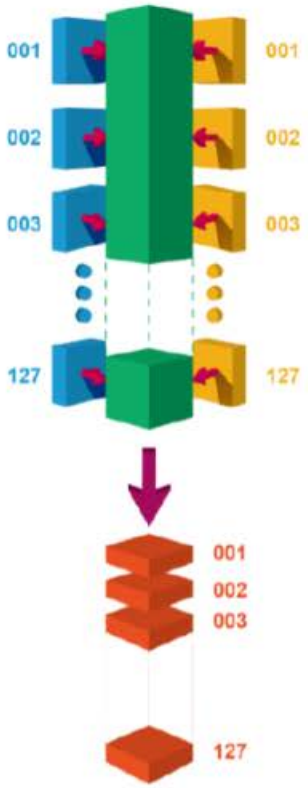
+

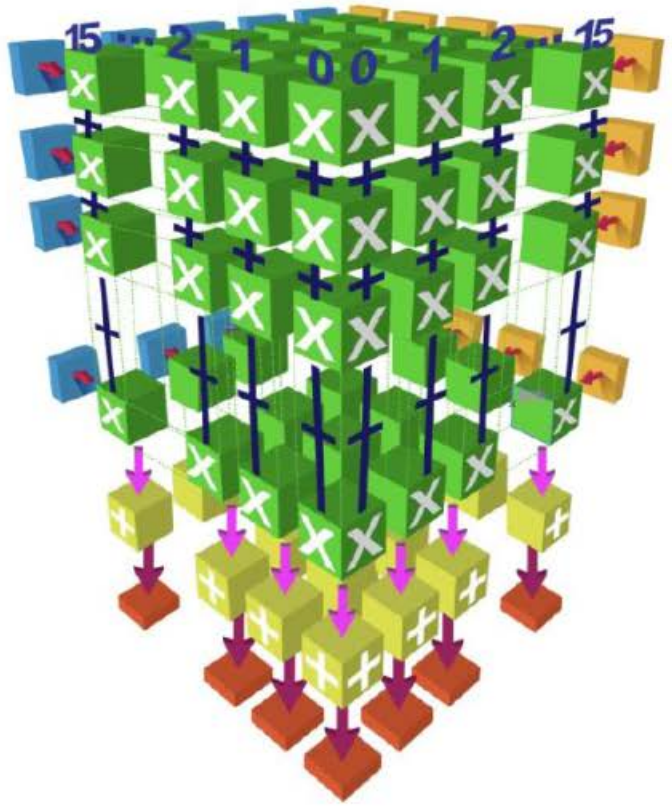
Vector Unit
Rich & Efficient Operations

+

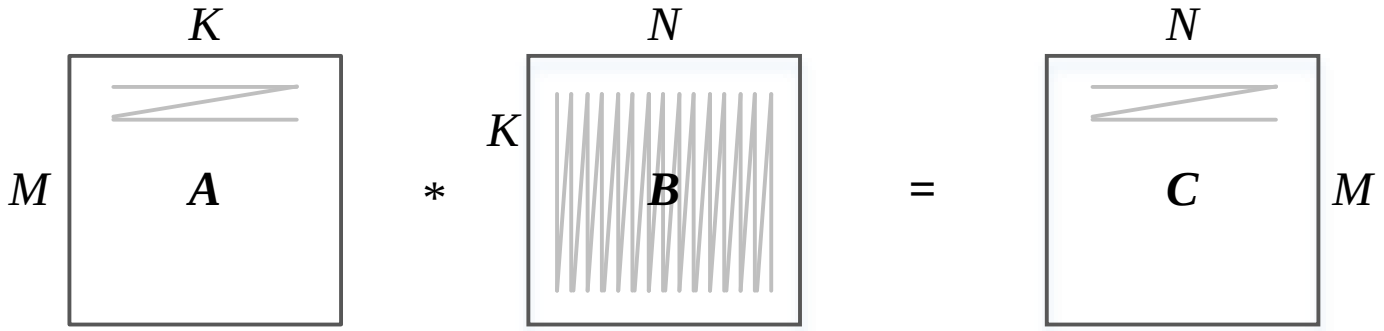
Cube Unit
High Intensity Computation







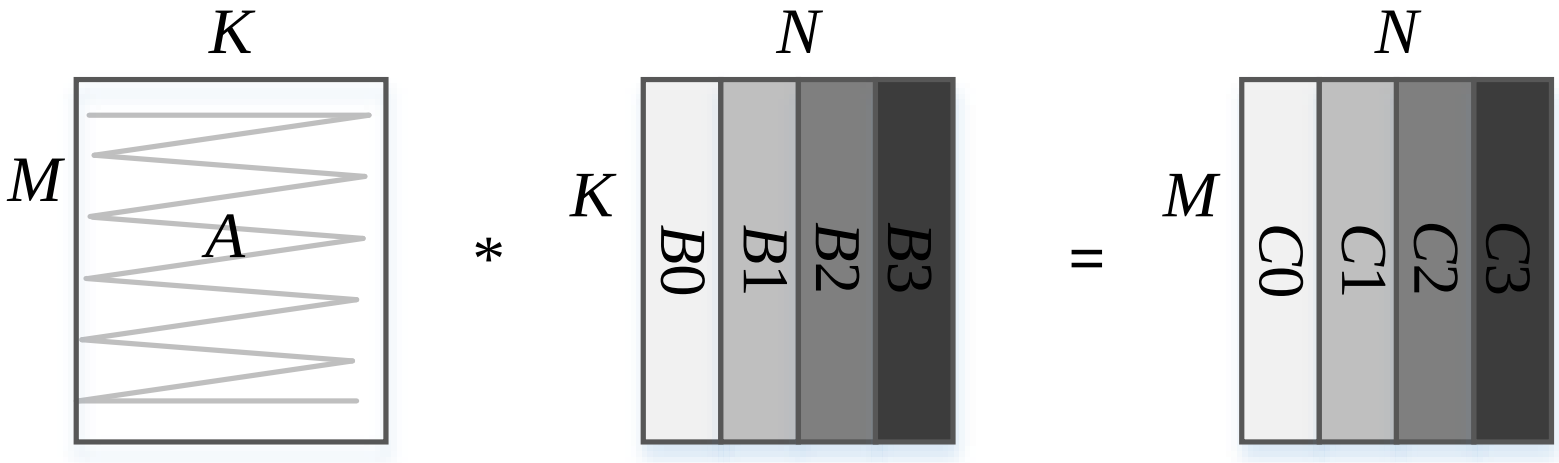
达芬奇架构（AI Core） —— 矩阵计算单元



- 上图表示一个矩阵A和另一个矩阵B之间的乘法运算C=A*B，其中M表示矩阵A的行数，K表示矩阵A的列数以及矩阵B的行数，N表示矩阵B的列数。请同学思考，这个矩阵乘法在CPU如何实现？

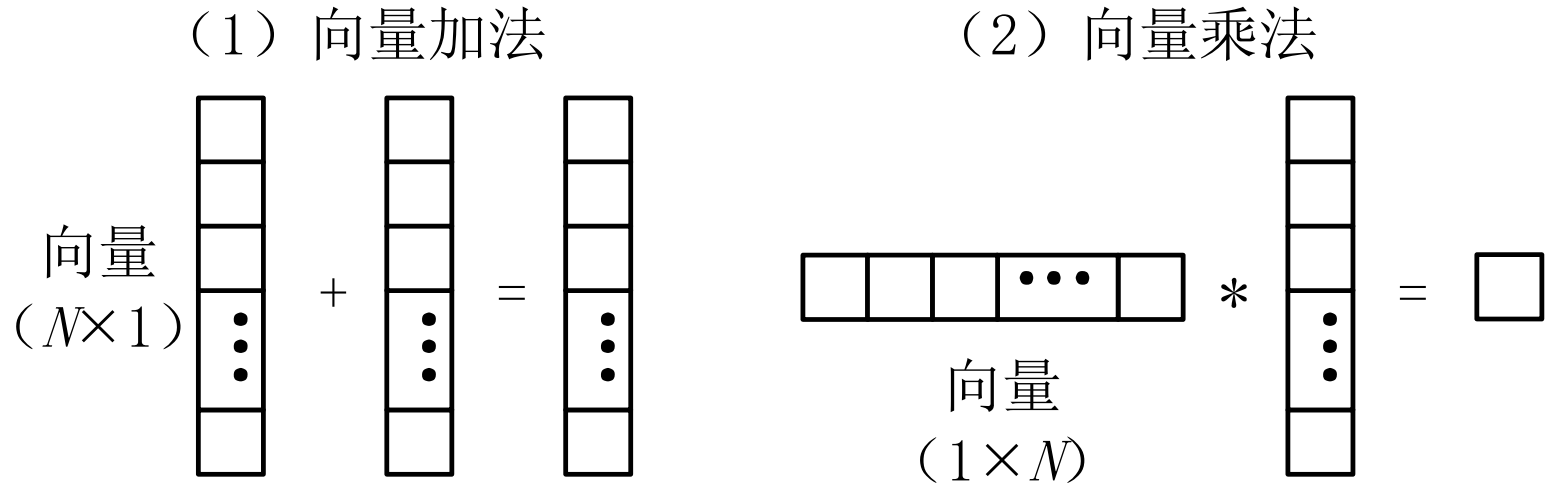
```
for (m=0; m<M, m++)
    for (n=0; n<N, n++)
        for (k=0; k<K, k++)
            C[m][n] += A[m][k]*B[k][n]
```

