QCon
全球软件开发大会

PyTorch与国产芯片的爱恨情仇

演讲人：ZOMI 酱

华为 / 昇腾生态技术专家

PyTorch + 国产芯片

昇腾AI训练解决方案

MindX昇腾应用使能

MindX SDK　MindSpeed加速套件 **MindIE推理引擎**

检索聚类 | 推荐搜索 | .....　分布式并行加速　推理运行时 | 推理服务化

MindSpore

PyTorch

昇腾计算架构 CANN

运行时 | 毕昇编译器 | Ascend C | 算子加速库 | GE图引擎

MindStudio

全流程工具链

算子开发工具

调试调优工具

...

模组　MDC　推理卡　小站　推理服务器　训练卡　训练服务器　集群

昇腾AI大模型训练解决方案

文本生成 | 视图生成 | 视图分析 | 广告推荐 | 自动驾驶 | 智能语音 | AI Agent | 科学智能

AI训练平台

AI框架

[M]昇思 MindSpore　...　**PyTorch**

云平台（HCS/HCSO/三方平台）

计算子系统

AI芯片使能CANN

网络子系统

iMaster NCE

存储子系统

DME　DPC

高效液冷 + 高功率供电

一体化机房

CCAE 统一运维管理平台

工具链

目 录

01

PyTorch 框架发展

我没有副标题哦

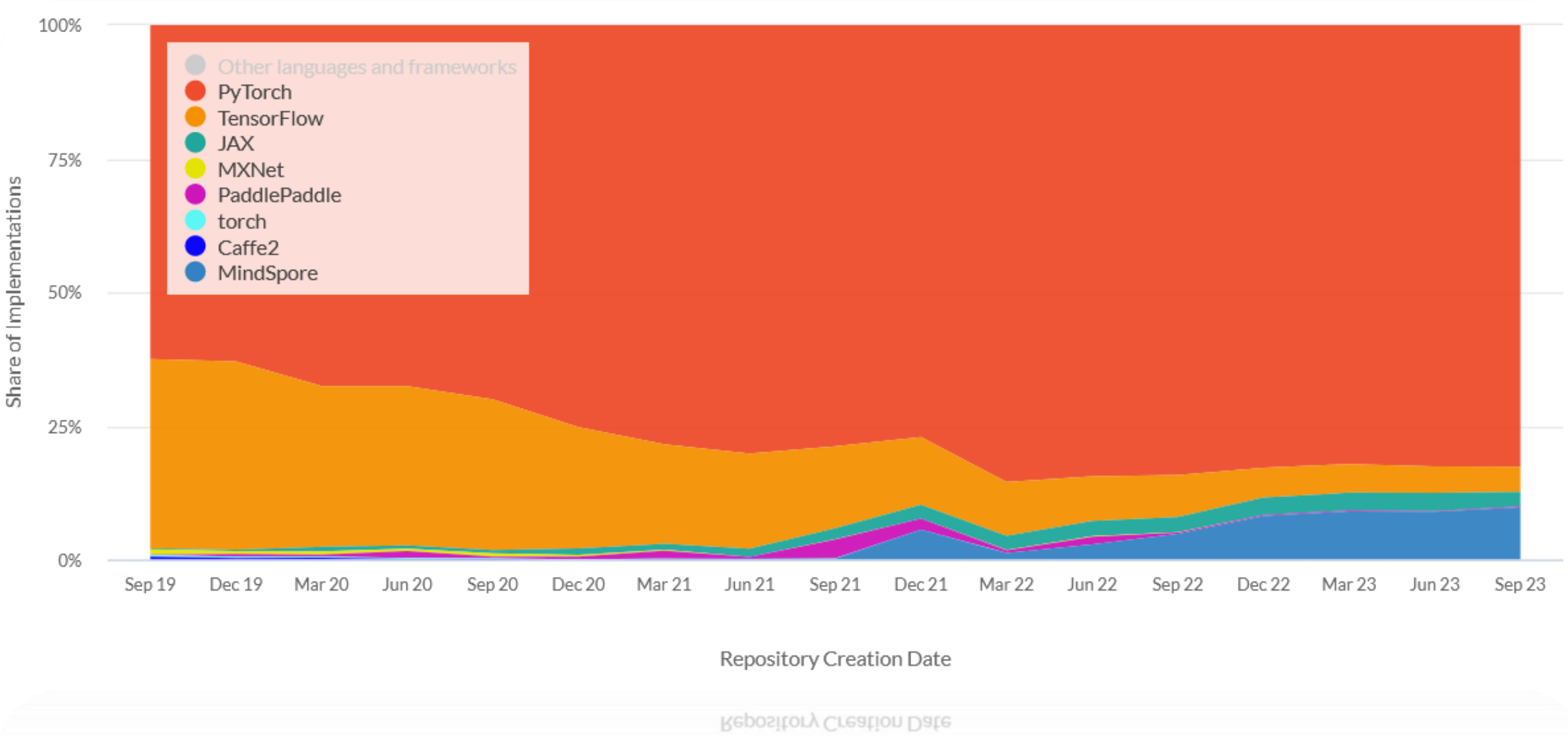
有没有开发者没用过 PyTorch?

可能，你不是开发者

为什么你们会用 PyTorch?

- 别人用，我也用
- 参考样例和模型代码多
- 论文和模型都是基于 PyTorch
- PyTorch 真的很好用，我哭死 (╥﹏╥)
- PyTorch 易用性很强
- 基于 PyTorch 的三方库很多
- 网上能找到 PyTorch 问题的资料很多

Paper With Code Trends



PyTorch 框架的发展历程

2002.10	**Torch**	Torch 是一个Facebook的开源机器学习库、一个科学计算框架和一种基于 Lua 编程语言的脚本语言。
2017.01	**PyTorch 0.1**	它标志着该项目的正式诞生。 这个版本为神经网络的构建提供了基本的构建块，例如 Tensor（类似于numpy数组，可自动求导）和 autograd（用于自动计算梯度）
2018.4	**PyTorch 0.4**	PyTorch 0.4 提出动态计算图（Dynamic Computational Graph），版本凭借其易用性、灵活性和高效性等特点，成为了深度学习研究和应用的首选框架之一。
2018.5	**PyTorch 1.0**	PyTorch 1.0引入了更为丰富的特性，包括动态计算图、更强大的GPU加速支持、改进的模型库以及更好的文档。
2019.10	**PyTorch 1.3**	PyTorch 1.3 版本的核心特性包括对移动设备的支持、命名张量、量化支持以及类型提升等。这些更新显著提升了 PyTorch 在移动端部署、模型性能和易用性方面的能力。
2022.09		PyTorch正式加入Linux基金会
2023.03	**PyTorch 2.0**	PyTorch 2.0 的核心特性可以概括为：通过 torch.compile API，提供了模型编译功能，能够在不改变模型代码的情况下加速模型，同时保持了与 PyTorch 1.x 的完全向后兼容性。

https://pytorch.org/foundation

Get in Touch

The success of PyTorch is only possible with the contributions and support of our developer community and member companies. To learn more about how you can join your industry peers in supporting PyTorch, or for any questions you may have, please fill out this form to be contacted by a PyTorch Foundation representative.

