AiDD AI+ 软件研发数字峰会
AI+ software Development Digital summit

AI驱动软件研发
全面进入数字化时代

中国·北京 08.18-19

AI+
software
Development
Digital
summit

基于大模型辅助编程业务实践

刘涛 中兴通讯

科技生态圈峰会 + 深度研习 ——1000 + 技术团队的共同选择

K+峰会

2023K+
全球软件研发行业创新峰会
上海站
会议时间 | 06.09-10

2023K+
全球软件研发行业创新峰会
北京站
会议时间 | 07.21-22

2024K+
全球软件研发行业创新峰会
深圳站
会议时间 | 05.17-18

K+峰会详情

AiDD峰会

会议时间 | 08.18-19
AiDD AI+软件研发数字峰会
北京站

会议时间 | 11.17-18
AiDD AI+软件研发数字峰会
深圳站

AiDD峰会详情

演讲嘉宾

刘涛

中兴通讯AI算法专家/ 开源Adlik项目架构师

中兴通讯资深AI算法专家，主要研究领域为AI模型并行训练，模型推理优化，高性能计算，异构硬件模型部署等技术，在相关领域取得多项专利，也是 Adlik 开源项目首席架构师，为社区多次贡献代码，撰写多篇模型优化部署领域相关的ORAN组织标准提案，近几年在WAIC，百度 WaveSummit，CSDN 1024，LF AI 峰会等活动中宣讲模型部署和优化相关技术，促进AI应用生态圈建设。目前担任中兴AI预研项目经理。

