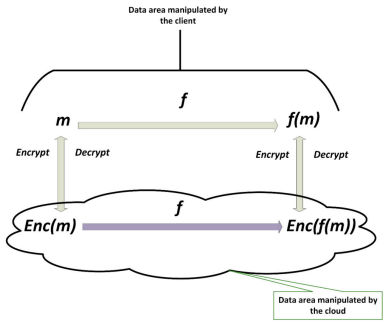
洞见科技 INSIGHTONE

# 全同态软硬协同加速

洞见科技 顾振

- **点**虽零散，机会不少
- **同**态虽卷，顶会不少

## 全同态加密的主要概念




- $\mathrm{Eval}_{ek(s)}(\mathrm{Enc}_s(\mu), f) = \mathrm{Enc}_s(f(\mu))$
- $\mathrm{Bootstrap}_s\big(\mathrm{Enc}_s(\mu)\big) = \mathrm{Eval}_{ek(s)}(\mathrm{Enc}_s(\mu), \mathrm{Dec}_s)$
- **大部分同**态加密可以实现明文乘法和有限次数的密文乘法或密文加法
- **同**时支持不限次数的密文乘法和密文加法的同态加密称为全同态

## 基于RLWE的全同态加密

$$\mathrm{RLWE}_s(\mu) = (a, as + \mu + e)$$

$$\varphi_s((a,b)) = b - as \qquad \varphi_s(\mathrm{RLWE}_s(\mu)) = \mu + e \approx \mu$$

$$ct_{1,2} = \mathrm{RLWE}_s(\mu_{1,2}) = (a_{1,2}, b_{1,2}) = (a_{1,2}, a_{1,2}s + \mu_{1,2} + e_{1,2})$$

$$\mu_1 + \mu_2 \approx \varphi_s(ct_1) + \varphi(ct_2)$$

$$\Rightarrow \mathrm{RLWE}_s(\mu_1 + \mu_2) \approx ct_1 + ct_2$$

$$\begin{aligned}\mu_1\mu_2 &\approx \varphi_s(ct_1)\varphi_s(ct_2) = (b_1 - a_1 s)(b_2 - a_2 s)\\ &= b_1 b_2 - (a_2 b_1 + a_1 b_2)s + a_1 a_2 s^2\\ &= \varphi_s((a_2 b_1 + a_1 b_2, b_1 b_2)) + a_1 a_2 \varphi_s((0, s^2))\end{aligned}$$

$$\Rightarrow \mathrm{RLWE}_s(\mu_1\mu_2) \approx (a_2 b_1 + a_1 b_2, b_1, b_2) + a_1 a_2 \cdot \mathrm{RLWE}_s(s^2)$$

Keyswitch　Keyswitching Key

# 基于RLWE的全同态加密（续）

$$\varphi_s\left(a \cdot \mathrm{RLWE}_s(s^2)\right) = a(s^2 + e) = as^2 + ae$$

- 这样直接的乘法噪声会将Keyswitching Key(KSK)对应RLWE密文里的noise扩大到覆盖明文范围，无法正确解密
- 全同态加密设计了decompose技术，即存在一种线性分解基$\{g_i\}$使得
  - $a = \sum_i a_i g_i$，其中$a_i$是$a$在$\{g_i\}$下表示的系数，并且$\|a_i\| \le B$
- 将Keyswitching Key由单个RLWE密文拓展为一组RLWE密文$\mathrm{RLWE}_s(s^2 g_i)$使得
  - $\varphi_s\left(\sum_i a_i \cdot \mathrm{RLWE}_s(s^2 g_i)\right) = \sum_i a_i\,(s^2 g_i + e_i) = as^2 + \sum_i e_i\, a_i$噪声的大小降低为$\ell B\|e\|$
- 典型的线性分解如二进制分解（digit-decompose）和CRT分解
  - 二进制分解中$g_i = 2^{i\beta}$其中$\beta$代表分解中每一段的比特数
- 部分同态设计中，会在KSK生成中采用不同的噪声采样参数以进一步降低Keyswitch带来的noise