达芬奇架构（AI Core） —— 矩阵分块计算



- 一般在矩阵较大时，由于芯片上计算和存储资源有限，往往需要对矩阵进行分块平铺处理（Tiling）。受限于片上缓存的容量，当一次难以装下整个矩阵B时，可以将矩阵B划分成为B0、B1、B2和B3等多个子矩阵。而每一个子矩阵的大小都可以适合一次性存储到芯片上的缓存中并与矩阵A进行计算从而得到结果子矩阵。这样做的目的是充分利用数据的局部性原理，尽可能的把缓存中的子矩阵数据重复使用完毕并得到所有相关的子矩阵结果后再读入新的子矩阵开始新的周期。如此往复可以依次将所有的子矩阵都一一搬运到缓存中，并完成整个矩阵计算的全过程，最终得到结果矩阵C。

达芬奇架构（AI Core） —— 向量计算单元



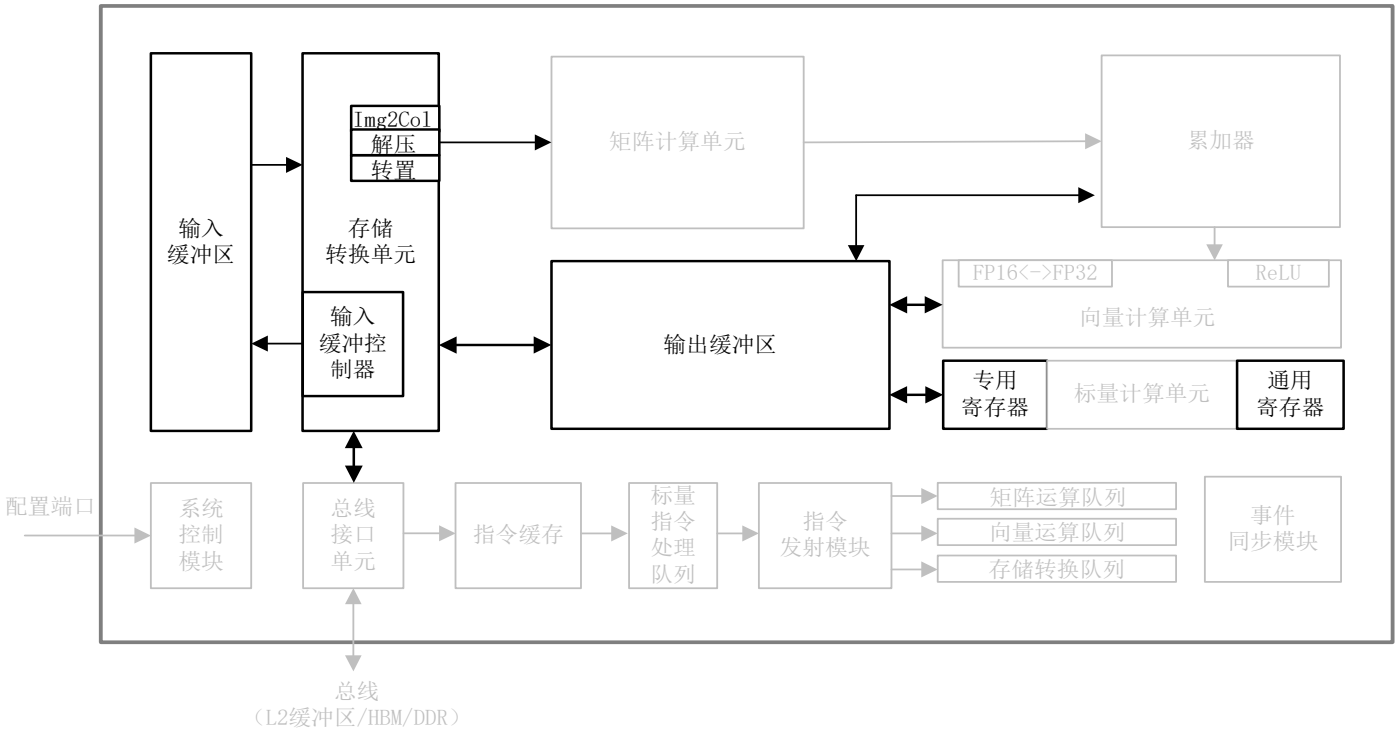
- AI Core中的向量计算单元主要负责完成和向量相关的运算，能够实现向量和标量，或双向量之间的计算，功能覆盖各种基本和多种定制的计算类型，主要包括FP32、FP16、INT32和INT8等数据类型的计算。
- 如上图所示，向量计算单元可以快速完成两个FP16类型的向量相加或者相乘。向量计算单元的源操作数和目的操作数通常都保存在输出缓冲器中。对向量计算单元而言，输入的数据可以不连续，这取决于输入数据的寻址模式。

达芬奇架构（AI Core）—— 标量计算单元

- 标量计算单元负责完成AI Core中与标量相关的运算。它相当于一个微型CPU，控制整个AI Core的运行。标量计算单元可以对程序中的循环进行控制，可以实现分支判断，其结果可以通过在事件同步模块中插入同步符的方式来控制AI Core中其它功能性单元的执行流水。它还为矩阵计算单元或向量计算单元提供数据地址和相关参数的计算，并且能够实现基本的算术运算。其它复杂度较高的标量运算则由专门的AI CPU通过算子完成。
- 在标量计算单元周围配备了多个通用寄存器（General Purpose Register，GPR）和专用寄存器（Special Purpose Register，SPR）。这些通用寄存器可以用于变量或地址的寄存，为算术逻辑运算提供源操作数和存储中间计算结果。专用寄存器的设计是为了支持指令集中一些指令的特殊功能，一般不可以直接访问，只有部分可以通过指令读写。

达芬奇架构（AI Core）—— 存储系统

AI Core采用了大容量的片上缓冲区设计，通过增大的片上缓存数据量来减少数据从片外存储系统搬运到AI Core中的频次，从而可以降低数据搬运过程中所产生的功耗，有效控制了整体计算的能耗。



- **数据通路：是指AI Core在完成一次计算任务时，数据在AI Core中的流通路径。**

  达芬奇架构数据通路的特点是**多进单出**，主要是考虑到神经网络在计算过程中，输入的数据种类繁多并且数量巨大，可以通过并行输入的方式来提高数据流入的效率。与此相反，将多种输入数据处理完成后往往只生成输出特征矩阵，数据种类相对单一，单输出的数据通路，可以节约芯片硬件资源。

存储单元由存储控制单元、缓冲区和寄存器组成。

- **存储控制单元：**

  通过总线接口直接访问AI Core之外的更低层级的缓存，也可以直通到DDR或HBM直接访问内存。其中还设置了**存储转换单元**，作为AI Core内部数据通路的传输控制器，负责AI Core内部数据在不同缓冲区之间的读写管理，以及完成一系列的格式转换操作，如补零，Img2Col，转置、解压缩等；

- **输入缓冲区：**

  用来暂时保留需要频繁重复使用的数据，不需要每次都通过总线接口到AI Core的外部读取，从而在减少总线上数据访问频次的同时也降低了总线上产生拥堵的风险，达到节省功耗、提高性能的效果；