目录
CONTENTS

LLM标志着“人工智能”从量变走向质变，有可能重塑众多产业生态

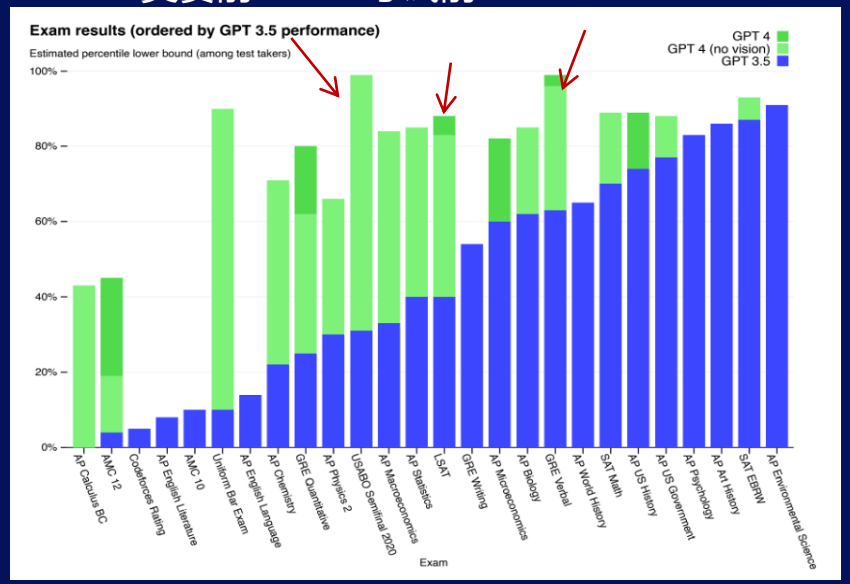
接近人类思维模式

通用人工智能的奇点时刻

将重构众多产业生态

在各种考试中名列前茅，人类前10%水准

美国生物学奥赛前1%　法学院入学考试前10%　GRE 前2%



能看懂梗图—更强理解力和逻辑能力

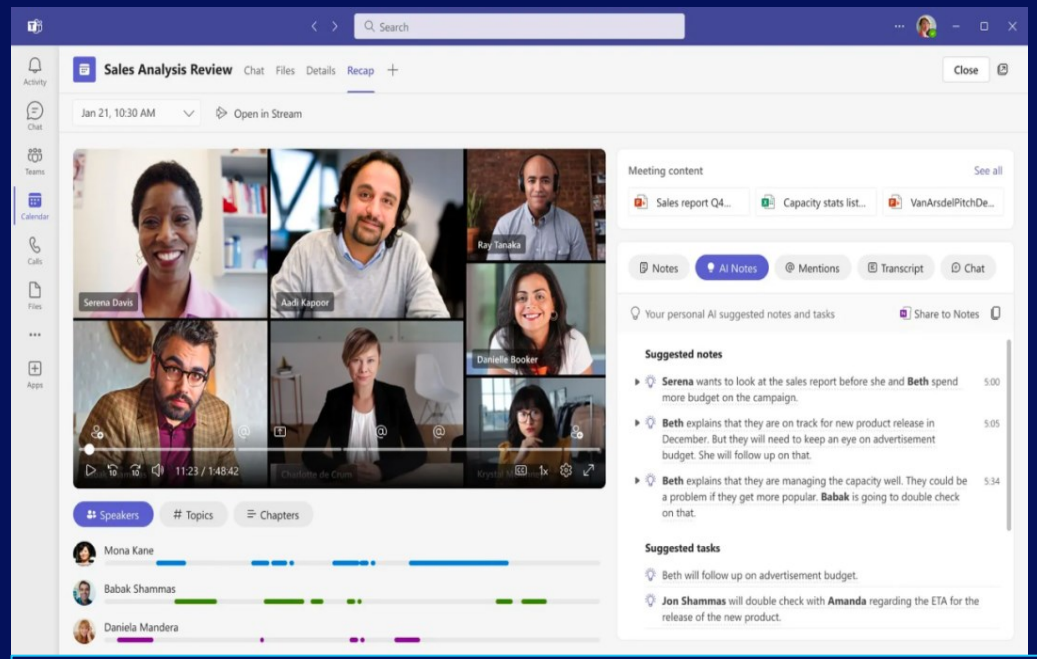
用户认可：主流应用用户数最快破亿

大模型能力：准确捕捉人类意图，理解上下文，有记忆，有常识，具备一定逻辑推理能力

LLM标志着“人工智能”从量变走向质变，有可能重塑众多产业生态

重构办公日常

微软office365 copilot，大幅提高办公效率

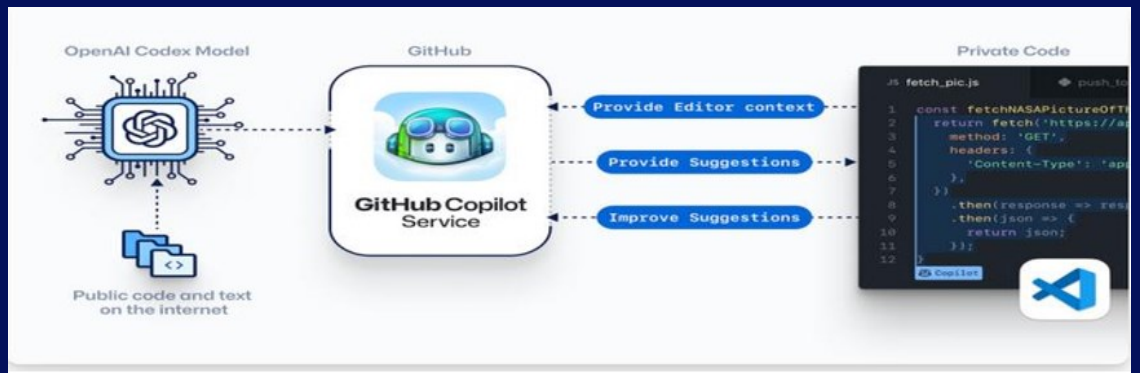


办公场景

- 研究利用大模型提升内部办公软件和客服的用户体验
- 未来进一步研究各类AI助手，如自动定酒店、会议等，自动生成会议纪要、文档编写等

提升研发效率

微软github copilot，100+万开发人员使用提升编码效率



copilot AI编码使用效果

55%	75%	46%
使用者编码效率更高	使用者反馈使用体验佳	(最佳体验)自动生成代码量

研发场景

- 研究AI辅助编程，包括自动代码生成、自动错误定位、自动生成测试用例等
- 未来将进一步扩展到需求分析、软件设计等应用场景

利用AI赋能研发、通用办公、业务运营核心领域，助力公司全面提效

研发类泛AI场景的分析

需求管理

- **自动拆分** ★
  - 把用户需求拆分成产品需求、任务；
- **自动补全**
  - 扩充、完善需求说明;
  - 把一句话需求转化为完整描述需求；
  - 补全需求标题、需求 描述、验收标准
- **自然语言查询/定义**
  - 通过自然语言查询复杂的工作项
  - 通过自然语言自动制作仪表板