First name*

Last name*

Email*

Country*

Please Select

Phone number

Member

Yes

No

Topic*

Please Select

Comments*

By submitting this form, I consent to receive marketing emails from the LF and its projects regarding their events, training, research, developments, and related announcements. I understand that I can unsubscribe at any time using the links in the footers of the emails I receive. Privacy Policy

SUBMIT

2.4 ▼

Search Docs

Community [+]

Developer Notes [+]

Language Bindings [+]

Python API [-]

torch

torch.nn

torch.nn.functional

torch.Tensor

Tensor Attributes

Tensor Views

torch.amp

torch.autograd

torch.library

torch.cpu

torch.cuda

Understanding CUDA Memory Usage

Generating a Snapshot

Using the visualizer

Snapshot API Reference

torch.mps

torch.xpu

torch.mtia

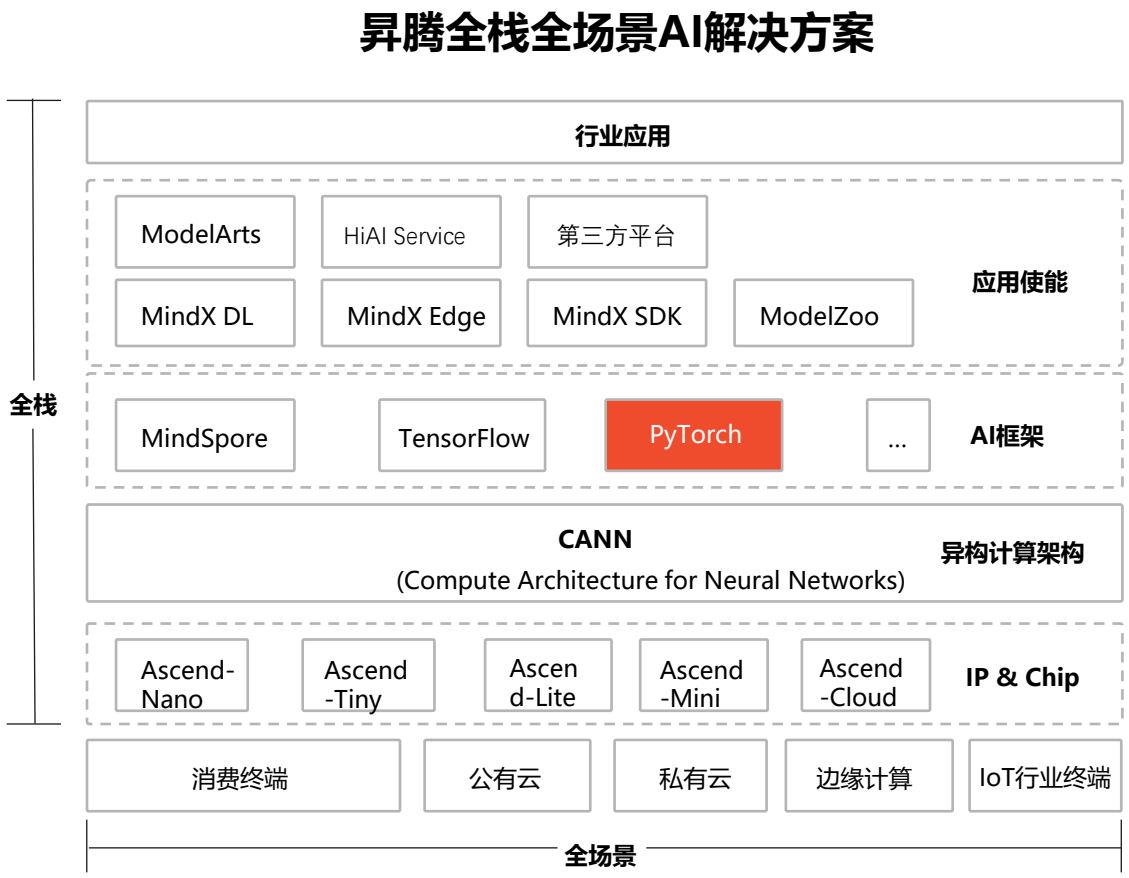
Docs > PyTorch documentation

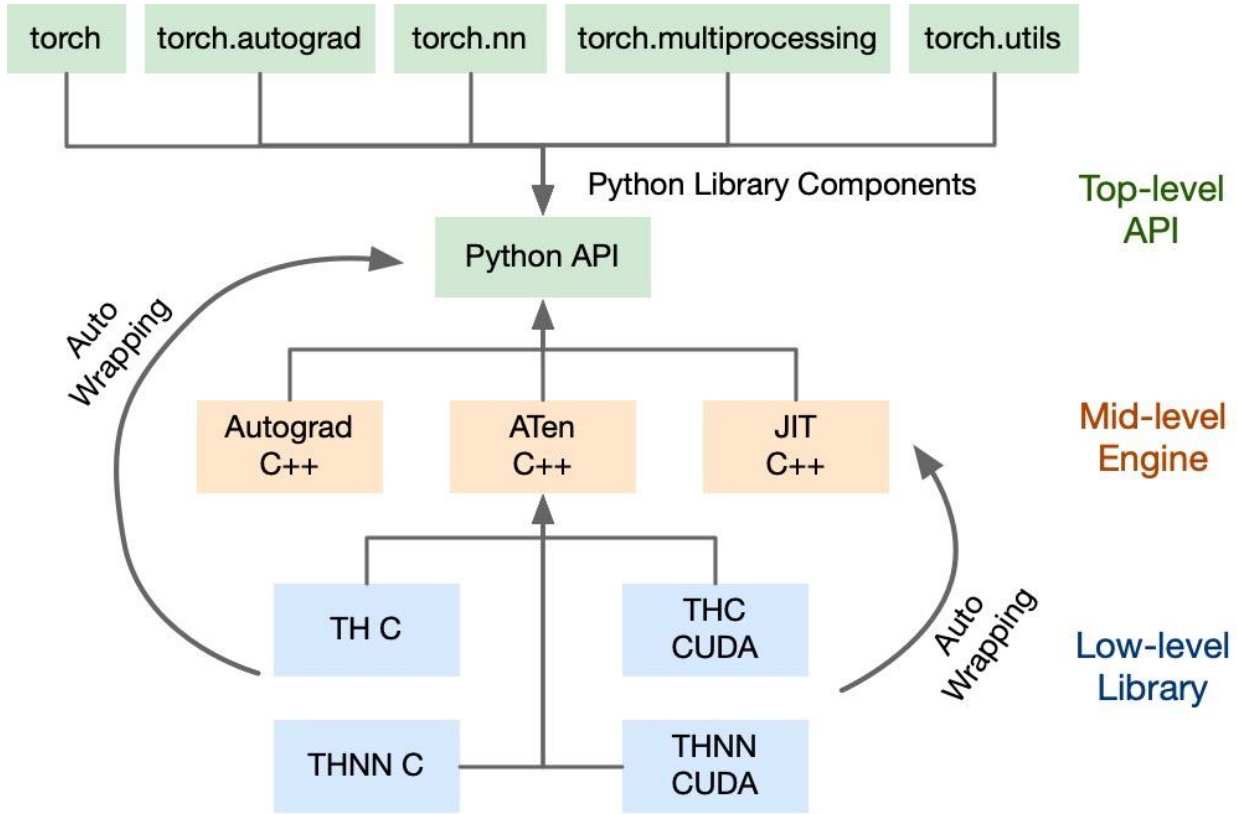
- Utilities
- Quantized Functions
- Lazy Modules Initialization

- torch.nn.functional
  - Convolution functions
  - Pooling functions
  - Attention Mechanisms
  - Non-linear activation functions
  - Linear functions
  - Dropout functions
  - Sparse functions
  - Distance functions
  - Loss functions
  - Vision functions
  - DataParallel functions (multi-GPU, distributed)
- torch.Tensor
  - Data types
  - Initializing and basic operations
  - Tensor class reference
- Tensor Attributes
  - torch.dtype
  - torch.device
  - torch.layout
  - torch.memory_format
- Tensor Views