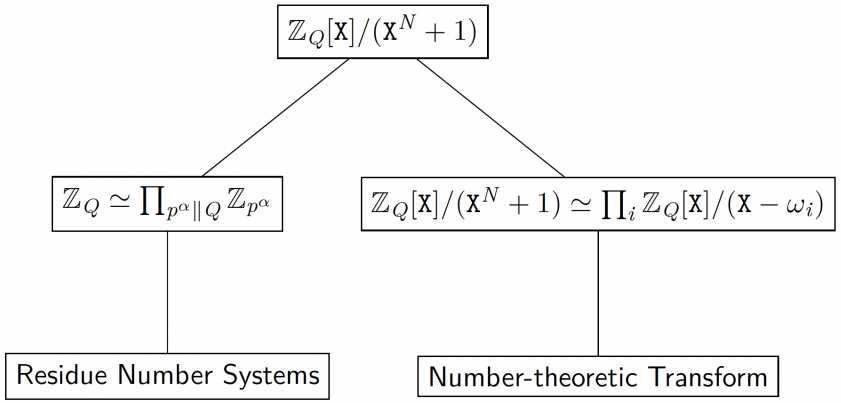
$$\sum_i a_i \cdot \mathrm{RLWE}_s(s^2 g_i)$$

- 基于RLWE的全同态中，Keyswitch这一表达形式是绝大部分软硬件优化的基准点

- 小测验：$\sigma$对FHE所用的环保持同态，即$\sigma(xy) = \sigma(x)\sigma(y)$ $\sigma(x + y) = \sigma(x) + \sigma(y)$，尝试给出从$\mathrm{RLWE}_s(\mu)$得到$\mathrm{RLWE}_s(\sigma(\mu))$的方法以及需要怎样的KSK
- 提示：$\sigma(\mu) = \sigma(\varphi_s((a, b))$
- 当$\sigma$为Galois Automorphism时，这一结果相当于一次GalAuto

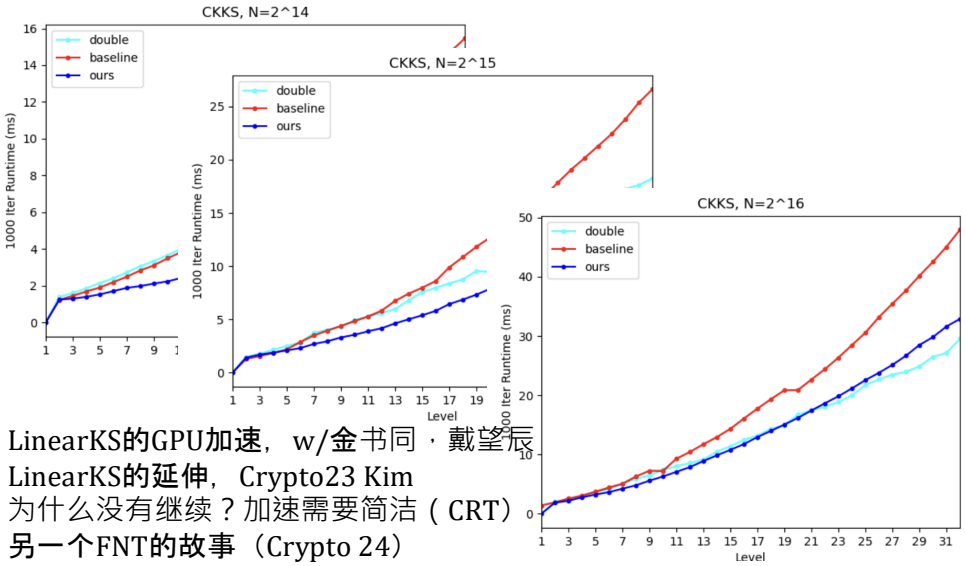
# RLWE的代数结构

The One Ring in FHE (so called "double-CRT")




$$\sum_i a_i \cdot \mathrm{RLWE}_s(s^2 g_i)$$

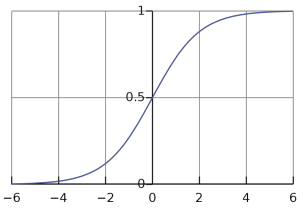
- $a_i$在比较小的环$R_{q_j}$上，而完整密文是在$\prod_j R_{q_j}$上
- 传统的计算方式是将$a_i$先进行环提升(Ring Lifting)到$\prod_j R_{q_j}$上同KSK相乘，需要的NTT次数与Q的大小平方相关
- LinearKS：环提升到一个比$R_{q_j}$大但比$\prod_j R_{q_j}$小的环$R_M$上，使得每一个分量的计算都可以在$R_M$中被表示（$M > q_j^2 N$）
- LinearKS的NTT次数与Q大小线性相关