- **输出缓冲区：**

  用来存放神经网络中每层计算的中间结果，从而在进入下一层计算是方便的获取数据。相比较通过总线读取数据的带宽低，延迟大，通过输出缓冲区可以大大提升计算效率；

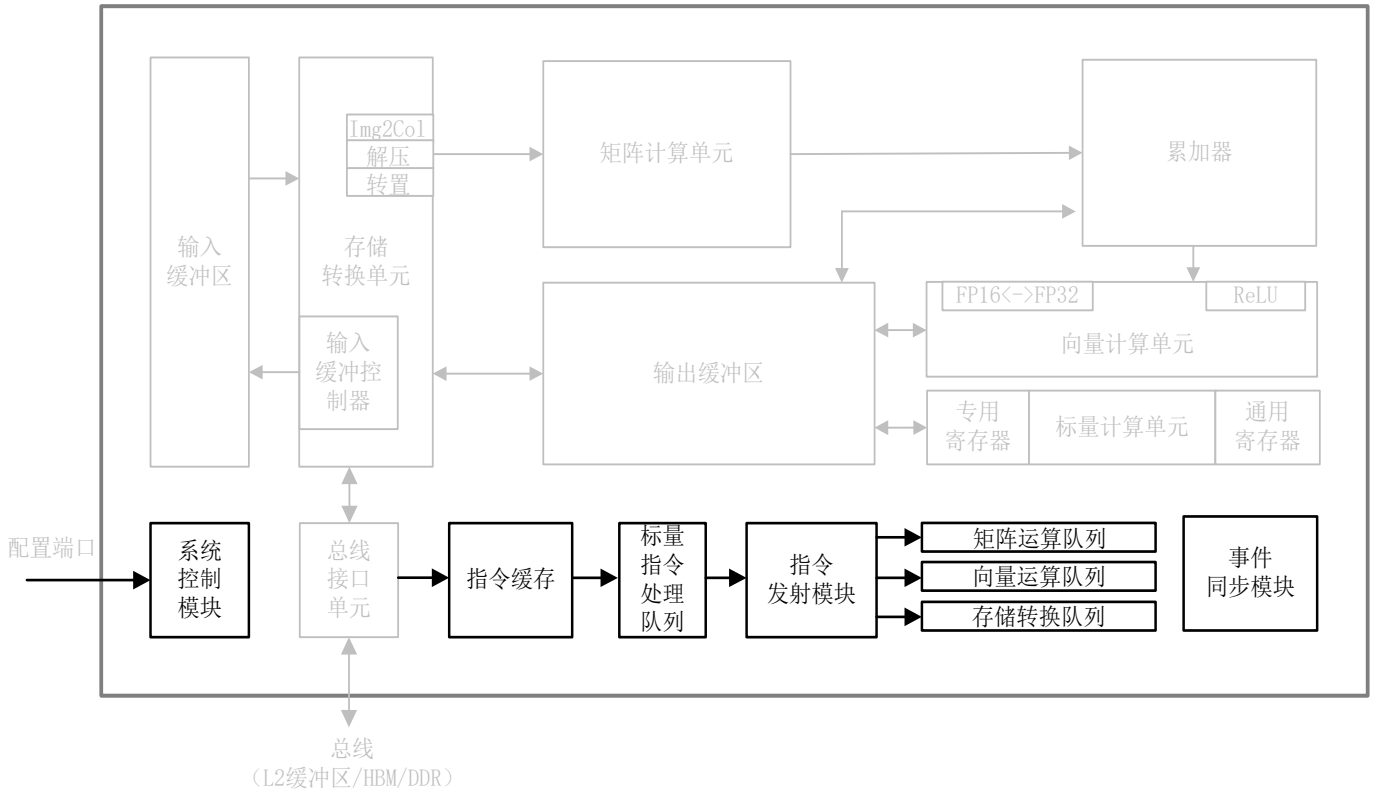
- **寄存器：**

  AI Core中的各类寄存器资源主要是标量计算单元在使用。

存储单元和相应的数据通路，构成了AI Core的存储系统。

达芬奇架构（AI Core）—— 控制单元

控制单元主要组成部分为系统控制模块、指令缓存、标量指令处理队列、指令发射模块、矩阵运算队列、向量运算队列、存储转换队列和事件同步模块。

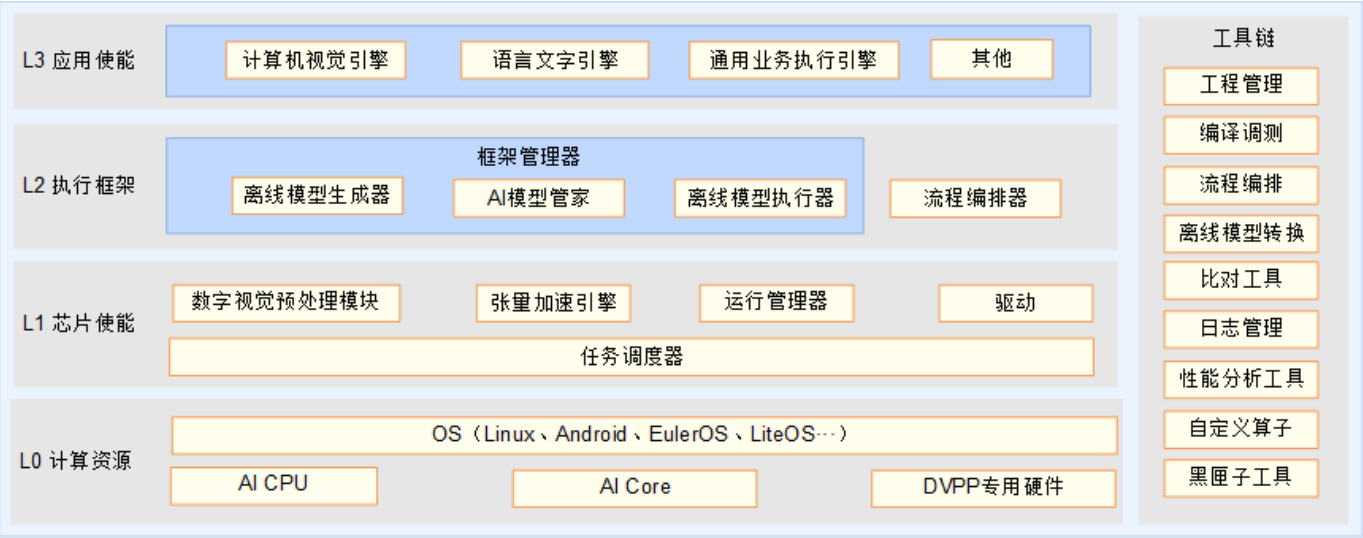


- **系统控制模块：**
  控制任务块（AI Core最小任计算务粒度）的执行进程，在任务块执行完成后，系统控制模块会进行中断处理和状态申报。如果执行过程出错，会把执行的错误状态报告给任务调度器；
- **指令缓存：**
  在指令执行过程中，可以提前预取后续指令，并一次读入多条指令进入缓存，提升指令执行效率；
- **标量指令处理队列：**
  指令被解码后便会被导入标量队列中，实现地址解码与运算控制，这些指令包括矩阵计算指令、向量计算指令以及存储转换指令等；
- **指令发射模块：**
  读取标量指令队列中配置好的指令地址和参数解码，然后根据指令类型分别发送到对应的指令执行队列中，而标量指令会驻留在标量指令处理队列中进行后续执行；
- **指令执行队列：**
  指令执行队列由矩阵运算队列、向量运算队列和存储转换队列组成，不同的指令进入相应的运算队列，队列中的指令按进入顺序执行；
- **事件同步模块：**
  时刻控制每条指令流水线的执行状态，并分析不同流水线的依赖关系，从而解决指令流水线之间的数据依赖和同步的问题。

目录

昇腾AI处理器软件栈逻辑架构 （1）

昇腾AI芯片的软件栈主要分为4个层次和一个辅助工具链。4个层次分别为L3应用使能层、L2执行框架层、L1芯片使能层和L0计算资源层。工具链主要提供了程序开发、编译调测、应用程序流程编排、日志管理和性能分析等辅助能力。这些主要组成部分在软件栈中功能和作用相互依赖，承载着数据流、计算流和控制流。



