Agentic AI 重塑计算架构: 从 GPU-centric 到 CPU-GPU 协同

AI算力范式跃迁

1990s — 2012
传统机器学习时代

CPU为中心

SVM | Random Forest | Logistic Regression

CHARACTERISTICS

- 低并行度
- 小数据集(MB~GB)
- 手动特征构建

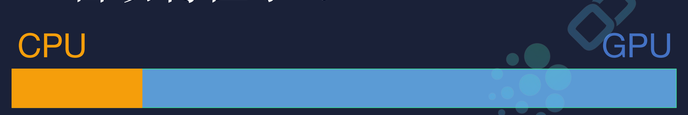
2012 — 2025
深度学习时代

GPU为中心

CNN | RNN | LLM | Transformer

CHARACTERISTICS

- 海量并行
- 大数据集(TB Scale)
- 自动特征学习



2025 — Now
Agentic AI 时代

CPU-GPU 协同

ReAct | Skills | MCP | Harness Engineering

CHARACTERISTICS

- 推理, 计划 & 工具
- 动态记忆管理
- 多轮对话，长程任务

AI Agent的工程演进

Phase · 01

Prompt Engineering

单轮请求–响应

- 指令设计
- Few-shot 示例
- 输出格式化
- 手动迭代

Human–In–Loop

CPU 利用率

→

Phase · 02

Context Engineering

RAG + 历史记录压缩

- RAG 流水线
- 历史记录总结
- 上下文组装
- Embedding 管理

Human–In–Loop

CPU 利用率

→

Phase · 03

Harness Engineering

工具脚手架+ API路由

- 工具注册 & 路由
- 权限管理
- 错误处理
- 结果解析

Human–In–Loop

CPU 利用率

→

Phase · 04

Loop Engineering

Plan>Exe>Obs>Ref>Replan

- 目标拆解
- 执行监控
- 自我反思
- 自适应重计划

Human–In–Loop

CPU 利用率

将人的因素从任务流中移出→ CPU利用率的提升.

AI Agent的记忆管理

短时记忆
Transient Memory

滑动提示词窗口

- 对话结束即消失，无法跨会话保留
- 记忆=Prompt 本身

CPU 操作：

- Token 计数
- 数组切分

→

关联记忆
Associative Memory

RAG & 向量检索

- 跨会话持久化，但检索结果具有概率性
- 记忆=Context 输出

CPU 操作：

- Embedding 查找
- 基于相似度的搜索
- 重排
- 上下文组装

→

结构化记忆
Structured Memory

知识图谱

- 以实体-关系-状态的图结构存储，支持精确查询
- 记忆=受治理的知识资产

CPU 操作：

- 图遍历
- 记忆实体提取& 连接
- 关系更新
- 选择性上下文组装

→

反思性记忆
Reflective Memory

动态自优化

- 从经验中提炼规则，主动修正行为模式
- 记忆= Loop的代谢物

CPU 操作：

- 记忆加载 & 移除
- 语义压缩
- 记忆合并
- 优先级调度

记忆从“感知当下”升维为“自我进化”，GPU 推理效率提升，CPU 记忆管理复杂度也随之增加。

主-从 AI Agent

Master Agent

有状态 · 长时

- Residency: 长周期，持久化状态(小时/天)
- Context: 长上下文窗口(100K+ tokens)
- Role: 任务拆解和协调
- Memory: 结构性或反思性的记忆
- Aggregation: 收集并综合sub-agent结果

Sub Agent

无状态 · 瞬时

- Residency: 短周期 (秒/分)
- Context: 聚焦, 最小化上下文(具体任务导向)
- Role: 单一原子任务执行
- Memory: 瞬时，任务完成后清除
- Output: 结果返回给Master后，终止

有状态和无状态负载混布： Cache thrashing · TLB thrashing
调度器必须感知 agent 生命周期与上下文亲和性。

Agent Sandbox 超卖（Oversubscription）

CPU空闲来源

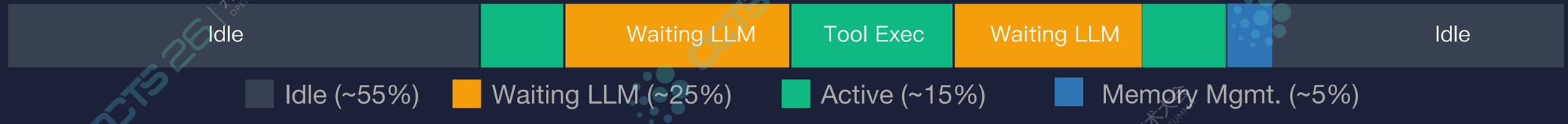
宏观闲置: 定时/人类触发的任务

- Cron jobs 或等待人类命令输入
- 期间CPU处于闲置状态.

