

AI 系统 组成&生态



ZOMI



Talk Overview

1. AI 系统概述

- AI 历史，现状与发展
- 算法与体系结构的进步
- AI 系统的组成与生态
- 大模型对AI系统的挑战

2. AI 芯片

3. AI 编译器

4. AI 推理引擎

5. AI 开发框架

6. 异构集群调度与管理

7. AI 大模型

Talk Overview

- AI 系统的组成与生态

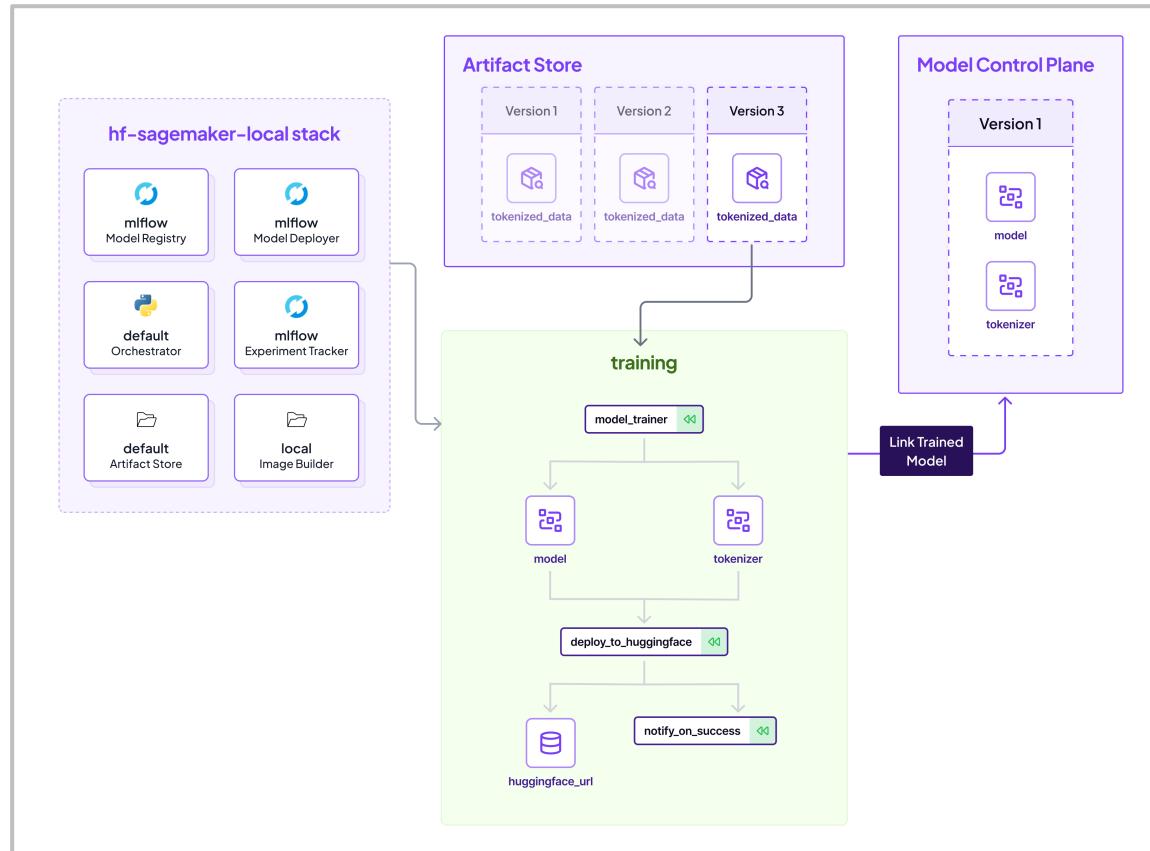
1. AI System Design Objectives – AI 系统设计目标