L3应用使能层：是应用级封装，面向特定的应用领域，提供不同的处理算法。为各种领域提供具有计算和处理能力的引擎可以直接使用下一层L2执行框架提供的框架调度能力，通过通用框架来生成相应的神经网络而实现具体的引擎功能。

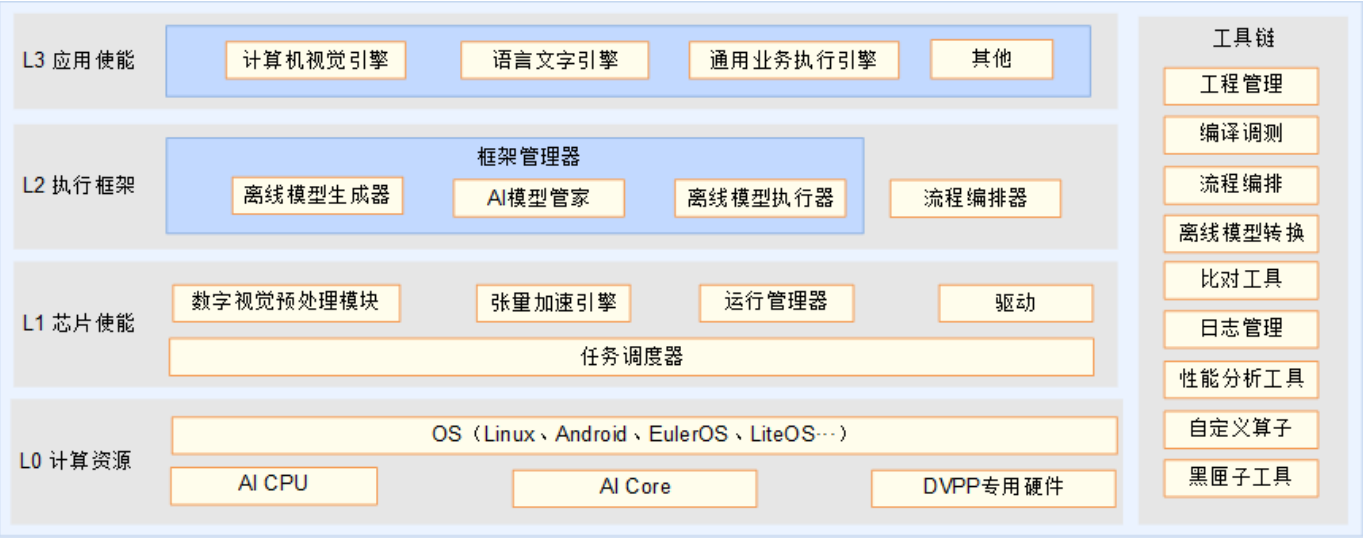
- 通用引擎：提供通用的神经网络推理能力；
- 计算机视觉引擎：提供视频或图像处理的算法封装；
- 语言文字引擎：提供语音、文本等数据的基础处理算法封装；

L2执行框架层：是框架调用能力和离线模型生成能力的封装。L3层将应用算法开发完并封装成引擎后，L2层会根据相关算法的特点进行合适深度学习框架的调用(如Caffe或TensorFlow)，来得到相应功能的神经网络，再通过框架管理器生成离线模型。L2层将神经网络的原始模型转化成可在昇腾AI芯片上运行的离线模型后，离线模型执行器将离线模型传送给L1芯片使能层进行任务分配。

- 在线框架：使用主流深度学习开源框架（Caffe/TensorFlow等），通过离线模型转换和加载，使其能在昇腾AI芯片上进行加速运算。
- 离线框架：提供神经网络的离线生成和执行能力，可以在脱离深度学习框架下使得离线模型(Offline Model，OM)具有同样的能力（主要是推理能力）。
- 框架管理器：包含离线模型生成器(OMG)、离线模型执行器(OME)和离线模型推理接口，支持模型的生成、加载、卸载和推理计算执行。
- OMG：负责将Caffe或TensorFlow框架下已经生成的模型文件和权重文件转换成离线模型文件，并可以在昇腾AI芯片上独立执行。
- OME：负责加载和卸载离线模型，并将加载成功的模型文件转换为可执行在昇腾AI芯片上的指令序列，完成执行前的程序编译工作。
- 流程编排器：向开发者提供用于深度学习计算的开发平台，包含计算资源、运行框架以及相关配套工具等，负责对模型的生成、加载和运算的调度。

昇腾AI处理器软件栈逻辑架构 （2）

昇腾AI芯片的软件栈主要分为4个层次和一个辅助工具链。4个层次分别为L3应用使能层、L2执行框架层、L1芯片使能层和L0计算资源层。工具链主要提供了程序开发、编译调测、应用程序流程编排、日志管理和性能分析等辅助能力。这些主要组成部分在软件栈中功能和作用相互依赖，承载着数据流、计算流和控制流。



L1芯片使能层：是离线模型通向昇腾AI芯片的桥梁。针对不同的计算任务，L1层通过加速库(Library)给离线模型计算提供加速功能。L1层是最接近底层计算资源的一层，负责给硬件输出算子层面的任务。

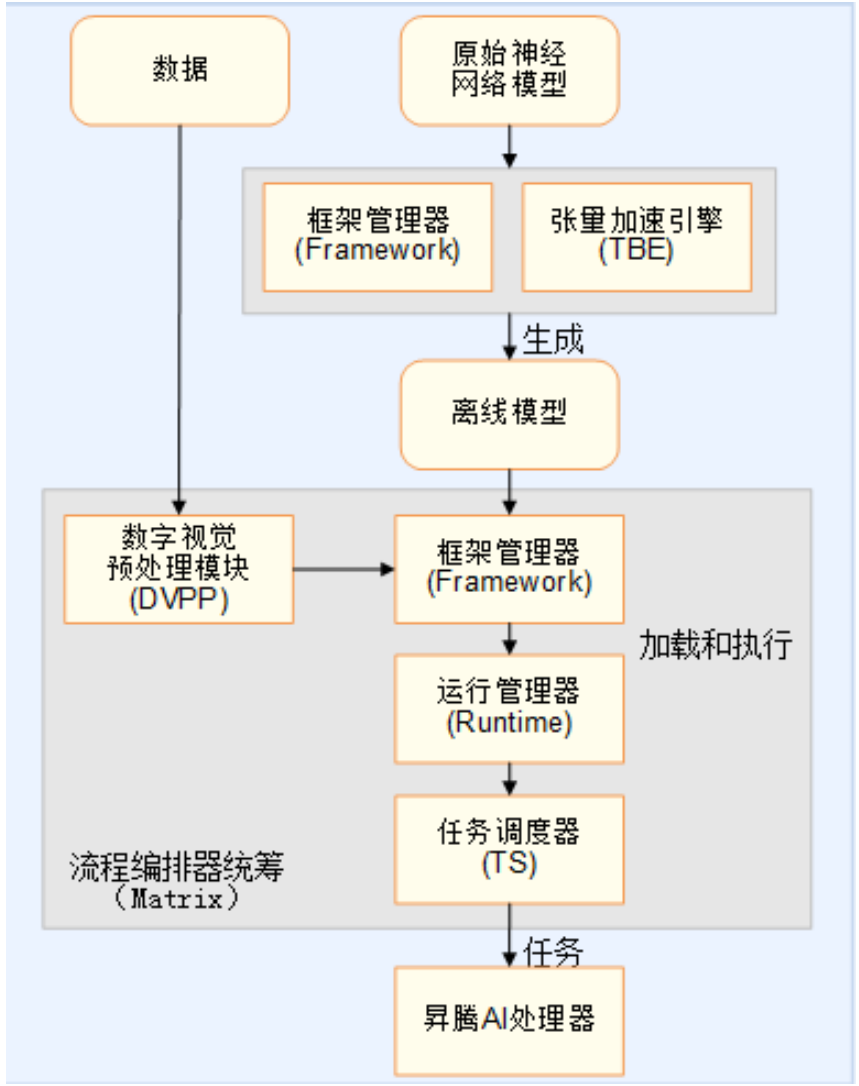
- 张量加速引擎：支持在线和离线模型的加速计算，包含标准算子加速库和和自定义算子的能力。为L2层提供具有功能完备性的算子。
- 运行管理器：负责与L2层通信，提供标准算子加速库的接口给L2层调用，让具体网络模型能找到优化后的、可执行的、可加速的算子进行功能上的最优实现。
- 任务调度器：根据具体任务类型处理和分发相应的计算核函数到AI CPU或者AI Core上，通过驱动激活硬件执行。
- 数字视觉预处理模块：是一个面向图像视频领域的多功能封装体，为上层提供使用底层专用硬件的各种数据（图像或视频）预处理能力。