PyTorch功能定位

PyTorch框架具有很多框架优势，简单易用、设备切换简单、灵活的动态图设计，PyTorch目前在开源生态中使用广泛。昇腾芯片在PyTorch上进行适配，可以加快昇腾全栈全场景AI解决方案推广使用。





https://pytorch.org/foundation

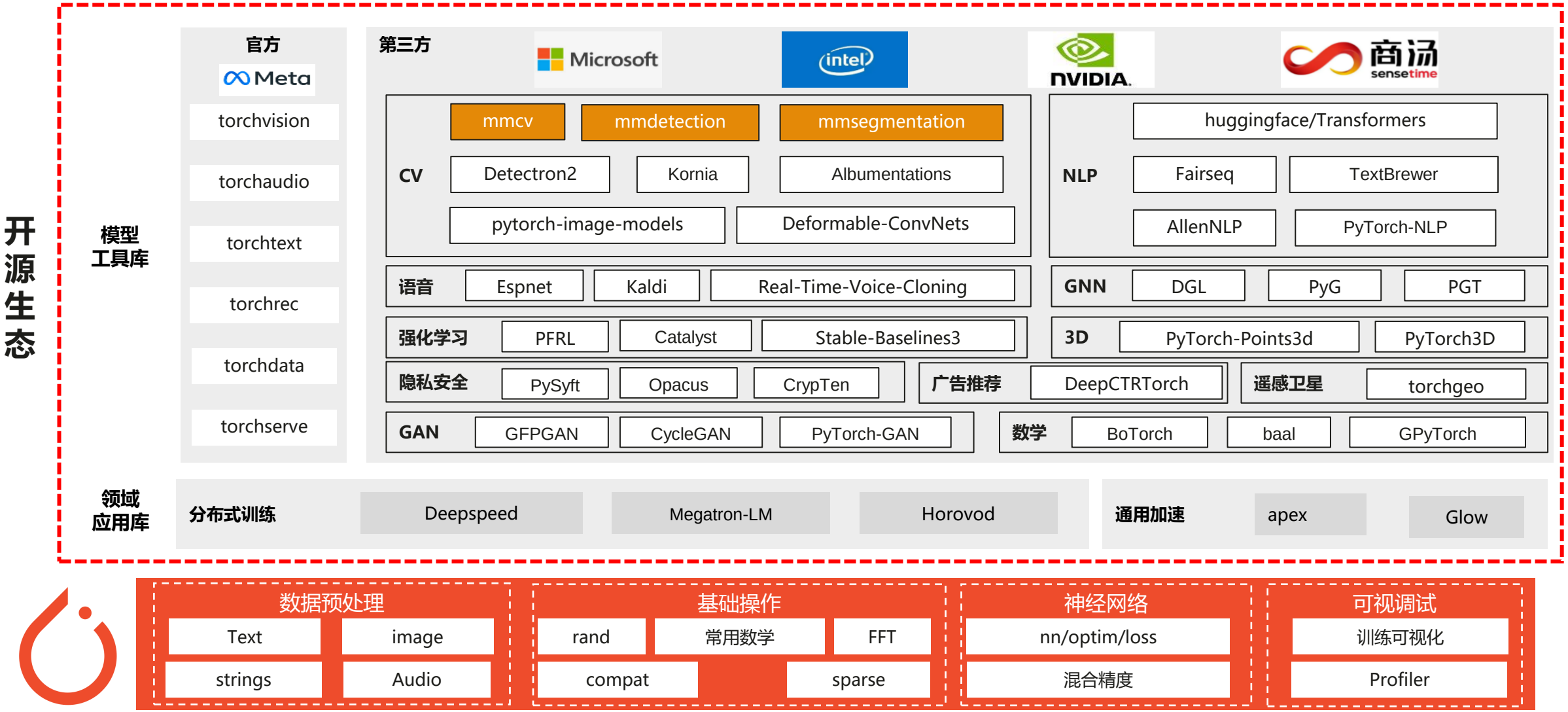
Premier Members

02

PyTorch 框架架构

我没有副标题哦

PyTorch底层聚焦易用性，向上做好社区服务赋能三方库

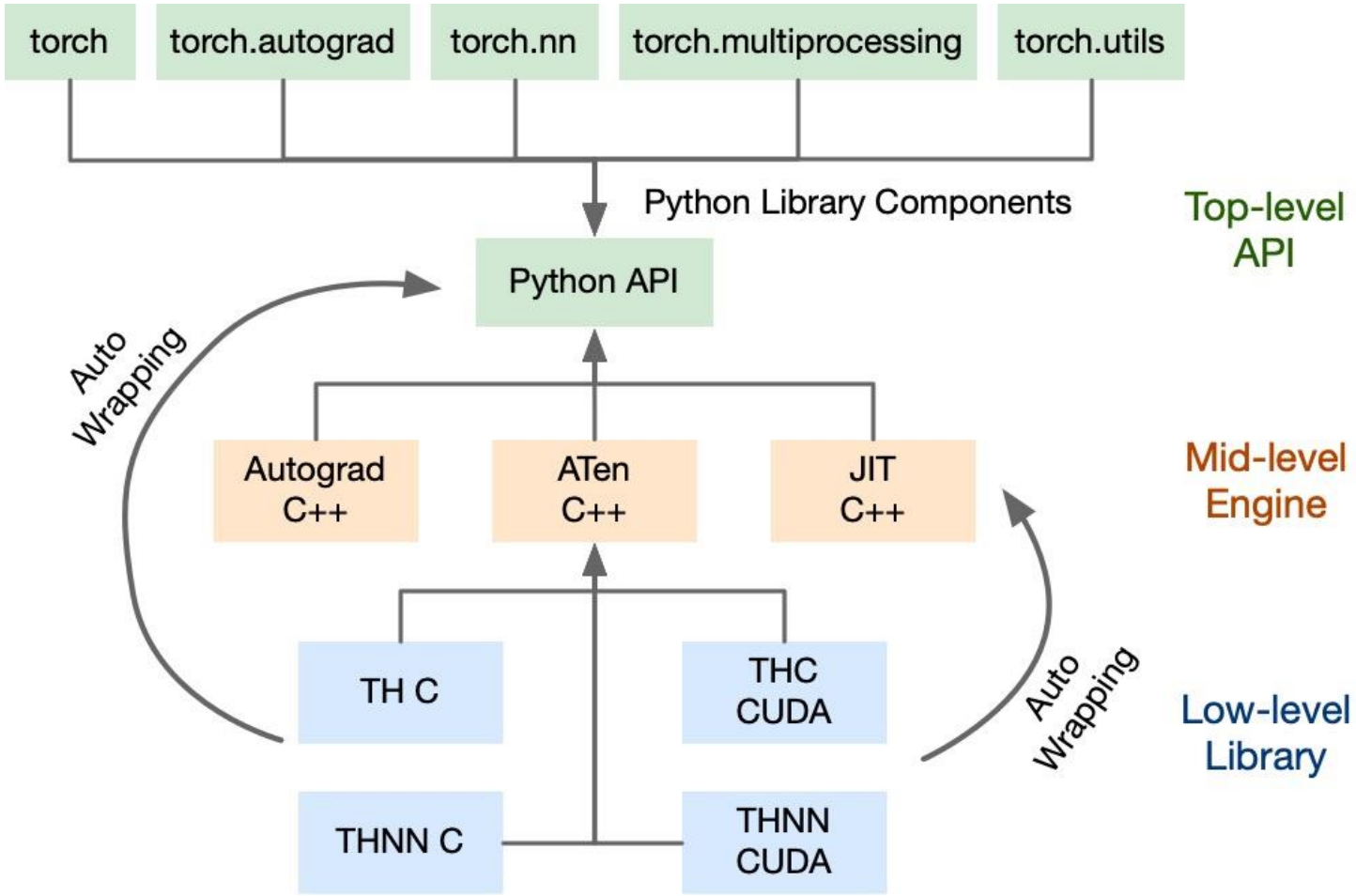


PyTorch 框架没有核心技术

PyTorch 框架没有核心技术

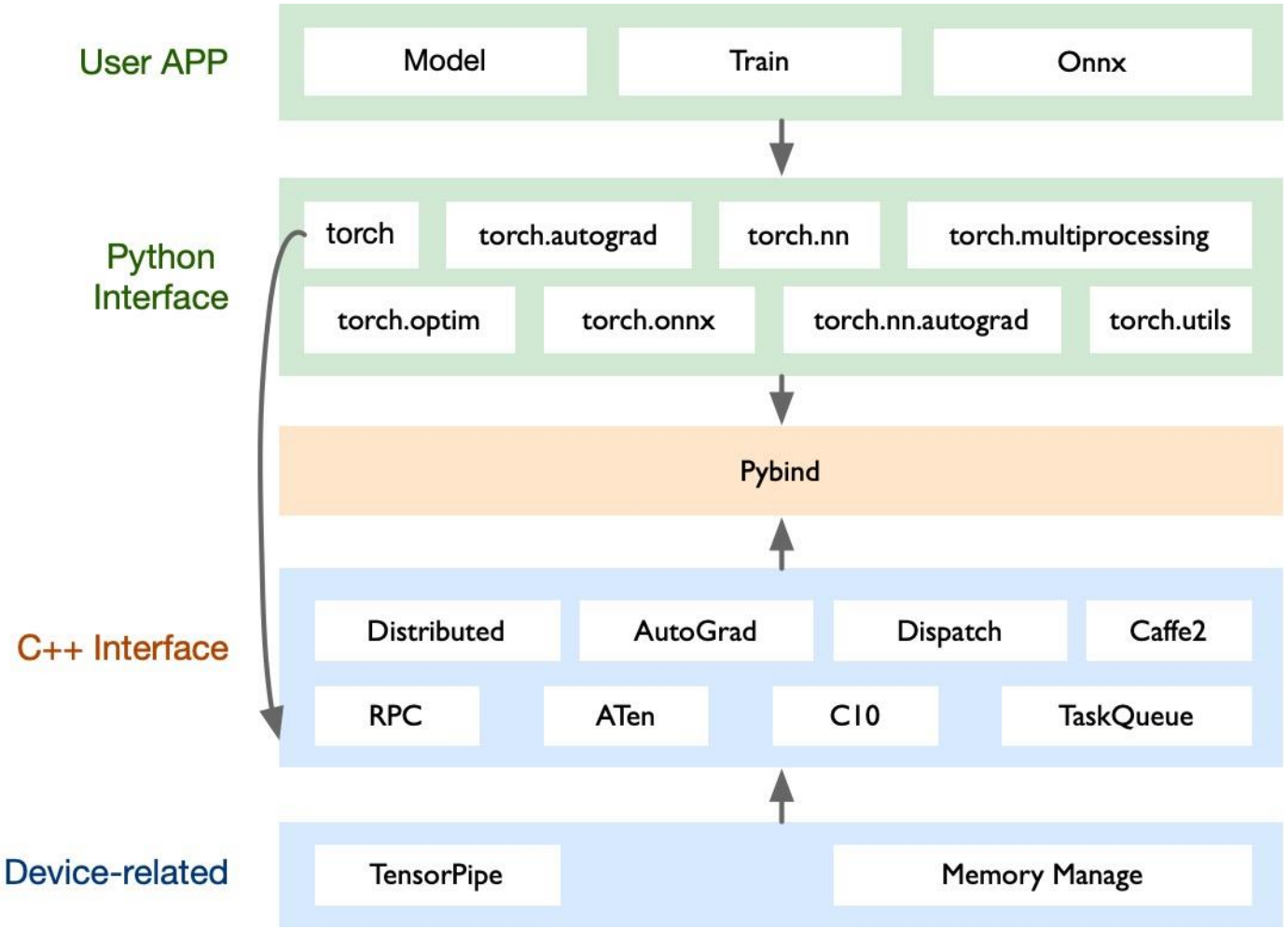
- 动态计算图（Dynamic Computational Graph）
- 分离控制流（program branches）与数据流（tensors，operations）
- Dynamo 从 Python 解析层实现最接近完美的动静统一
- 内存管理机制，针对底层硬件进行加速优化