2. AI system composition – AI 系统组成

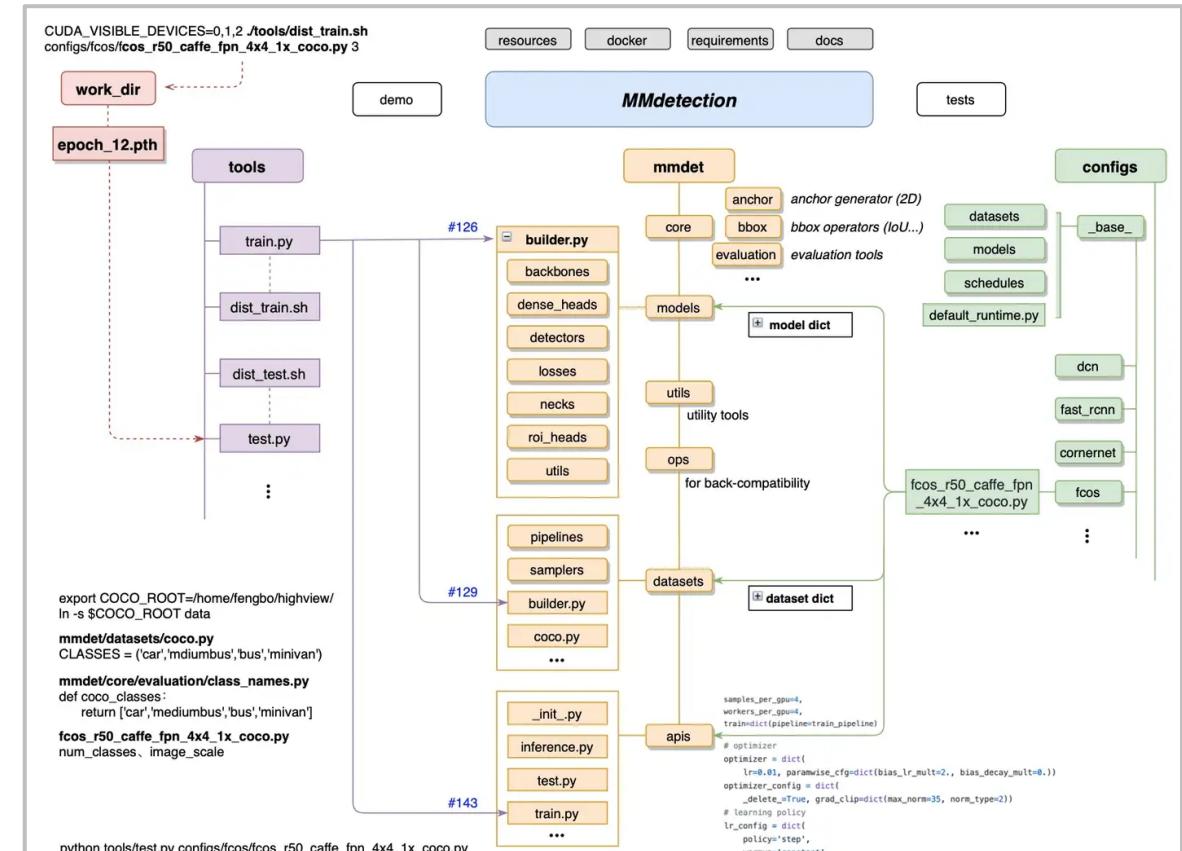
3. AI system Stack Tech – AI 系统技术详图

4. AI System Ecology – AI 系统生态

Huggingface 组件架构



MMDetection 套件架构





1. AI 系统设计目标



高效编程语言、框架、工具

- **编程语言**：更具 AI 表达能力和简洁的神经网络计算原语和编程语言。让开发者能够提升开发效率，屏蔽底层细节，更灵活的编程原语支持。
- AI 除了特定领域模型的算子和流程可以复用（例如，LLM 在 NLP 领域被广泛作为基础结构），其他领域的新的结构、新算子的设计与开发仍以试错（Trial And Error）的方式进行。如何利用宿主语言（或者原生语言如Python）灵活表达新的算子、新的AI范式，屏蔽底层实现所对上暴漏的API接口呢？

高效编程语言、框架、工具

- **开发框架**：更直观编辑、调试和实验工具。让开发者可以完整从 AI 开发、测试、调优与修复程序，提升所开发 AI 算法的性能与鲁棒性。
- 训练过程不是一蹴而就，其中伴随着不收敛、Nan、OOM 内存溢出等算法问题与系统缺陷（Bug），工具与系统本身如何在设计之初就考虑到这点，提供良好的可观测、可调试性，允许开发者自定义扩展等支持，需要工具与系统设计者在系统设计之初就考虑到的。