L0计算资源层：是昇腾AI芯片的硬件算力基础，提供计算资源，执行具体的计算任务。

- AI Core：算力核心，负责大算力的计算任务，主要完成神经网络的矩阵相关计算。
- AI CPU：负责较为复杂的计算和执行控制功能，完成控制算子、标量和向量等通用计算。
- DVPP专用硬件：负责输入数据（如图像和视频数据）的预处理操作，在特定场景下为AI Core提供满足计算需求的数据格式。
- 操作系统：作用是使上述三者紧密辅助，组成一个完善的硬件系统，为昇腾AI芯片的深度神经网络计算提供执行上的保障。

工具链：工具链是一套支持昇腾AI处理器，并可以方便程序员进行开发的工具平台，提供了自定义算子的开发、调试和网络移植、优化及分析功能的支撑。另外在面向程序员的编程界面提供了一套桌面化的编程服务，极大的降低了深度神经网络相关应用程序的开发门槛。

昇腾AI处理器神经网络软件流

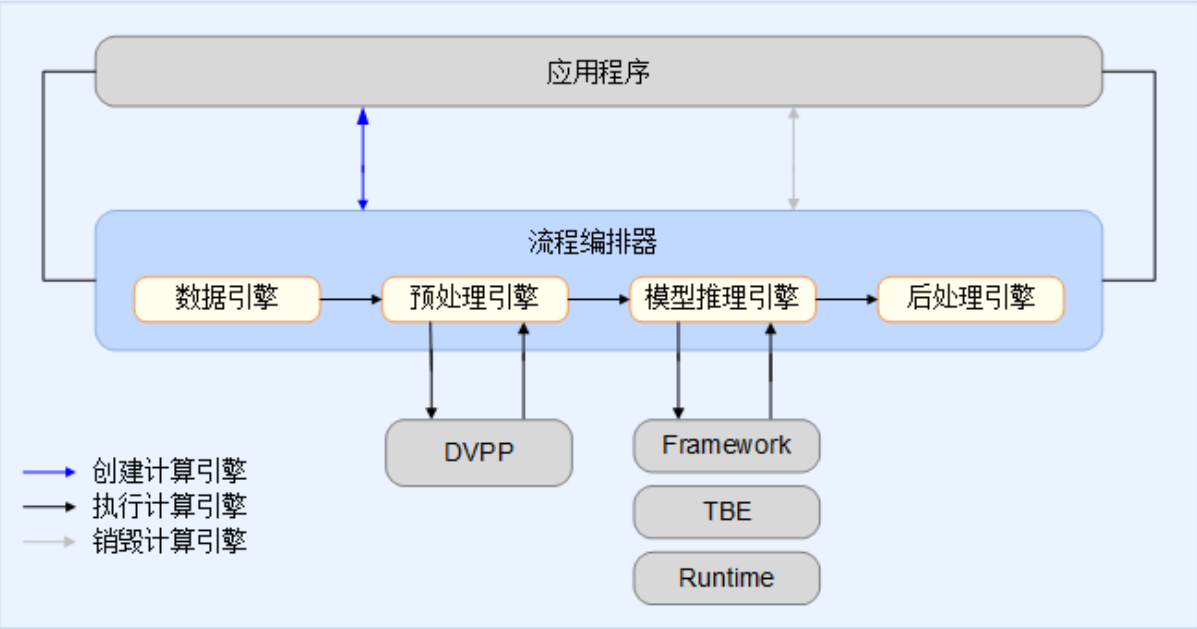


功能齐全且支撑神经网络高性能计算的软件流，是深度学习框架到昇腾AI芯片之间的一座桥梁，完成一个神经网络应用的实现和执行。

这条神经网络软件流围绕离线模型的生成、加载和执行，形成了一个完整的功能集群。聚集了如下功能块：

- **流程编排器**：负责完成神经网络在昇腾AI芯片上的落地与实现，统筹了整个神经网络生效的过程，控制离线模型的加载和执行过程；
- **数字视觉预处理模块**：在输入之前进行一次数据处理和修饰，来满足计算的格式需求；
- **张量加速引擎**：作为神经网络算子兵工厂，为神经网络模型源源不断提供功能强大的计算算子；
- **框架管理器**：专门将原始神经网络模型打造成昇腾AI芯片支持的形态，并且将塑造后的模型与昇腾AI芯片相融合，引导神经网络运行并高效发挥出性能；
- **运行管理器**：为神经网络的任务下发和分配提供了各种资源管理通道；
- **任务调度器**：作为一个硬件执行的任务驱动者，为昇腾AI芯片提供具体的目标任务；运行管理器和任务调度器联合互动，共同组成了神经网络任务流通向硬件资源的大坝系统，实时监控和有效分发不同类型的执行任务。

神经网络软件流—— 流程编排器（Matrix ）



基于开发者板（Atlas 200 DK）的场景

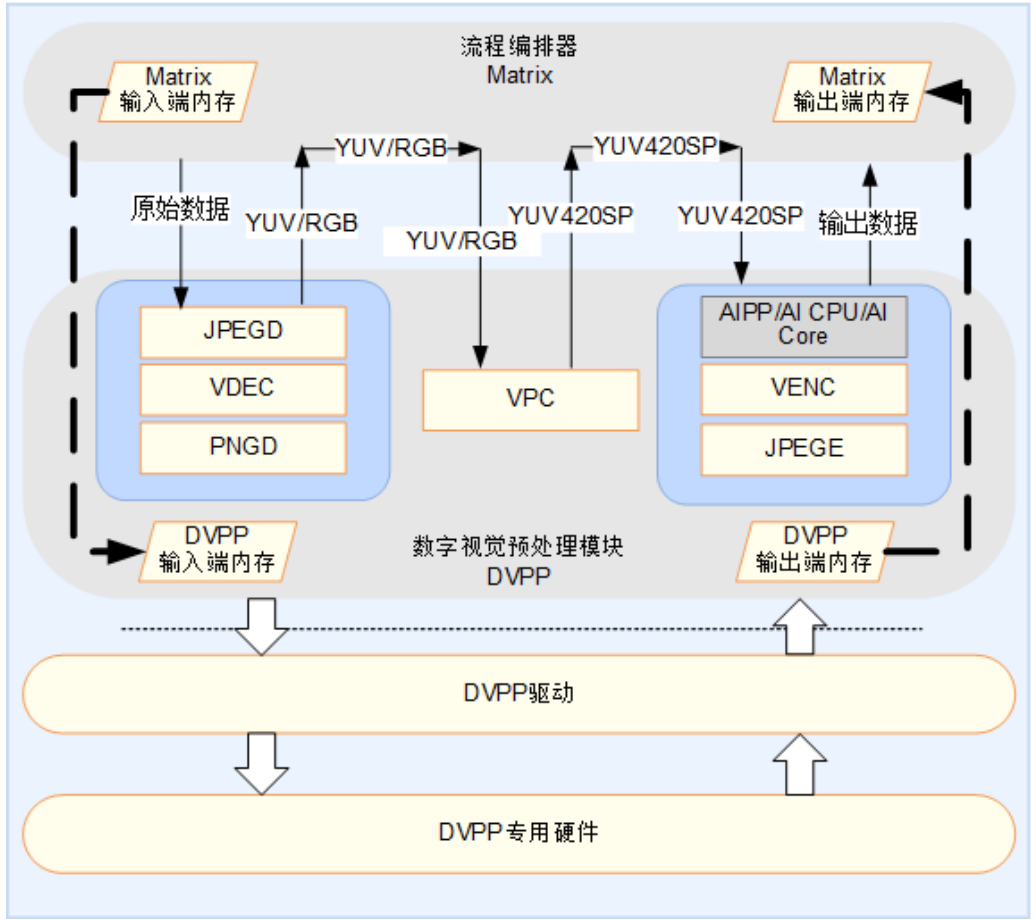
整个深度神经网络应用通常包括四个引擎：

- **数据引擎**：主要准备神经网络需要的数据集（如MNIST数据集）和进行相应数据的处理（如图片过滤等），作为后续计算引擎的数据来源。
- **预处理引擎**：主要进行媒体数据的预处理，完成图像和视频编解码以及格式转换等操作，并且数字视觉预处理各功能模块都需要统一通过流程编排器进行调用。
- **模型推理引擎**：利用加载好的模型和输入的数据流完成神经网络的前向计算。
- **后处理引擎**：对模型推理引擎输出的数据进行后续处理，如图像识别的加框和加标识等处理操作。