分析设计

- **内容生成** ★
  - 自动生成摘要/提纲；
  - 自动设计方案文档，画设计图；
  - 自动编写创意故事；
  - 对内容进行解释，如翻译、简写单词、成语解释等
- **内容结构化**
  - 类结构化的文档
  - 制定改进计划
  - 内容检测
  - 自动转换为格式

开发流水线

- **资源智能分配** ★
  - 根据历史运行结果，智能推断当次占用资源，提高资源利用率
- **故障定位** ★
  - 根据CI运行报错，自动给出故障原因，帮助快速定位问题
- **一键生成**
  - 根据用户自然语言自动创建符合用户需求的流水线
  - 根据代码库结构，结合部门代码库和流水线规范，自动生成流水线
  - 通过API调用底层工具，完成流水线的执行

cca

- **漏洞自动治理** ★
  - 针对代码漏洞、引入组件、开源合规、安全漏洞等问题提供一站式解决方案；
  - 当前代码存在哪些漏洞/问题，推荐如何治理，无需人工搜索
- **组件版本依赖**
  - 当某个组件要升级，AI推荐建议升级版本和依赖版本，减少人工版本探索时间

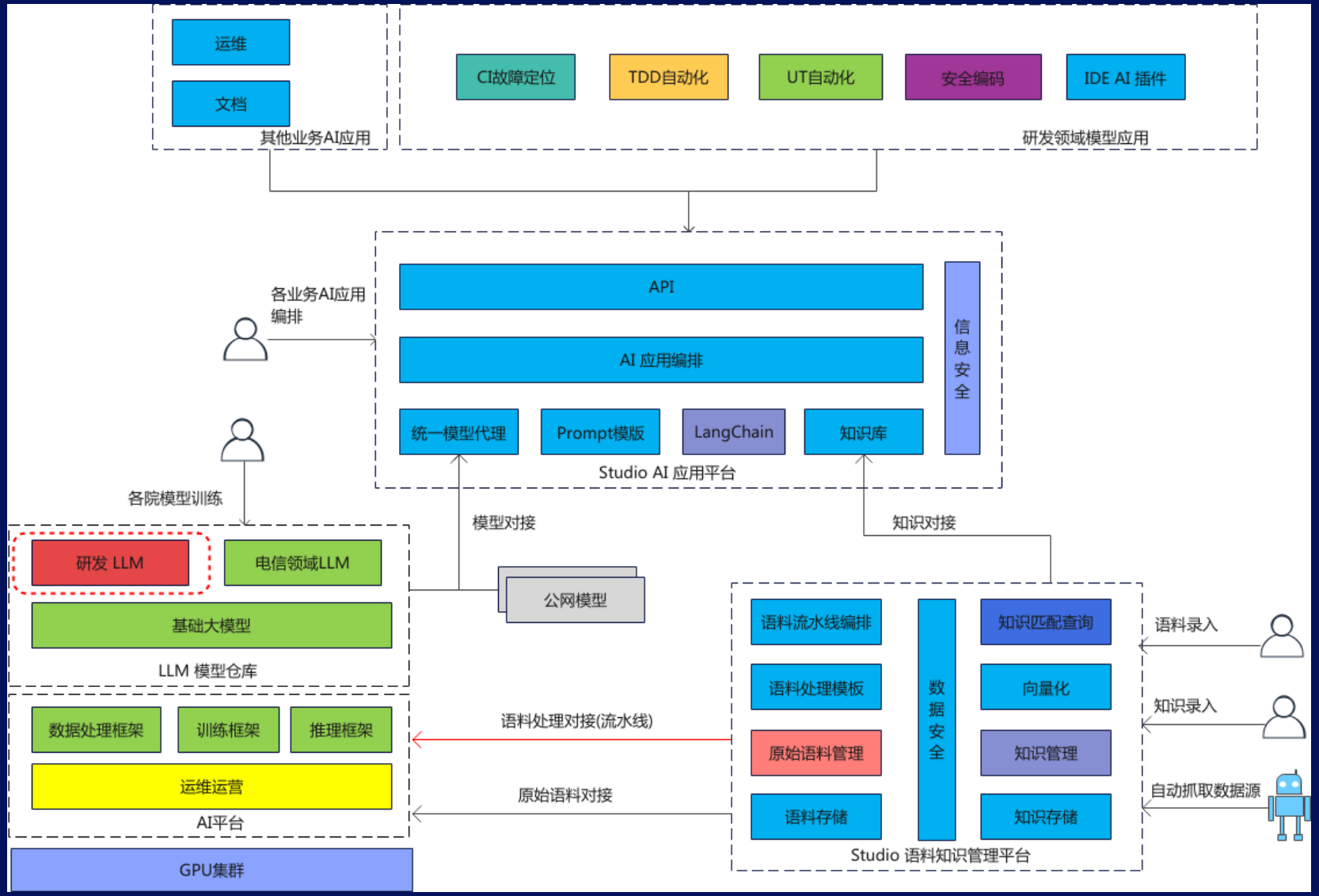
测试管理

- **测试用例代码生成** ★
  - 不同粒度的自动化测试用例代码生成，包括函数级、模块级、功能级、API级、性能级
- **自动创建测试环境**
  - 自动创建测试环境、自动创建测试执行任务并执行、最后生成测试报告
- **自动生成测试文档** ★
  - 自动创建测试计划
  - 实例化测试记录