微观闲置: 等待大模型响应

- 200ms – 2s per LLM call.
- 等待大模型响应时，CPU处于闲置状态

Agent 行为时间线示例 (24h view)



vCPU 超卖

- Statistical multiplexing, vCPU时间片轮转调度
- 超卖天然可行
- 4x ~ 5x 超卖，受限于内存容量.
- 同时应对宏观闲置和微观闲置

超卖的影响

- 大幅提升CPU利用率
- GPU集群推理吞吐压力提升

GPU 推理优化

Batching

Static vs Continuous

- 把固定 micro-batch 换成滚动调度槽。
- 请求完成立即释放槽位,新请求中途补入。

Static Batching

Continuous Batching

Specialization

Prefill-Decode · Attn-FFN分离

- Prefill 偏算、Decode 偏存;
- Attention 偏存、FFN 偏算,
- 算性各异,分池流水。

Caching

KV Cache

- 新请求命中已缓存前缀,直接跳过 prefill 计算。
- 用存储换算力。

三大方向协同作用，共同减少 CPU 等待 LLM 回复的时间

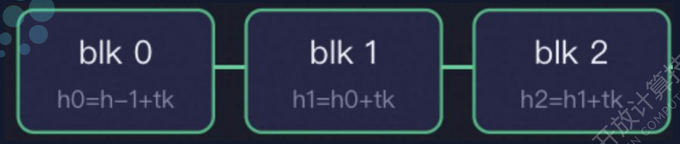
KV Caching优化

命中率

重用前缀 KV 计算

1 · vLLM

Block Hash + PagedAttention



H_K = Hash(H_{K-1}+ tokens)

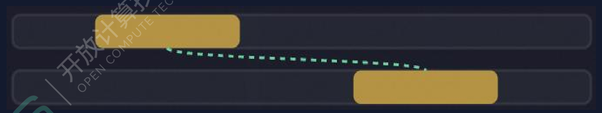
2 · SGLang

RadixAttention

树以 token 序列为路径，共享前缀的请求自然落在同一条分支。

3 · vLLM (EPIC)

Position-Independent Caching (ICML 2025)



脱离 token 位置依赖，同一语义片段无论出现在哪儿都可复用 KV。

存储效率

相同容量，更多 KV

1 · KV 量化

- TurboQuant/RabitQ: FP16 → 4-bit/2-bit
- 精度损失受控，KV 体积下降约75%，命中同样多上下文。

2 · KV 压缩

- DeepSeek V4： CSA / HCA · KV 压缩
- CSA: 每 m =4个 token 的 KV cache 压缩为一个条目
- HCA: 每 m’=128 个token的 KV cache 合并为单个条目
- ~10X KV 压缩比

Engram: CPU-GPU协同算力典范

无 ENGRAM

Agentic CPU
工具执行，记忆管理

GPU机头CPU
前/后处理，内核启动

GPU
LLM 推理

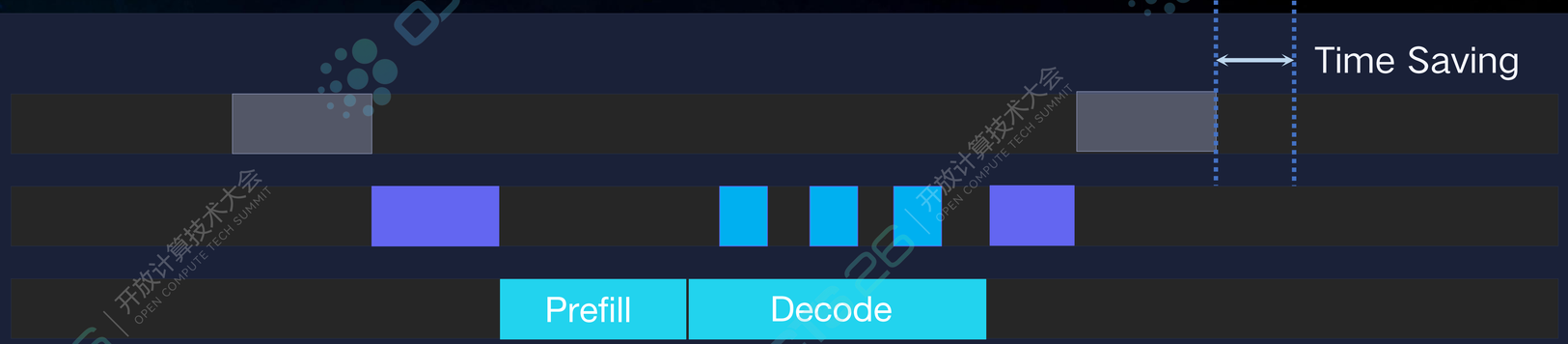
所有知识都从大模型内部提取— 每个事实都消耗译码.