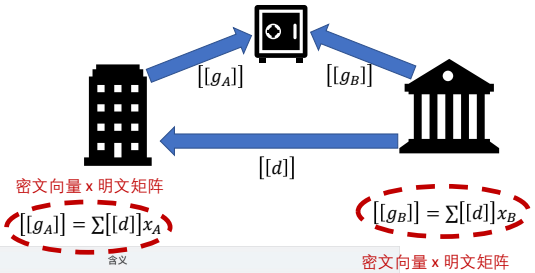



- LinearKS的GPU加速，w/金书同，戴望辰
- LinearKS的延伸，Crypto23 Kim
- 为什么没有继续？加速需要简洁（CRT）
- 另一个FNT的故事（Crypto 24）

# Bootstrapping学习路线

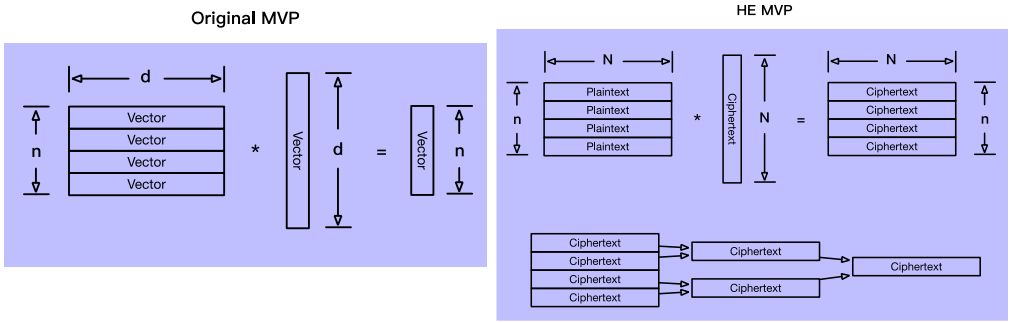
- TFHE：PBS-Keyswitch=BlindRotate=n*CMUX=n*EP=n*(KS+KS)
- CKKS:
  Boot=PolyEval+S2C+C2S=BSGS(PolyEval+MVP)=HomMul+Auto+KS