三个主要步骤：

- **首先，**应用程序调用流程编排器，根据网络模型由流程编排器创建计算引擎流程图，并初始化计算引擎。在初始化过程中，模型推理引擎通过AI模型管家的初始化接口加载模型，完成计算引擎流程图的创建。再将数据灌入到数据引擎中，如果媒体数据格式不满足要求，则预处理引擎会完成预处理。
- **接着，**模型推理引擎调用离线模型执行器中的AI模型管家进行推理计算。模型推理引擎计算结束后，调用流程编排器提供的数据输出接口将推理结果返回给后处理引擎，后处理引擎通过回调函数将推理结果返回给应用程序，完成计算引擎流程图的执行。
- **最后，**在程序计算完成后，流程编排器将计算引擎流程图进行销毁，释放资源，从而完成了开发者板场景的神经网络功能的全部实现。

神经网络软件流—— 数字视觉预处理（DVPP）



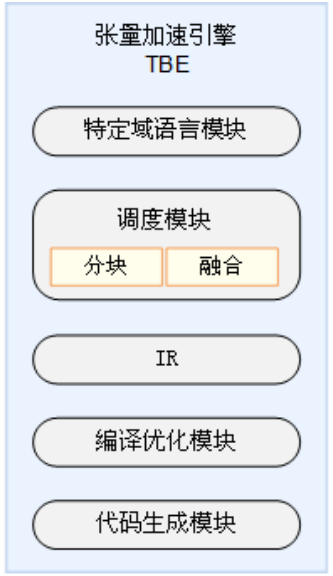
DVPP功能架构

当输入数据进入数据引擎时，引擎一旦检查发现数据格式不满足后续AI Core的处理需求，则可开启数字视觉预处理模块进行数据预处理。

- **数字视觉预处理模块执行流程：**
  - **流程编排器**，位于框架最上层，负责调度DVPP中的功能模块进行相应处理以及管理数据流。
  - **数字视觉处理模块**，位于功能架构的中上层，为流程编排器提供调用视频图形处理模块的编程接口，通过这些接口可以配置编解码或视觉预处理模块的相关参数。
  - **DVPP驱动**，位于功能架构的中下层，主要负责设备管理、引擎管理和引擎模组的驱动。驱动会根据数字视觉处理模块下发的任务，分配对应的DVPP硬件引擎，同时还对硬件模块中的寄存器进行读写，完成其它一些硬件初始化工作。
  - **DVPP模块组**，处于最底层的真实硬件计算资源，是一个独立于昇腾AI芯片中其它模块的一个单独的专用加速器，专门负责执行与图像和视频想对应的编解码和预处理任务。

- **DVPP对外提供以下6个接口：**
  - 视频解码（VDEC）：提供H.264/H.265的视频解码功能
  - 视频编码（VENC）：提供图像编码输出视频编码功能
  - JPEG图片解码（JPEGD）：提供JPEG图片的解码功能
  - JPEG图片编码（JPEGE）：提供JPEG图片的编码功能
  - PNG图片解码（PNGD）：提供PNG图片的解码功能
  - 视觉预处理单元（VPC）：提供图像的格式转换、缩放、裁剪功能

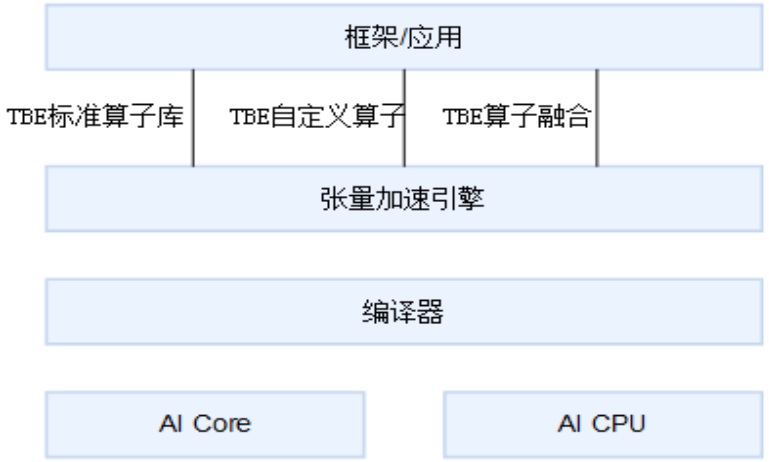
神经网络软件流—— 张量加速引擎（TBE）



TBE模块构成

张量加速引擎提供了基于TVM开发自定义算子的能力，通过TBE语言和自定义算子编程开发界面可以完成相应神经网络算子的开发。包含如下模块：

- **特性语言（Domain-Specific Language，DSL）模块**
- **调度（Schedule）模块**
- **中间表示（Intermediate Representation，IR）模块**
- **编译器传递（Pass）模块**
- **代码生成（CodeGen）模块**



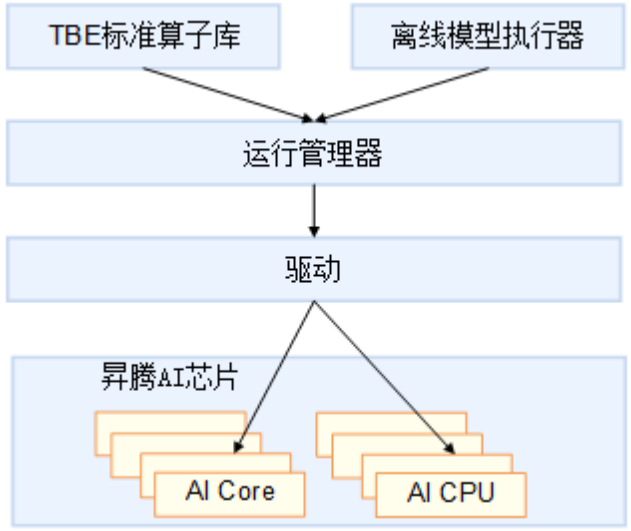
TBE算子应用场景

在提供算子开发能力的同时，也提供了标准算子调用以及算子融合优化的能力，使得昇腾AI芯片在实际的神经网络应用中，可以满足功能多样化的需求，构建网络的方法也会更加方便灵活。

- **TBE标准算子：**
  张量加速引擎提供了一套完整的TBE算子加速库，库中的算子功能与神经网络中的常见标准算子保持了一一对应关系，并且由软件栈提供了编程接口供调用算子使用，为上层深度学习中各种框架或者应用提供了加速的同时尽量避免了开发昇腾AI芯片底层的适配代码。
- **TBE自定义算子：**
  如果在神经网络模型构造中，出现了新的算子，这时张量加速引擎中提供的标准算子库就可能无法满足开发需求。此时可以通过TBE语言进行自定义算子开发，这种开发方式和GPU上利用CUDA C++的方式相似，可以实现更多功能的算子，灵活编写各种网络模型。编写完成的算子会交给编译器进行编译，最终执行在AI Core或AI CPU上发挥出芯片的加速能力。
- **TBE算子融合：**
  在合适的场景下，张量加速引擎提供的算子融合能力会促进算子性能的提升，让神经网络算子可以基于不同层级的缓冲区进行多级别的缓存融合，使得昇腾AI芯片在执行融合后的算子时片上资源利用率获得显著提升。