- 多进程机制优化，Tensor引用计数机制
- 高效 C++ 核心实现
- .....
- 整体架构的设计理念：命令式编程实现动态计算图，通过C++分离宿主语言对AI描述，实现内存、多进程、硬件到大规模集群的加速

PyTorch 框架简单分层



1. Python API：宿主语言的API层
2. AutoGrad：自动微分模块
3. ATen：Tensor的C++原型定义模块
4. JIT：优化的即时编译器模块

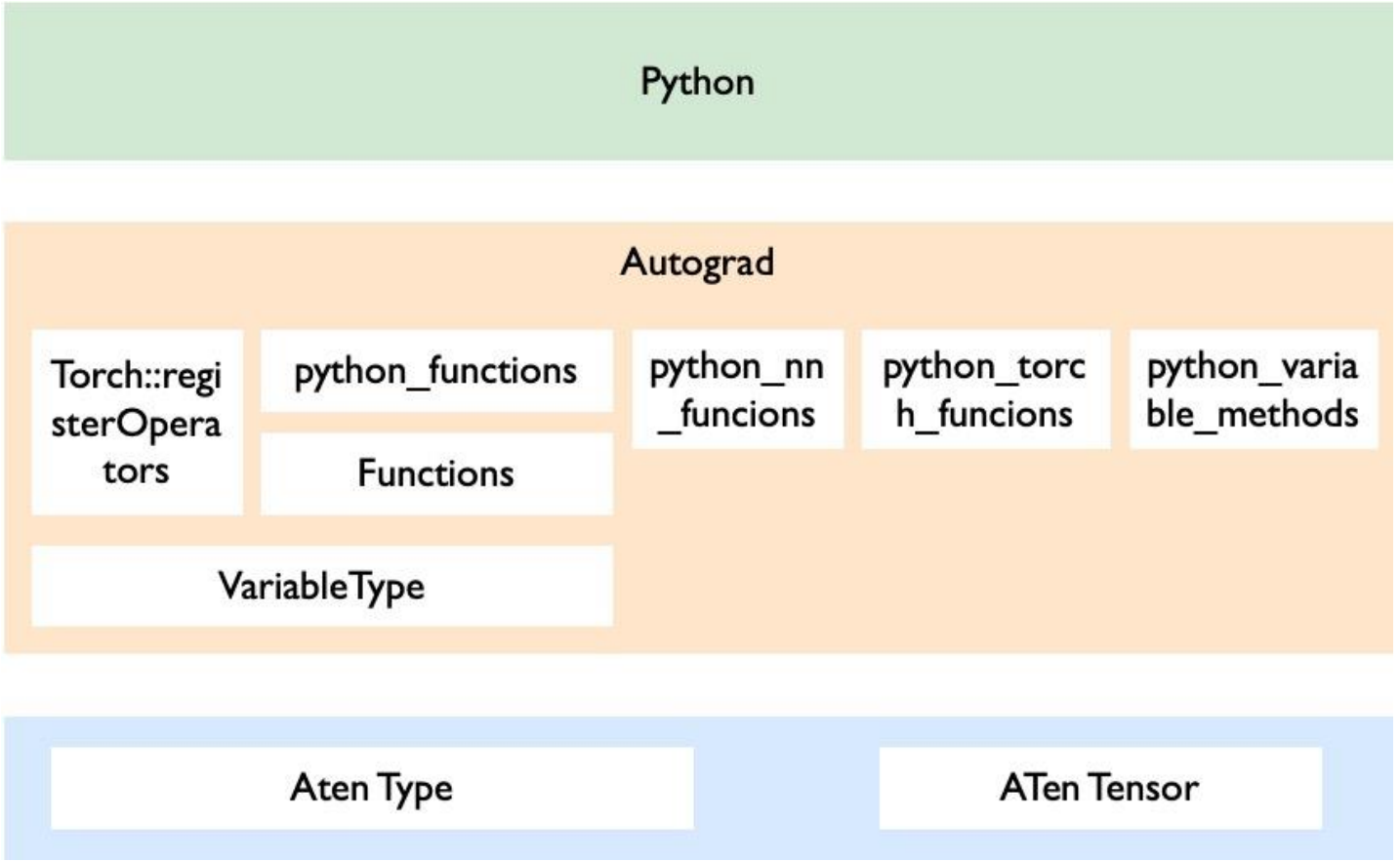
Pytorch组件关系



1. torch.nn.ops /torch.Module：算法开发提供算子API接口，构建AI模型
2. torch.optim：优化器，在模型训练过程中调整模型参数
3. Aten/c10/c10d：Tensor的C++原型定义和Tensor基本操作、算子定义
4. AutoGrad：自动微分模块，训练过程中求模型参数梯度
5. Dispatch：算子分发模块，根据后端选择对应的算子
6. TaskQueue：多线程调度模块
7. Memory manage：内存管理模块
8. TensorPipe：底层通信模块