高效编程语言、框架、工具

- **全生命周期**：支持深度学习生命周期中的各个环节，从模型压缩、推理、安全、隐私保护等。不仅能构建 AI 算法，支持全生命周期的 AI 应用开发，并在系统内对全生命周期进行分析与优化。
- 当前的 AI 工程化场景，已经不是灵感一现单一优化就能迅速取得领先优势，更多的是能否有完善的 AI 基础设施，快速复现 AI 社区工作，批量验证新的想法和试错。因此，一套完善全流程生命周期管理能够大幅度提升 AI 算法生产力。

支持快速变化 AI 算法和任务

- 除深度学习训练与推理，还能支持强化学习、AutoML、Stable Diffusion、GAN 等**新训练范式**。
新范式造成对之前单一支持单模型外，在多模型层面，训练与推理任务层面产生了新的 AI 系统抽象与资源，作业管理需求。
- **提供更强大和可扩展的计算能力。**让开发者的 AI 应用可扩展，并部署于可以并行计算的节点或者集群，应对大数据和大模型挑战。当前 AI 应用通过大模型产生更好的效果，促使系统需要 AI 集群，同时由于企业 IT 基础设施不端完善，能够不断沉淀新的数据，也会伴随着大数据的问题。大模型与大数据促使存储与计算层面系统在摩尔定律失效大背景下，通过并行与分布式计算的范式扩展算力与存储的支持。

新挑战下系统设计、实现和演化问题

- **自动推导计算图**：通过符号执行或即时编译，获取计算图信息，进而数据驱动方式做定制化的优化，如PyTorch2.0 Dynamo。
- **自动并行**：面对部署场景的多样化体系结构，训练阶段异构硬件的趋势，框架让用户透明的进行任务放置，并行化，以期以最为优化的方式在指定硬件配置下，并行化与减少 I/O，逼近硬件提供的性能上限。
- **弹性计算**：并扩展到多个计算节点，面对云与集群场景，如何自动将任务扩展与部署，进而支撑分布式计算，弹性计算，让开发者按需使用资源，也是云原生背景下，AI 系统所需要考虑和支持。

新挑战下系统设计、实现和演化问题

- **持续优化**：由于 AI 训练业务是迭代且周期较长，给系统以内省优化的机会，即不断监控，不断优化当前系统配置与运行时策略，以期弥补纯静态优化获取信息不足，运行时干扰造成的相比最优化的策略的差距。
- **通用泛化**：探索并解决新挑战下的系统设计、实现和演化的问题。动态性支持、利用稀疏性进行加速优化、混合精度训练与部署、混合训练范式（强化学习）等。

提供大规模 AI 集群的企业级环境部署

- **多租环境训练部署**：面对多组织，多工程师共享集群资源，以及迫切使用 AI 算力资源日益增长的需求，提供公平，稳定，高效的多租环境。
- **跨平台推理部署**：面对割裂的边缘侧硬件与软件栈，让模型训练一次，跨平台部署到不同软硬件平台，是推理场景需要解决的重要问题。
- **安全与隐私的需求**：AI 算法类似传统程序接受输入，处理后产生输出。相比传统程序其解释性差，造成更容易产生安全问题，容易被攻击。同时模型本身的重要信息为权重，要注意模型的隐私保护。如果是企业级环境或公有云环境，会有更高的安全和隐私保护要求。



AI 系统设计的4个小目标

1. 提供高效的 AI 编程语言、开发框架和全流程工具（AI框架、AI推理引擎）
2. 系统级支持快速变化的 AI 算法和任务（AI 芯片、AI 框架、AI编译器泛化能力）
3. 探索并解决新挑战下的系统设计、实现和演化问题（大模型、AI4SIC等应用）
4. 提供大规模 AI 集群的企业级环境部署（AI 集群管理与调度）

2. AI 系统组成



AI系统全栈架构图



应用层：提供具体算法

- 为不同应用场景提供已经建模好和优化好的不同AI算法。
- 提供算法：(1)、计算机视觉.
(2)、语音语义. (3)、机器学习.
(4)、自然语言处理
- 面向单位: (1)、科研院所. (2)、开源社区、(3)、应用单位



开发层：提供前端编程语言

- 提供用户前端的编程语言，接口和工具链。
- 尽可能让用户表达 AI 任务与算法，尽量少让用户关注底层实现（如声明式编程、命令式编程）是提升开发体验的较好的手段。
- 目前除了Python以外，还有 Moj 等。