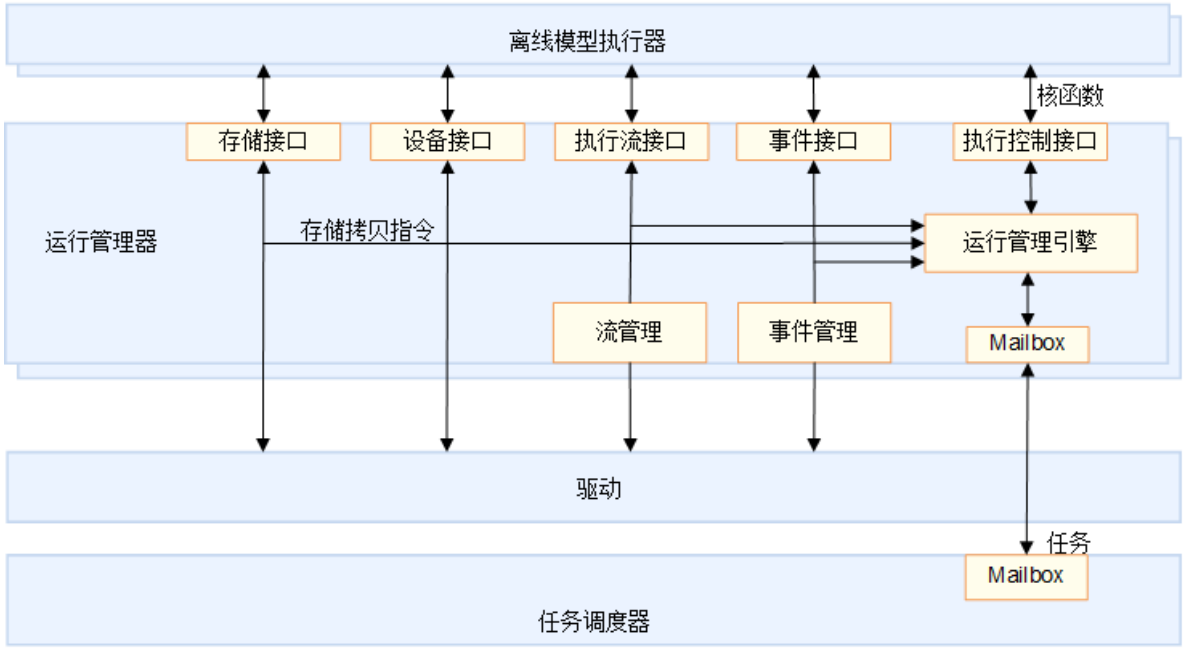
神经网络软件流—— 运行管理器（Runtime）

运行管理器为神经网络的任务分配提供了资源管理通道。昇腾AI处理器通过运行管理器而执行在应用程序的进程空间中，为应用程序提供了存储（Memory）管理、设备（Device）管理、执行流（Stream）管理、事件（Event）管理、核函数（Kernel）执行等功能。



运行管理器在软件栈中的上下文关系

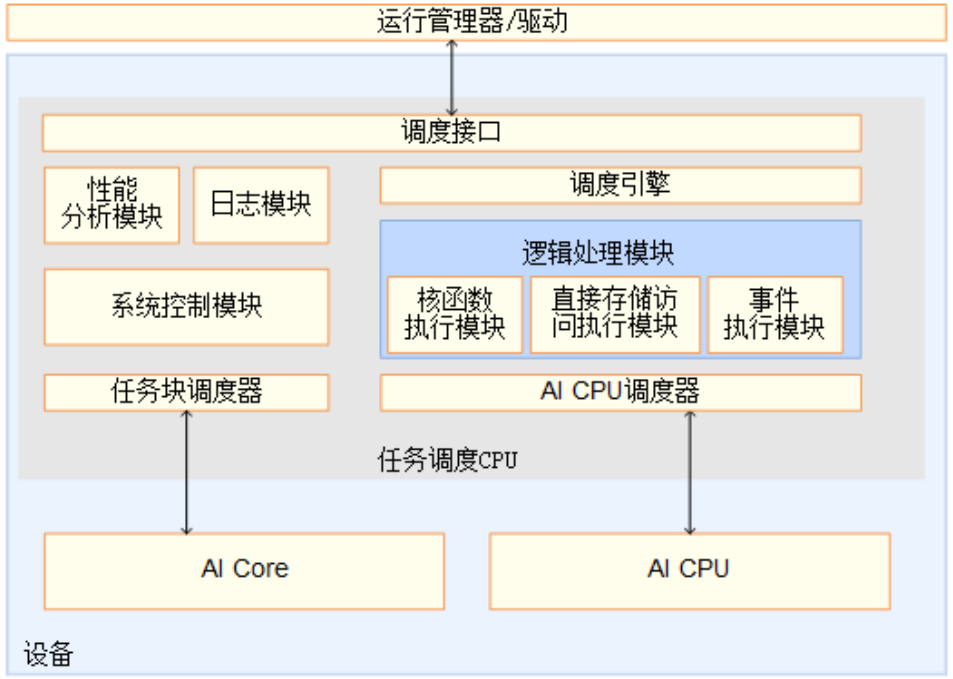
- **TBE标准算子库**：提供神经网络中需要用到的算子；
- **离线模型执行器**：进行离线模型的加载和执行；
- **驱动**：与昇腾AI处理器进行底层交互；



运行管理器的功能模块和对外接口

- **存储接口**：提供设备上HBM（High Bandwidth Memory，高带宽存储器）或DDR（Double Data Rate，双倍速率内存）内存的申请、释放和复制等，包括设备到主机、主机到设备以及设备到设备之间的数据拷贝。
- **设备接口**：提供底层设备的数量和属性查询，以及选中、复位等操作。
- **执行流接口**：提供执行流的创建、释放、优先级定义、回调函数设置、对事件的依赖定义和同步等。
- **事件接口**：在多个执行流之间需要进行同步时，进行同步事件的创建、释放、记录和依赖定义等，确保多个执行流得以同步执行完成并输出模型最终结果。
- **执行控制接口**：Mailbox完成核函数的加载和存储异步拷贝等任务的派发。

神经网络软件流—— 任务调度器（Task Scheduler ）



任务调度器功能框架图

任务调度器模块构成：

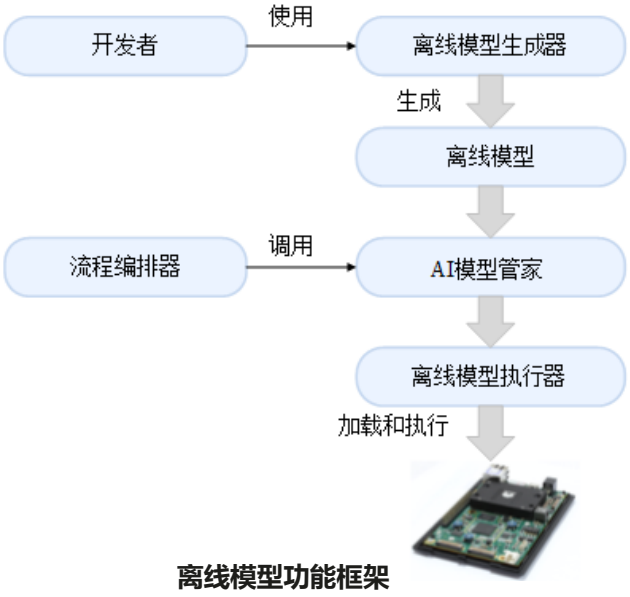
- 调度接口（Interface）：实现与运行管理器与驱动之间的通信和交互
- 调度引擎（Engine）：任务调度实现的主体，实现任务的组织，依赖及调度控制主流程
- 调度逻辑处理模块：实现AICPU, AI Core的任务调度处理逻辑
- AI CPU调度器：实现 AI CPU的状态管理及任务调度
- 任务块调度器：实现AI Core任务的分发和返回
- 系统控制（SysCtrl）模块：负责设置系统配置，芯片功能初始化等功能
- 性能分析（Profile）：实现性能分析任务
- 日志（Log）：实现日志控制

任务调度器运行在设备侧的任务调度CPU上，负责将运行管理器分发的具体任务进一步派发到AI CPU上。它也可以通过硬件任务块调度器（Block Scheduler，BS）把任务分配到到AI CORE上执行，并在执行完成后返回任务执行的结果给运行管理器。