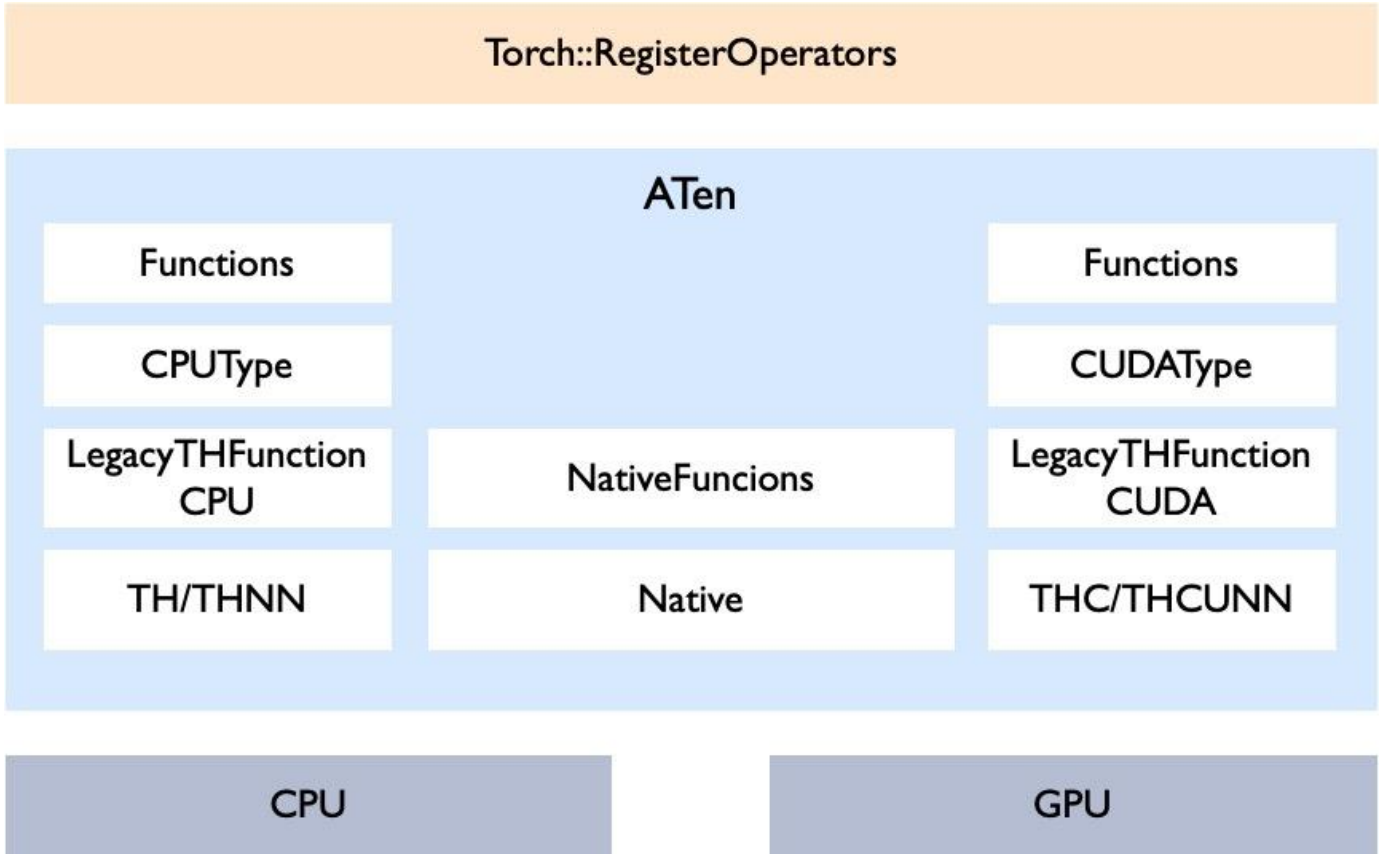
PyTorch 核心组件关系

AutoGrad：模块是一种自动微分系统，提供了对Tensors 上所有运算操作的自动微分功能，属于define-by-run 类型框架。



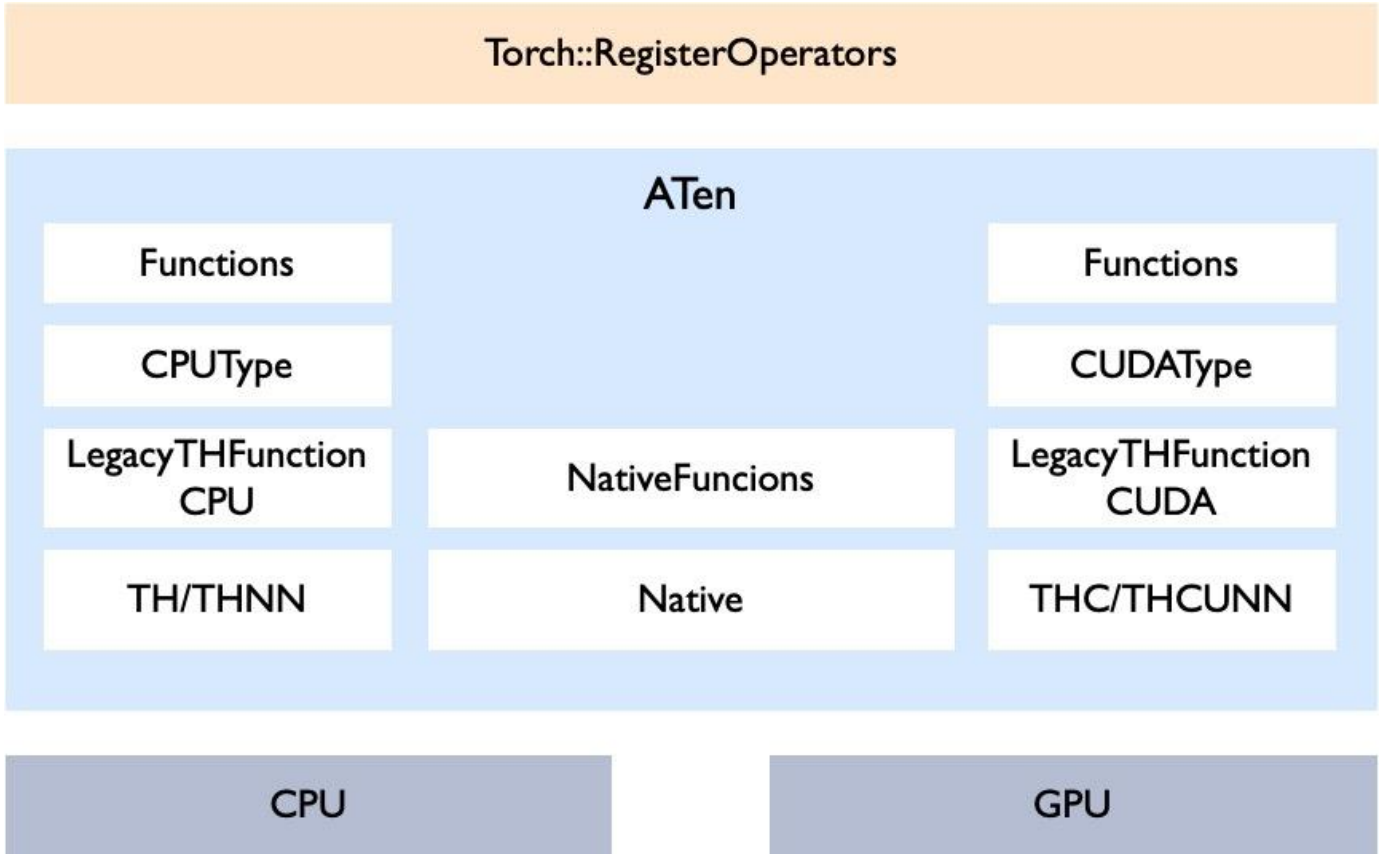
PyTorch 核心组件关系

- **C10：** Caffe Tensor Library 缩写，目录主要实现了最基础的数据的管理。目前存放的都是最核心、最精简的、最基础的Tensor库代码，比如Tensor，Storage，Allocator类都在这里面。



PyTorch 核心组件关系

- **ATen：** A Tensor library for C++11 缩写，目录主要声明和定义 Tensor 运算相关的逻辑处理和计算处理，比如卷积，池化计算等，是 PyTorch 的 C++ Tensor 库。

