# SPU产生原因

实际需求：用户向大模型提问，用户的提示词需要隐私保护，大模型在加密环境推理，用户得到结果后解密查看。模型和提示词都得到了保护

数据的重要性：

互联网公司最重要的资产

数据联合后又更大的价值

隐私计算技术路线：

MPC(HE, SS)

DP

TEE

…

隐私计算的困境：

1. 易用性差：类型简单、算子底层、抽象

（数据需要定义在环/域上，只有加、乘、与、或等基础算子）

1. 性能不好

需求：（SPU）

易用：类似高级编程语言（提供原生AI框架支持）

性能好：系统性优化（编译器优化）

# SPU简介

## SPU：虚拟的加密处理设备

分为前端、编译器、运行时三个部分

### 前端

组成：JAX、Tensorflow、PyTorch（原生AI前端，复用了一些能力，如自动求导）

目前JAX支持最好

### 编译器

SPU编译器目的是把机器学习前端的中间表达，加上隐私保护语义，翻译到加密计算友好的运行时。选用llvm ir框架进行二次编译和优化

编译器链路：-> XLA -> MLIR (PPHLO -> PPLLO) -> 下层MPC

PP: privacy preserving

HLO: high level ops

LLO: low level ops

PPHLO: 有平台无关的优化

PPLLO: 基于MPC协议的优化

这个编译器的好处是经过ir处理，用户写出来的代码就是安全的。

这个带隐私保护语义的ir简单来说，就是让任何和密态变量计算的明/密文产生的结果都成为密态变量

加密计算优化：例如，

一些顺序调整，明文计算结束后再算密文

选择器的公共部分提取出来

### 运行时

可以当成一个SIMD指令集的虚拟设备

SIMD的优化：让网络通信和矩阵计算交替进行

指令并行

数据并行

多种协议支持：内置SEMI2K，ABY3，Cheetah

多部署模式支持：

面向虚拟设备编程

一次编写到处执行

（也就是说把设备的计算能力都抽象到一台SPU设备上，不管实际有多少个计算节点）

## 编程界面

基本使用：

原生AI框架编程（jax.numpy和numpy）

使用JIT编译执行

生态无缝衔接（容易迁移AI的代码过来进行隐私保护）

更改安全协议：修改配置就行，代码逻辑不用改

优化和错误排查：profiling, tracing, debugging工具

# SPU现状和发展

开源共建：支持PPML、FL、SCQL

学术支持：计算机系统、隐私保护机器学习、密码协议创新

构建加密计算生态：作为AI和密码学的桥梁

构建类似sklearn的安全机器学习库

构建类似pandas的安全数据分析库

支持更多密码学后端