采用 ENGRAM

Agentic CPU
工具执行，记忆管理

GPU机头CPU
前/后处理, 内核启动, Engram

GPU
LLM 推理



Engram 利用CPU查询静态知识—LLM仅在静态知识上做推理.

阿里云磐久UMX

Unified Memory Extension · AI 总线级存算互连架构

互连芯片+介质创新+软件系统

Scale-Up 思路

基于计算总线构建超节点内存/存储池

原生 AI 架构

GPU/CPU超节点解决容量+成本+性能挑战

协议基础

UALink + CXL 计算总线； Load/Store 内存语义

极致性能

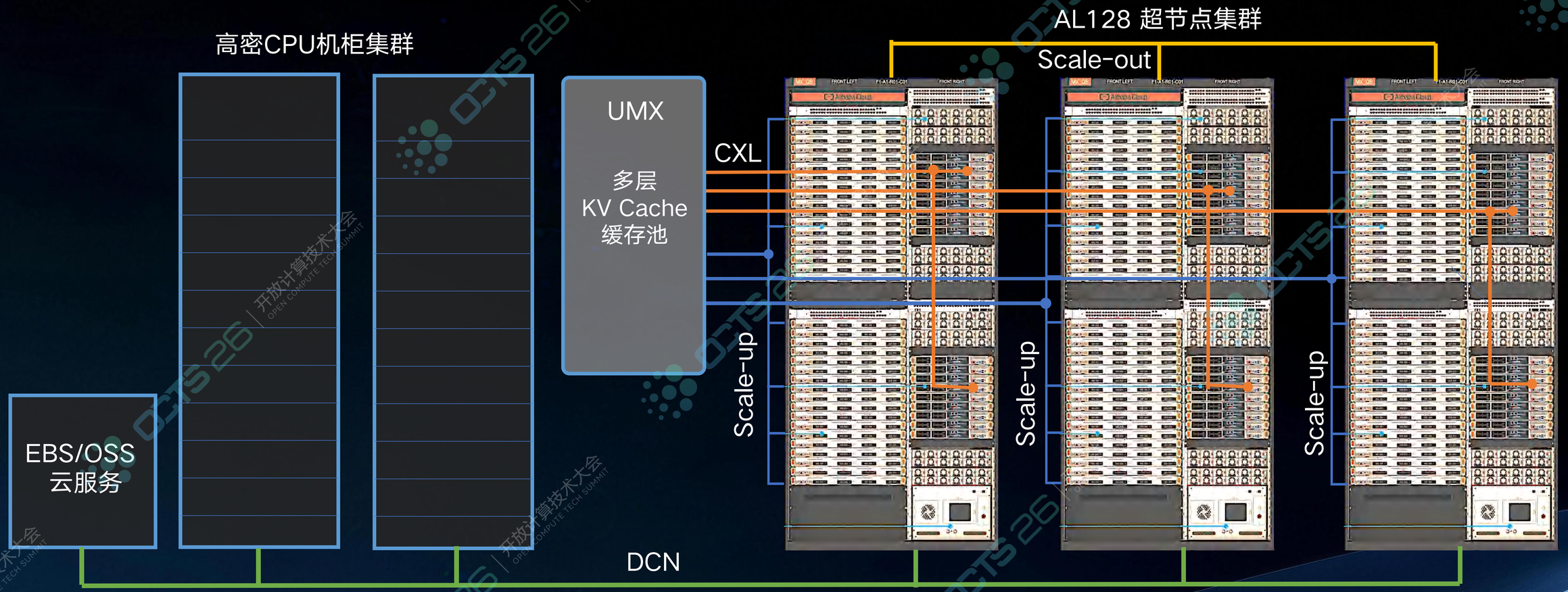
高带宽、低延迟；消除存储墙

目标场景

极速AI推理 · Agent记忆；超长KV Cache

通过超节点 Scale Up 高带宽互连，将内存 / 存储池以纳秒级延迟统一接入算力节点，消除 AI Agent 时代的“存储墙”，实现极致性能与存算深度耦合。

AI Infra. for Agentic AI



全栈算力异构，CPU- GPU 数量之比接近 1:1

TAKEAWAY

Agentic 时代的计算是协作共生的。

智能不再是只运行在 GPU 上的一种工作负载——它是构建在CPU、GPU、多级内存和互连架构的协同工作之上的综合呈现。

THANKS