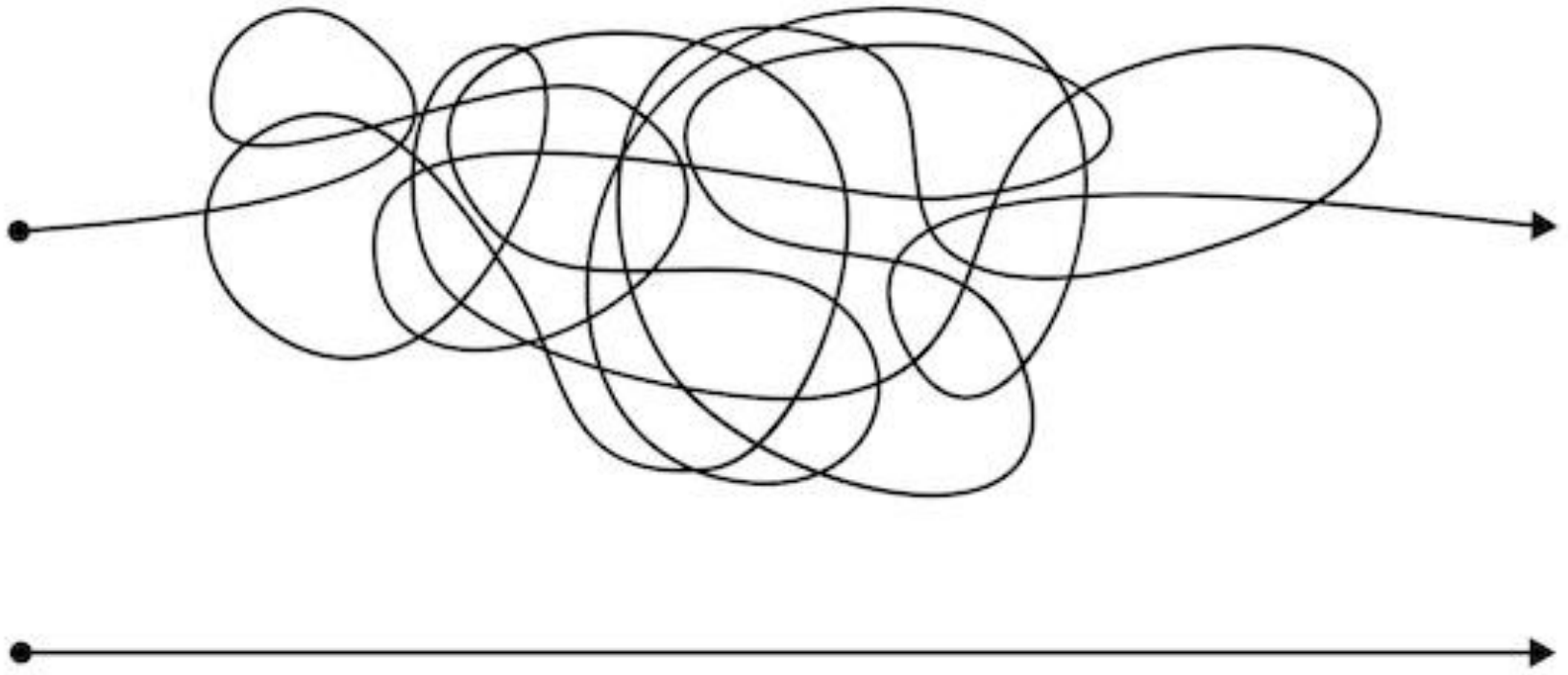


03

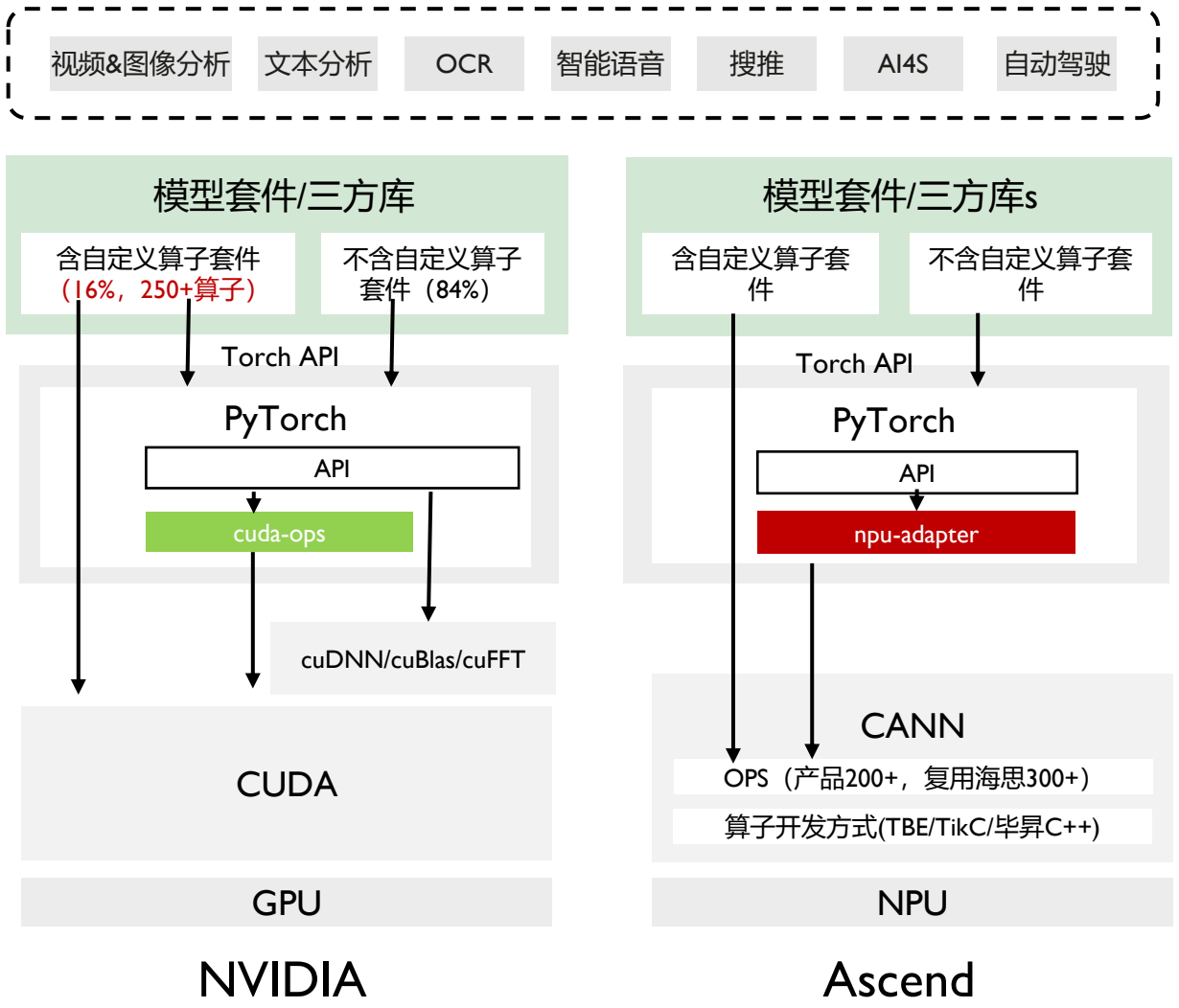
昇腾遇上 PyTorch

想说爱你不容易

昇腾在PyTorch上的尝试



PyTorch+昇腾总体架构



PyTorch适配昇腾，我们具体做什么

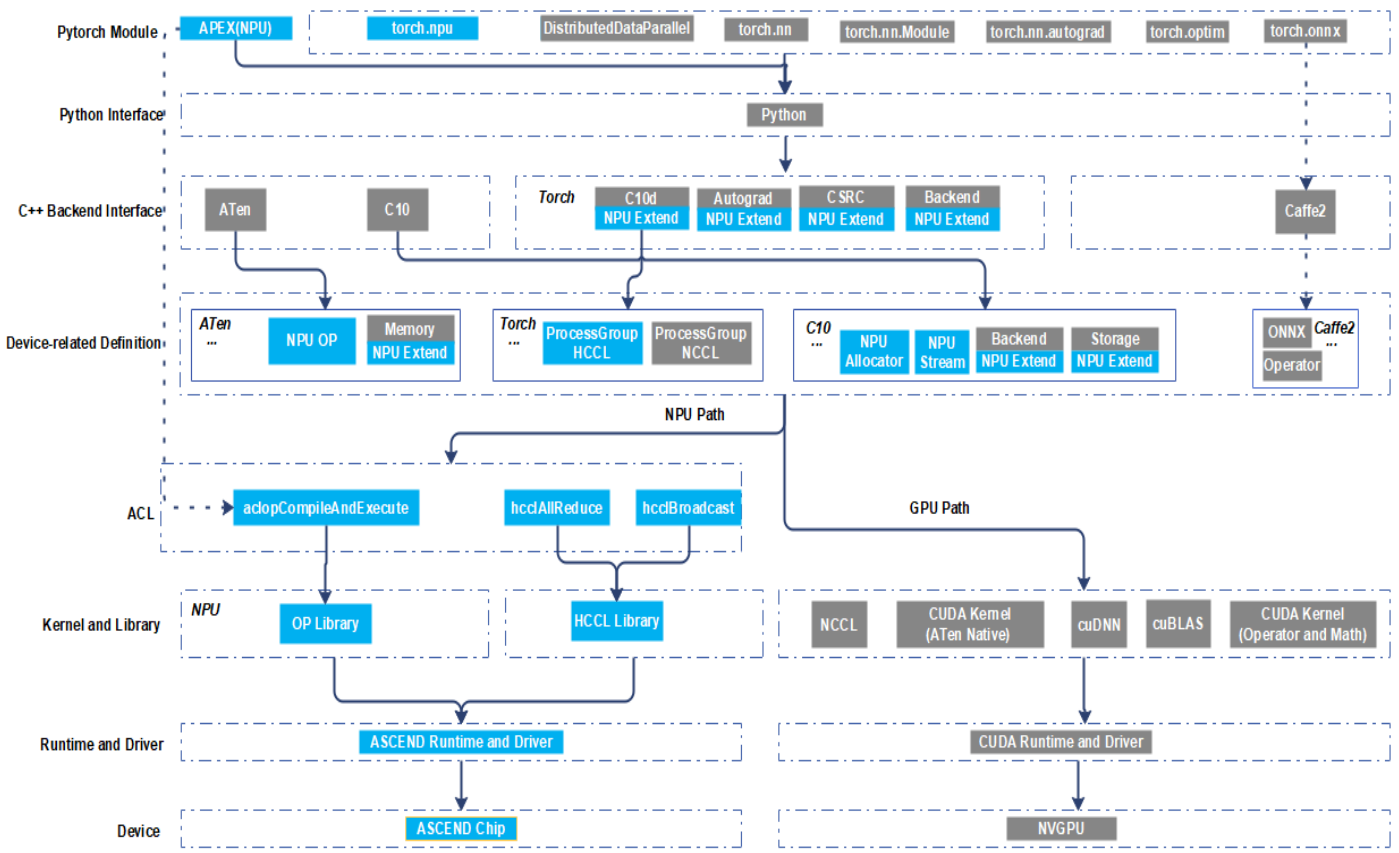
1. **算子补齐开发：**PyTorch自带大量算子，需要在昇腾设备上适配这些算子。
2. **框架版本适配：**插件化适配，接入框架算子，以及进行框架新特性适配开发。
3. **模型迁移&调优：**模型从GPU迁移到NPU，进行模型性能和精度调优、分布式训练等。