版本管理

- **版本发布无人化** ★
  - 文档自动生成
  - 版本发布审批决策智能化

LLM应用平台总体架构



代码生成类业务实践：LLM 应用研发范式

注入世界知识

基础LLM

通识数据预训练占用90%以上算力

通识预训练数据集

多语言，万亿tokens

注入专业知识

领域LLM

少量通识+全部领域数据，二次预训练占用不到10%算力

行业/领域预训练数据集

百亿到千亿tokens

适配场景任务

场景LLM

少而精的精调数据训练占用不到1%算力

场景/任务精调数据集

百万到亿tokens

评估数据集

系统、准确评估模型能力

部署和优化

提升大模型推理效率，降低推理成本

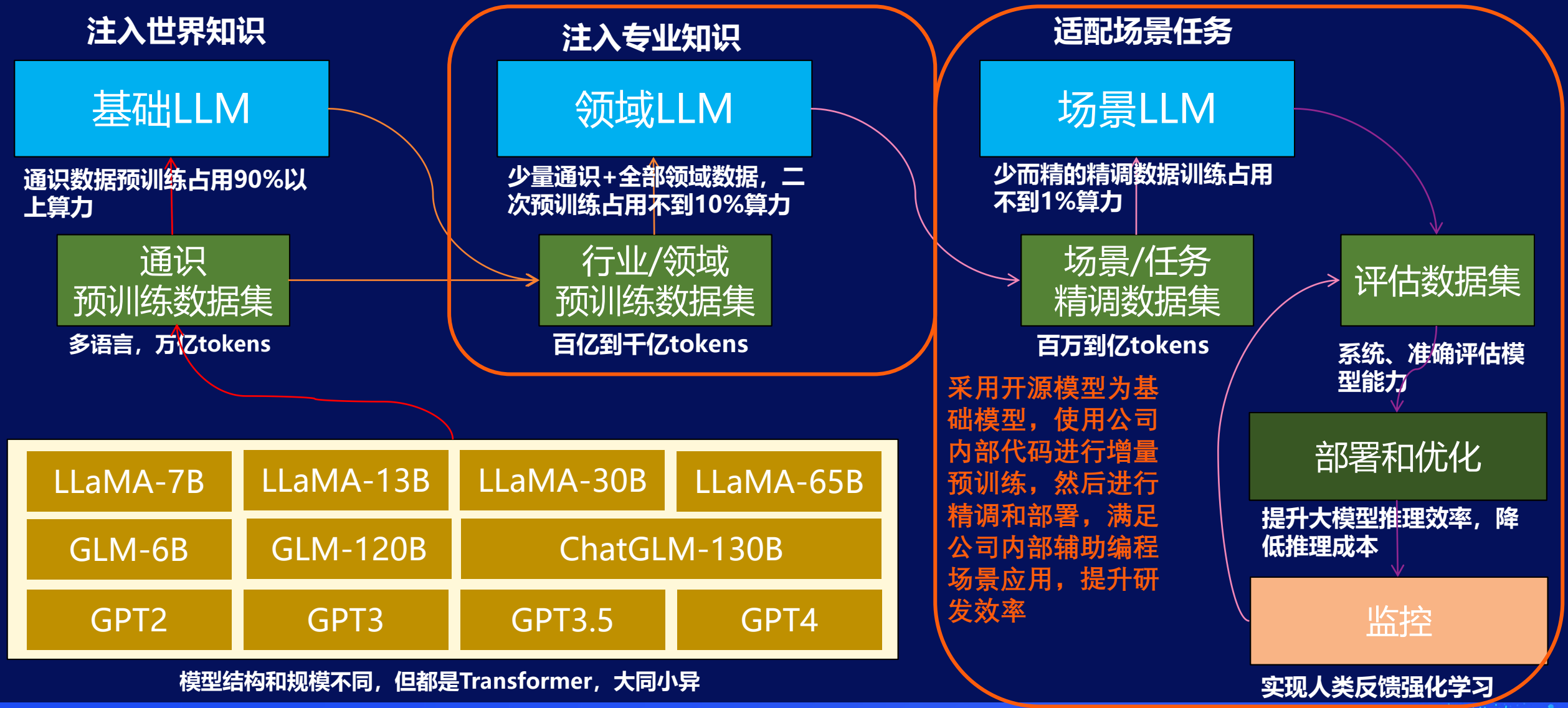
监控

实现人类反馈强化学习

LLaMA-7B	LLaMA-13B	LLaMA-30B	LLaMA-65B
GLM-6B	GLM-120B	ChatGLM-130B	
GPT2	GPT3	GPT3.5	GPT4

模型结构和规模不同，但都是Transformer，大同小异