框架层：提供面向 AI 业务表示抽象

- 负责静态程序分析与计算图构建，编译优化等。
- 框架本身通过提供用户编程的 API 获取用户表达 AI 模型，数据读取等意图，在静态程序分析阶段完成尽可能的自动前向计算图构建，自动求导补全反向传播计算图，计算图整体编译优化，算子内循环编译优化等任务。



编译编程：提供硬件编程模型 & 框架 IR 编译优化

- 编程模型：如 CUDA 编程模型，一是通过层次结构来组织软件执行线程，二是通过层次结构来组织硬件内存的访问。



编译编程：提供硬件编程模型 & 框架 IR 编译优化

- 运行时：负责系统的运行时系统动态调度与优化。当获取的AI模型计算图部署于单卡，多卡或分布式环境，运行期的框架需要对整体的计算图按照执行顺序调度算子与任务的执行，多路复用资源，硬件驱动，做好内存等资源的分配与释放。



编译编程：提供硬件编程模型 & 框架 IR 编译优化

- **编译优化**：编译器或框架根据算子计算语义，对算子执行方式进行优化，降低内核启动与访存代价。
- 同时深度学习编译器还支持循环优化等类似传统编译器的优化策略和面向 AI 的优化策略。



硬件体系结构：AI 芯片 & 网络加速器

- **硬件接口抽象**：GPU\TPU\NP
U和各种 DSA。统一硬件接口
抽象可以复用编译优化策略，
让优化与具体底层设备和体系
结构适当解耦。
- **可扩展网络**：RDMA , InfiniBand , NVLink 等。提供高效 NPU
到 NPU 互联，更高带宽，更
灵活通信原语与高效的通信聚
合算法。



3. AI 系统技术栈



AI系统技术栈

应用算法 (Algorithm)	领域算法	视觉、自然语言处理、语音语义 : CNN、RNN、LSTM、GAN、Transformer、Stable Diffusion、Reinforce Learning				
开发编程 (Development)	集成开发环境	编程环境 : VS Code, Jupyter Notebook, PyCharm				
	编程语言	与主流语言集成 : PyTorch and Tensorflow etc. inside Python				
框架 (Frameworks)	AI框架、推理引擎	中间表达 (IR)、前端 API	编译优化 (Compilation)	图优化与图执行调度	推理优化	
		基本数据结构 : 张量 (Tensor)	词法、语法、语义分析	计算图动静统一	常量折叠&冗余节点消除等	
		基本计算单元 : 有向无环图 (DAG)	自动微分	计算图控制流	模型压缩 : 蒸馏剪枝与量化	
异构与编译 (Compiler)	编程编译	前端优化	数据格式布局转换	内存分配	公共表达式消除	
		后端优化	代码优化、代码生成	算子循环优化	令和内存优化	
		编程模型 : CUDA/Ascend C/Band C		AI编译器 : TVM、XLA、TC		
		LLVM/GCC				
	底层通信	HCCL/NCCL、通信原语、集合通信算法				
体系结构 (Architecture)	底层硬件	AI芯片 : CPU/GPU/AISC/FPGA/TPU/NPU				