4. **套件和三方库适配：**套件中有丰富的模型和算子，三方库补齐PyTorch的能力，适配时需要补齐算子，调测模型。
5. **工具链（MindStudio）：**提供精度对比、溢出检测、Profiling工具、算子开发等工具，支撑模型调优。

初代方案1：侵入式适配方案

1. **断代研发，版本升级困难：**PyTorch版本升级较快，每次需要重新适配新代码，开发周期很长；

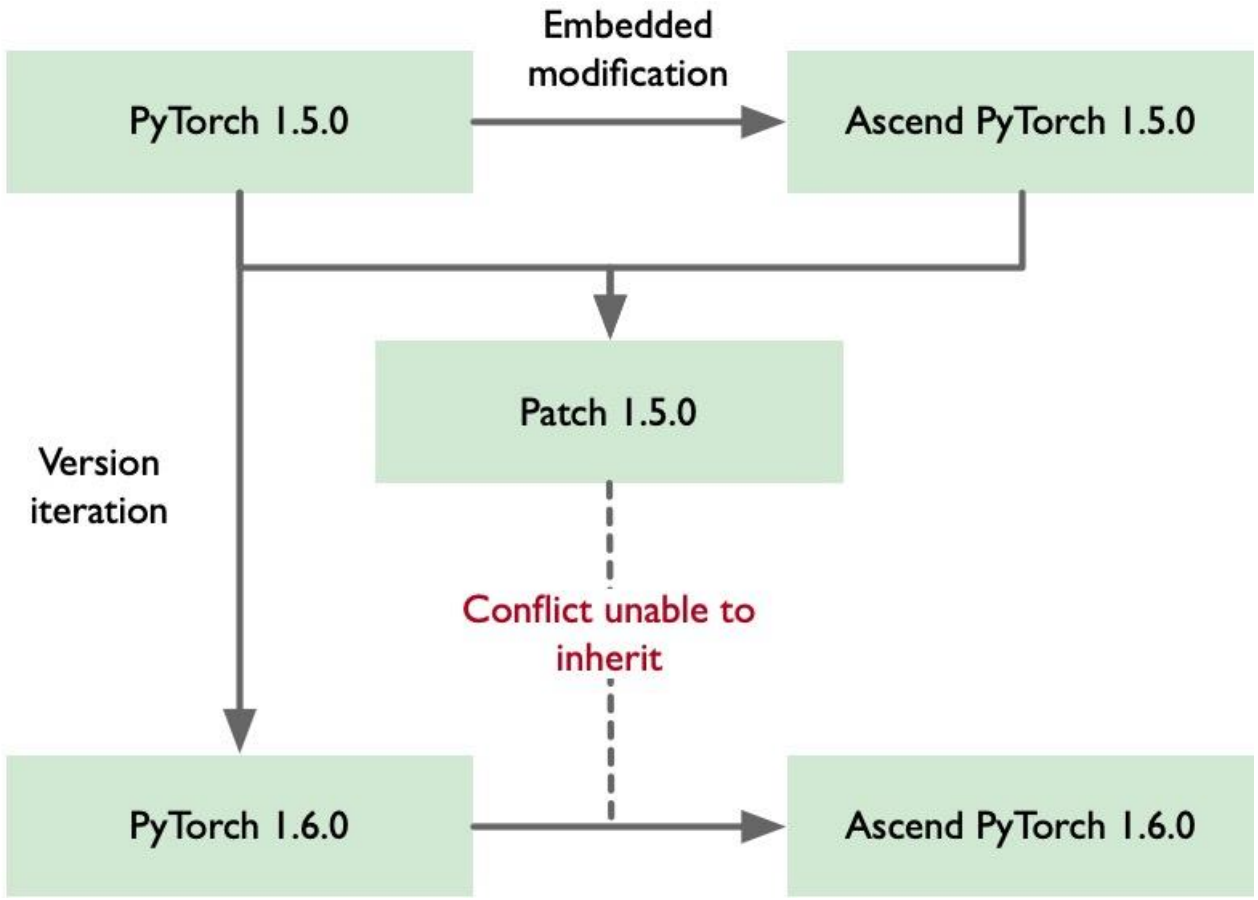
2. **代码量大，维护困难：**代码严重耦合，影响框架正常功能；代码庞大，社区开发者和使用者基本没有可能贡献代码；

3. **安装困难，用户使用门槛高：**需要基于PyTorch源码编译安装，耗时较长；PyTorch第三方库较多，对网络环境有较高依赖，下载困难；支持版本范围有限。