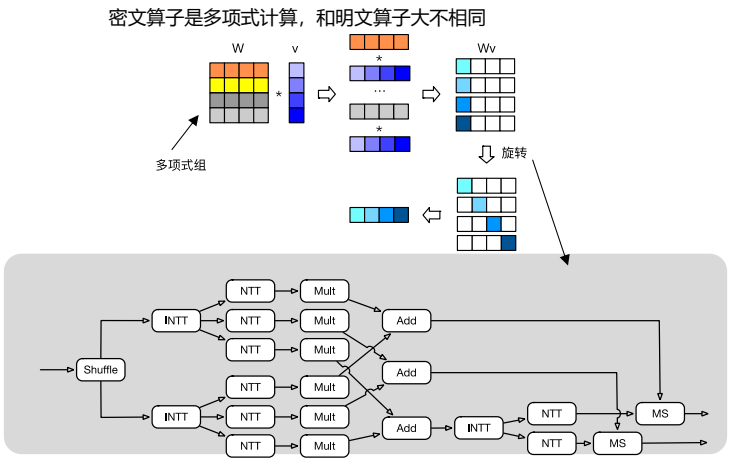




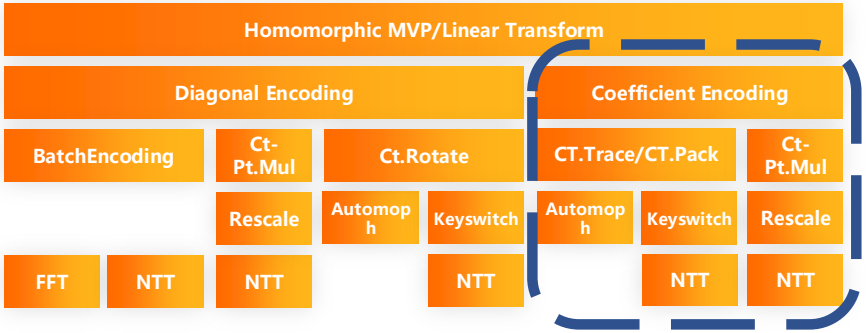

| 符号 | 含义 |
|---|---|
| $[\![\cdot]\!]$ | 同态加密 |
| $w_i^A$ | $w_A x_A$ |
| $w_i^B$ | $w_B x_B$ |
| loss | $log(1+exp(-yw^Tx)) \approx log2 - \frac{1}{2}yw^Tx + \frac{1}{8}(w^Tx)^2$ |
| 梯度g | $(\frac{1}{1+exp(-yw^Tx)} - 1)yx \approx (\frac{1}{2}yw^Tx - 1)\frac{1}{2}yx$ |
| 残差d | $(\frac{1}{1+exp(-yw^Tx)} - 1)y \approx (\frac{1}{2}yw^Tx - 1)\frac{1}{2}y$；即 g=dx |

- 传统的矩阵向量乘 vs. 密文的矩阵向量乘

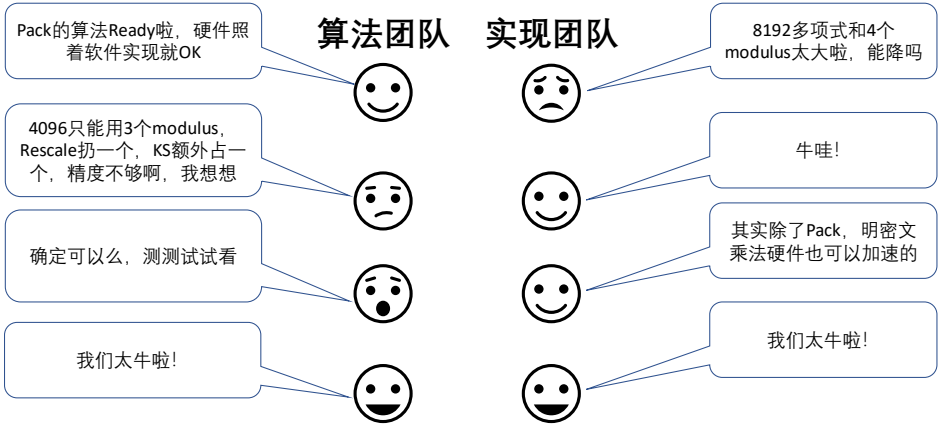




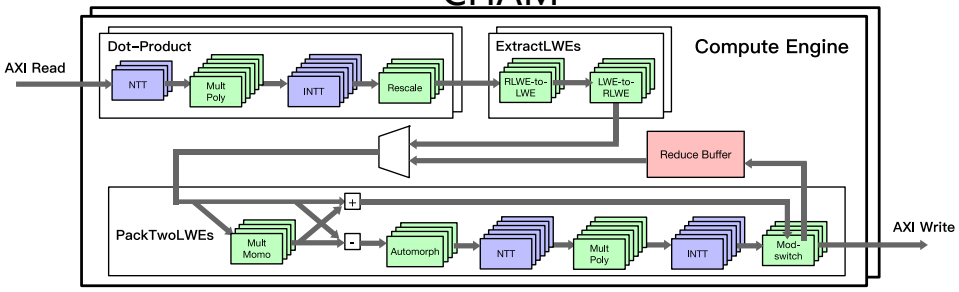

# CHAM: A Customized Homomorphic Encryption Accelerator for Fast Matrix-Vector Product

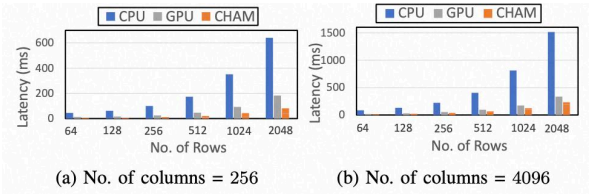


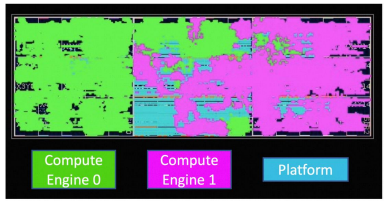

- 应用层：纵向联邦学习提炼关键算子——逻辑回归，对应隐私计算密态矩阵向量乘法
- 协议、算法层：采用Cheetah系数编码配合PackLWE算法适配硬件特性
- 编译层：针对CHAM的编译套件
- 硬件层：针对硬件特性配合算法团队调整算法实现为hardware-friendly