4. AI 系统生态



核心 AI 系统软硬件

核心 AI 系统软硬件

AI 任务
执行和编译优化

AI 集群资源
管理和调度

AI 加速器
及高性能网络

AI 算法和框架

AI 算法和框架

新 AI 算法
新 AI 范式支持

AI 框架、AI 推理引
擎支持与演进

AI 编译架构
及优化

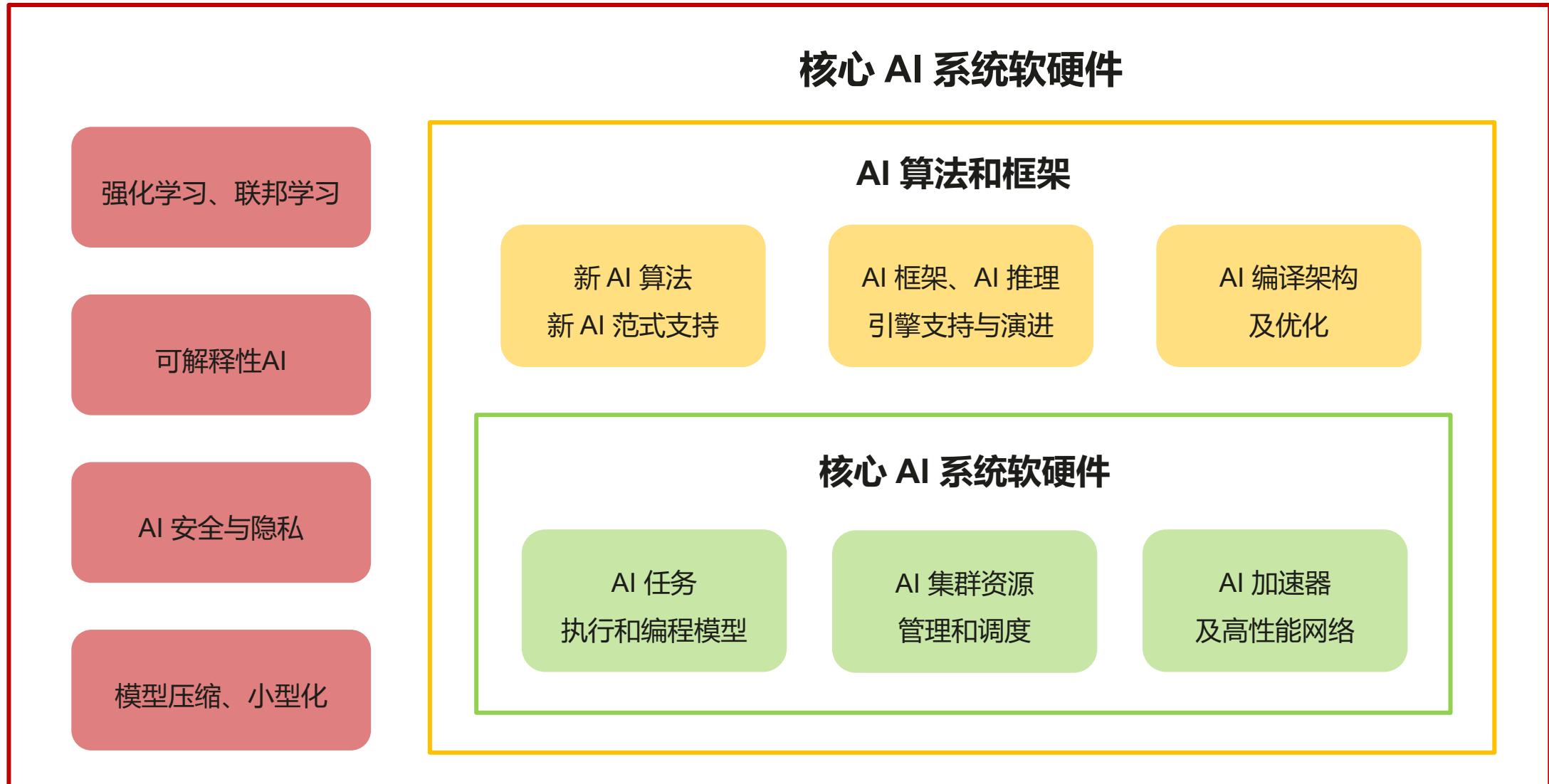
核心 AI 系统软硬件

AI 任务
执行和编程模型

AI 集群资源
管理和调度

AI 加速器
及高性能网络

广泛的 AI 系统生态



总结



总结

1. AI 系统设计目标是面向开发者提供不同层次软硬件组成，支持从底层芯片到应用的每一层的系统任务和目标。
2. AI 系统组成为 5 层：应用层、开发层、框架层、编程编译层、体系结构层，涵盖 AI 系统全栈。
3. AI 系统技术详图中打开了每一层的主要技术点，AI 系统全栈介绍中会介绍和侧重的点。
4. AI 系统生态实际上很广泛，我们需要做到有所为、有所不为，聚焦在行的领域发力。



Thank you

把AI系统带入每个开发者、每个家庭、
每个组织，构建万物互联的智能世界

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

Copyright © 2023 XXX Technologies Co., Ltd.
All Rights Reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. XXX may change the information at any time without notice.



Course chenzomi12.github.io

GitHub github.com/chenzomi12/DeepLearningSystem