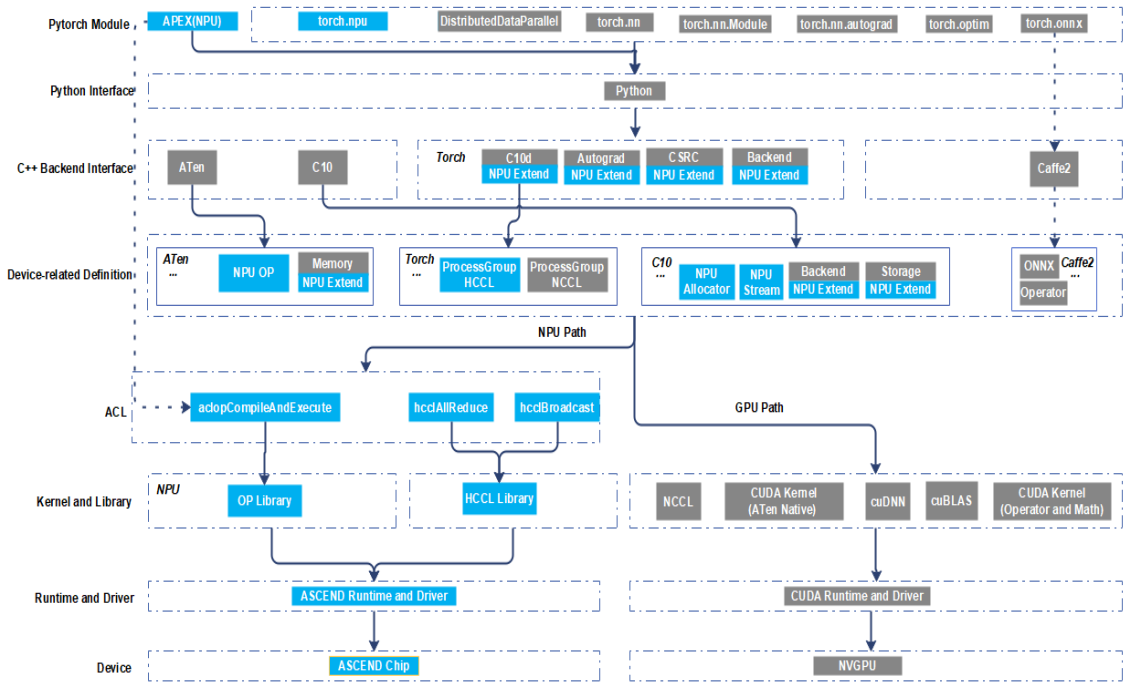
初代方案痛点1：断代研发，版本升级困难

- 官方PyTorch版本间存在重构修改，侵入式修改无法继承使用，版本升级人力成本高，难以快速支撑客户需求。



初代方案痛点2：代码量大，开发维护困难

- 侵入式修改PyTorch框架，代码量巨大（约100W行），安全问题难以定界，全量扫描整改压力大



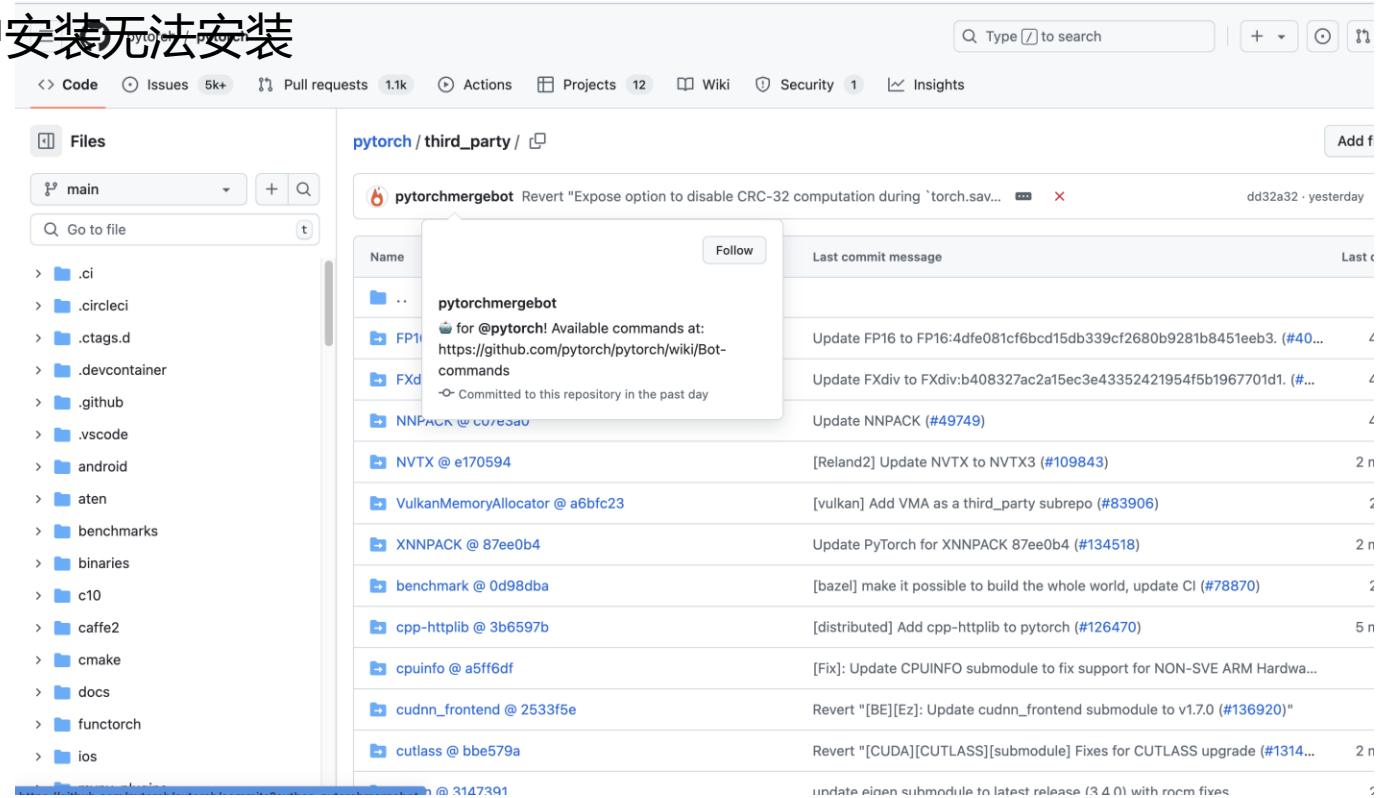
约10W行　　约90W行

```
--- pytorch-v1.5.0/aten/src/ATen/native/BatchLinearAlgebra.cpp  2021-04-10 18:39:32.000000000 +0800
+++ pytorch-develop-150/aten/src/ATen/native/BatchLinearAlgebra.cpp 2022-08-11 23:00:30.883471525 +08
@@ -680,7 +680,7 @@
 std::tuple<Tensor&, Tensor&> triangular_solve_out(Tensor& result, Tensor& clone_A, const Tensor& sel
                                                   bool upper, bool transpose, bool unitriangular) {
   Tensor result_tmp, clone_A_tmp;
-  std::tie(result_tmp, clone_A_tmp) = at::_triangular_solve_helper(self, A, upper, transpose, unitri
+  std::tie(result_tmp, clone_A_tmp) = at::native::triangular_solve(self, A, upper, transpose, unitri
   result.resize_as_(result_tmp).copy_(result_tmp);
   clone_A.resize_as_(clone_A_tmp).copy_(clone_A_tmp);
   return std::tuple<Tensor&, Tensor&>(result, clone_A);
diff -Nur '--exclude=.git*' '--exclude=.jenkins' '--exclude=android' '--exclude=OWNERS' '--exclude=th
--- pytorch-v1.5.0/aten/src/ATen/native/cpu/Activation.cpp  2021-04-10 18:39:32.000000000 +0800
+++ pytorch-develop-150/aten/src/ATen/native/cpu/Activation.cpp 2022-08-11 23:00:30.899471524 +0800
@@ -339,20 +339,20 @@

 void hardsigmoid_backward_kernel(TensorIterator& iter) {
   AT_DISPATCH_FLOATING_TYPES(iter.dtype(), "hardsigmoid_backward", [&] {
-    auto zero = scalar_t(0.0f);
-    auto one = scalar_t(1.0f);
+    auto neg_three = scalar_t(-3.0f);
+    auto three = scalar_t(3.0f);
     using Vec = Vec256<scalar_t>;
     Vec kZeroVec(0.0f);
     Vec kOneSixthVec(1.0f / 6.0f);
     cpu_kernel_vec(
         iter,
```

痛点3：安装困难，用户使用门槛高

- 基于源码编译，第三方依赖复杂且存在缺失，用户安装极其困难
- 依赖库50+，且存在缺失
- 平均安装时间大于一天
- 部分用户安装无法安装



缺失

03

昇腾遇上 PyTorch

改变不了你，就改变自己