# CHAM



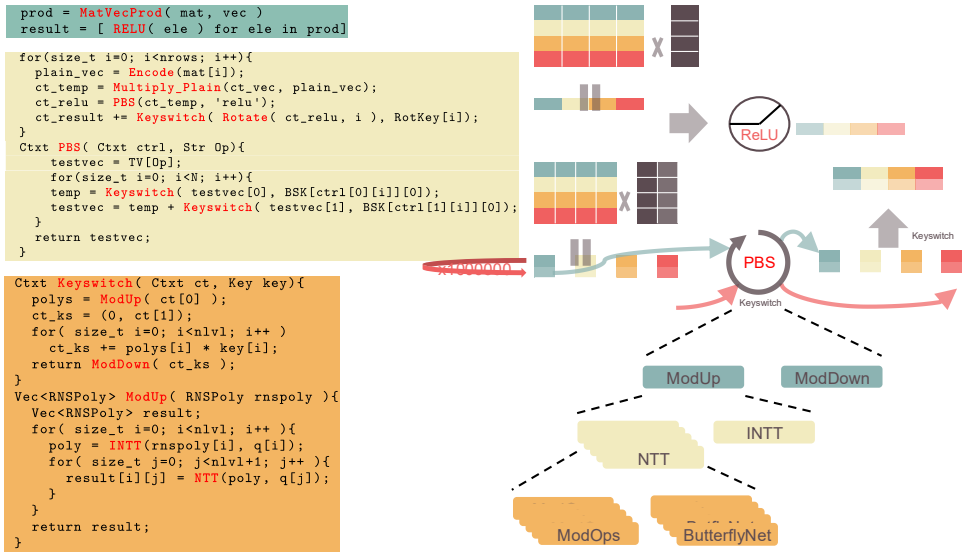

# 隐私计算软硬件加速技术栈


- **算法**倾向存储换计算
- 硬件倾向计算换存储/带宽

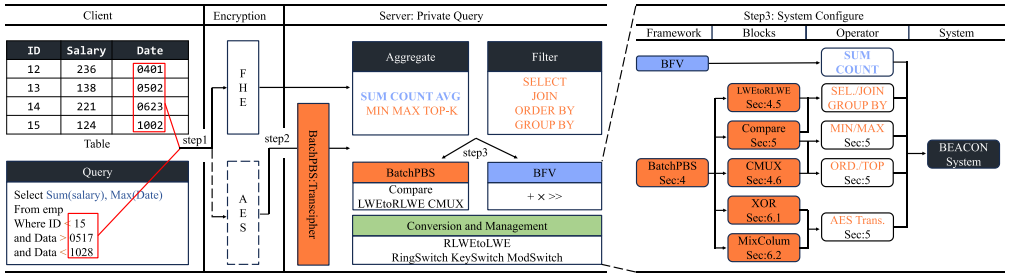
- Crypto23 HERMES PackLWE
- Kim's KS

- **算法到硬件的技**术栈深且耦合紧密，贯通算法到硬件的优化难度巨大（蝴蝶效应）

- HPCA23 TensorFHE

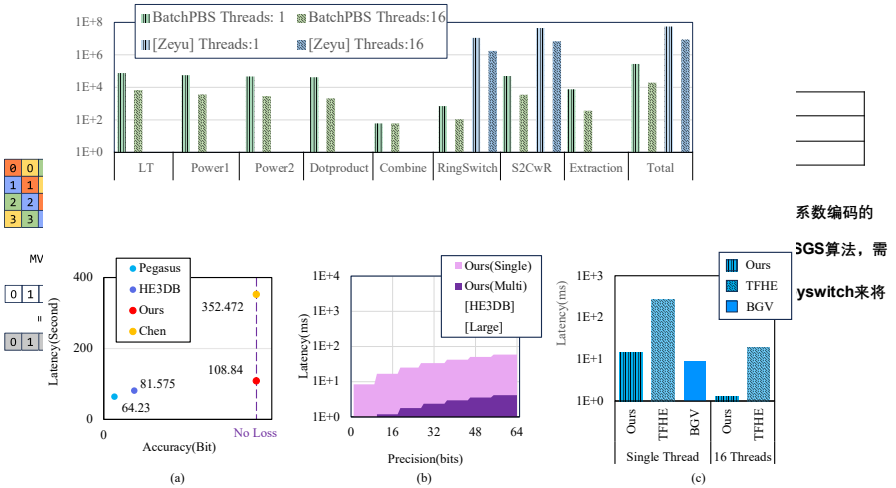



# BatchPBS




- 全同态加密数据库等工作默认数据已经FHE加密过，但用户数据FHE加密上传所需通信成本高、计算量大
- BEACON提供了一种高效的AES密文同态转换FHE密文的方案，并提供高效的批量查询SQL如OrderBy/Select/Join/GroupBy等

# BatchPBS

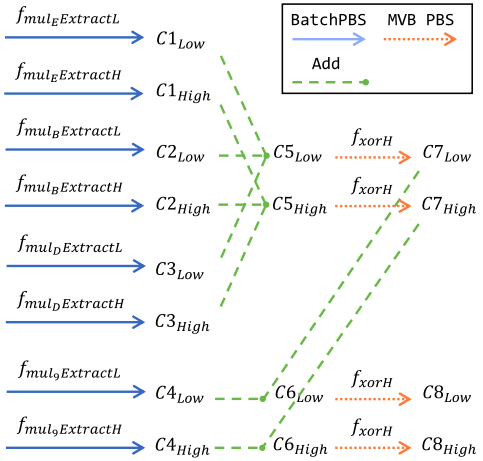