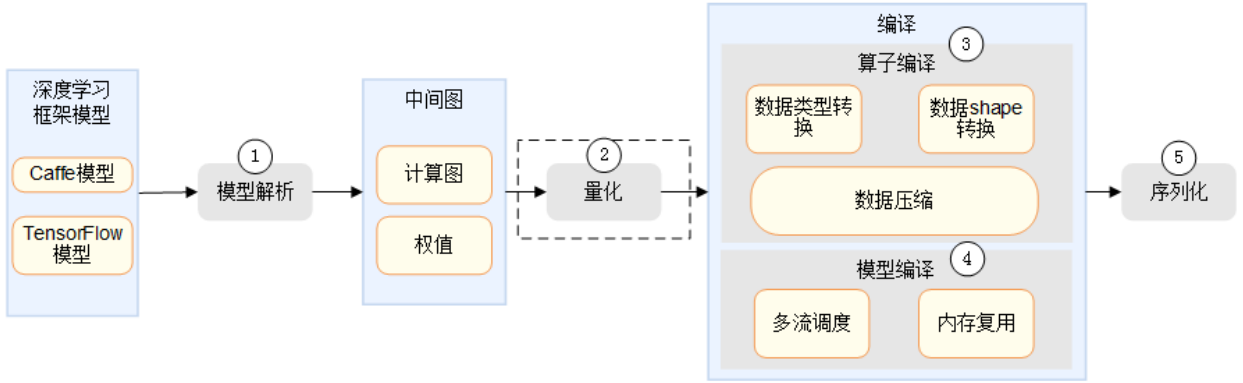
任务调度器处理的主要任务：

- **AI Core任务**：AI Core计算任务；
- **AI CPU任务**：AI CPU计算任务；
- **内存拷贝任务**：执行内存异步拷贝；
- **事件记录任务**：记录事件发生信息，如果存在等待该事件的任务，则这些任务在事件记录完成后可以解除等待，继续执行，消除由事件记录而导致执行流的阻塞；
- **事件等待任务**：如果等待的事件已经发生，则此等待任务直接完成；否则，把此任务加入待处理列表，同时暂停事件等待任务所在执行流中后续所有任务的处理，在等待的事件发生时，再进行事件等待任务的处理；
- **清理维护（Maintenance）任务**：根据任务参数不同进行相应的清理工作，回收计算资源；
- **性能分析（Profiling）任务**：控制性能分析操作的启停，以对计算的性能进行记录和分析；

神经网络软件流—— 框架管理器



离线模型功能框架



框架管理器包含三个部分：

- **离线模型生成器（Offline Model Generator，OMG）**
- **离线模型执行器（Offline Model Executor，OME）**
- **AI模型管家（AI Model Manager）**

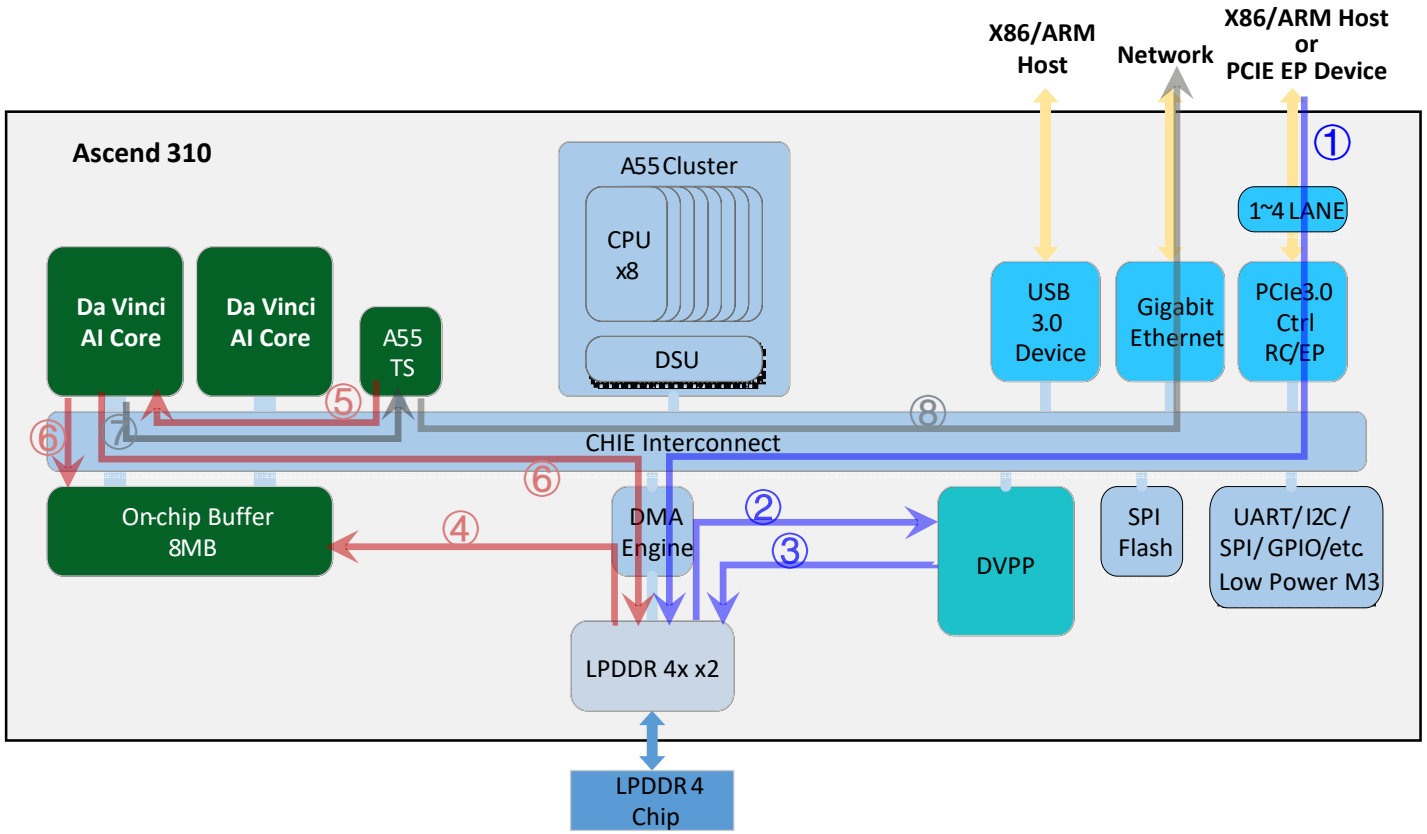
开发者使用离线模型生成器来生成离线模型，以om为后缀的文件进行保存。随后，软件栈中的流程编排器调用框架管理器中AI模型管家，启动离线模型执行器，将离线模型加载到昇腾AI芯片上，最后再通过整个软件栈完成离线模型的执行。从离线模型的诞生，到加载进入昇腾AI芯片硬件，直至最后的功能运行，离线框架管理器始终发挥着管理的作用。

离线模型生成流程：

- **解析**
  - 在解析过程中，离线模型生成器支持不同框架下的原始网络模型解析，提炼出原始模型的网络结构、权重参数，再通过图的表示法，由统一的中间图（IR Graph）来重新定义网络结构。
- **量化**
  - 解析完成后生成了中间图，如果模型还需要进行量化处理，则可以基于中间图的结构和权重，通过自动量化工具来进行量化。
- **算子编译**
- **模型编译**
  - 在完成模型量化后，需要对模型进行编译，编译分为算子编译和模型编译两个部分，算子编译提供算子的具体实现，生成算子特定的离线结构；模型编译将算子模型聚合连接生成离线模型结构。
- **序列化**
  - 编译后产生的离线模型存放于内存中，还需要进行序列化。序列化过程中主要提供签名及加密功能给模型文件，对离线模型进行进一步封装和完整性保护。序列化完成后可以将离线模型从内存输出到外部文件中以供异地的昇腾AI芯片调用和执行。

目录

昇腾AI处理器数据流程图 —— 以人脸识别推理应用为例



1、Camera数据采集和处理

① 从摄像头传入压缩视频流，通过PCIE存储值DDR内存中。

② DVPP将压缩视频流读入缓存。

③ DVPP经过预处理，将解压缩的帧写入DDR内存。

2、对数据进行推理

④ 任务调度器（TS）向直接存储访问引擎（DMA）发送指令，将AI资源从DDR预加载到片上缓冲区。

⑤ 任务调度器（TS） 配置AI Core以执行任务。

⑥ AI Core工作时，它将读取特征图和权重并将结果写入DDR或片上缓冲区。

3、人脸识别结果输出

⑦ AI Core完成处理后，发送信号给任务调度器（TS），任务调度器检查结果，如果需要会分配另一个任务，并返回步骤④ 。

⑧ 当最后一个AI任务完成，任务调度器（TS）会将结果报告给Host。

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and organization for a fully connected, intelligent world.

HUAWEI