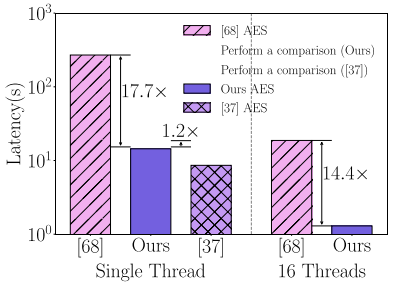

系数编码的

GS算法，需

yswitch来将

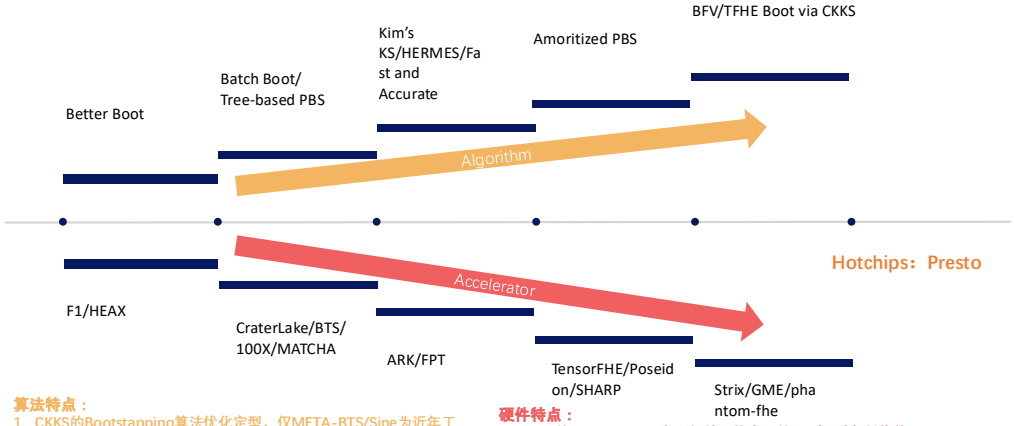
# BatchPBS




- **FHE提供了8b-in/8b-out的任意LUT与ArithOp，如何实现AES?**
- **1b-XOR with 2b-LUT: (0 a)+(0 b)=(a&b a^b)->(0 a^b)**
- **S-box可以完整fit到LUT里**

## 未来：软硬件优化趋势




**算法特点：**

1. CKKS的Bootstapping算法优化定型，仅META-BTS/Sine为近年工作（更高精度）。CKKS与TFHE开始汇流
2. **TFHE分支主要**发展为两方面，**LUT的高利用率（Tree-based/MVB）与PBS的批**处理（BatchBoot/AmorPBS）

**硬件特点：**

1. CKKS的Bootstapping实现仍然是热点，从"黑盒"到定制裁剪（SHARP/FPT）
2. ASIC持续产出、GPU劲头强势、FPGA零星产出