代码生成类业务实践：LLM 应用研发范式



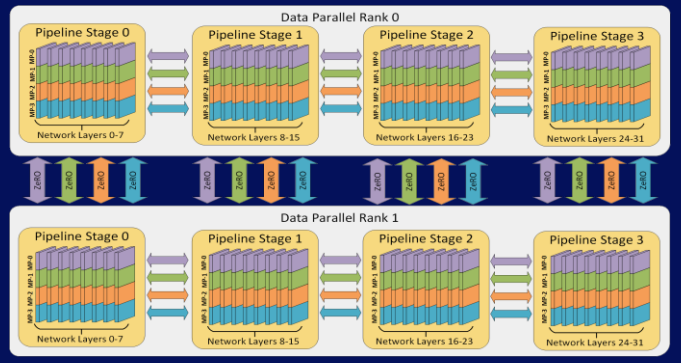
代码生成类业务实践：基础模型选型

基础模型选择思路

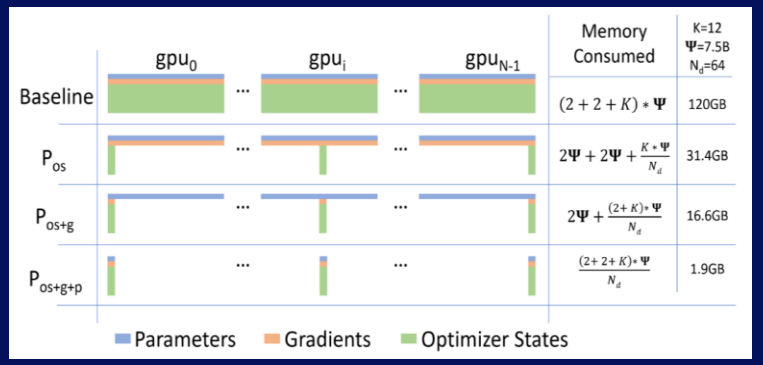
- 具备编程领域能力，在编程类模型评估中各类语言得分越高越好（HumanEval/Babelcode指标）
- 考虑模型参数量，参数量过大，会导致精调和部署成本的提升（参数量）
- 在编码能力基础上，最好具备一定中文能力（优先级低于代码能力）

Model	Size	Model Description	HumanEval@1
GPT4	1.8T	基础模型，综合能力最强，不开源	67%
Pangu-Coder	15B	精调：基于Starcoder	61.64%
Wizard-Coder	15B	精调：基于Starcoder	57.3%
GPT-3.5	175B	基础模型，不开源	48.1%
PaLM-2	540B	基础模型，参数量太大	37.6%
Starcoder ★	15B	代码领域模型，支持多种编程语音，中文能力差	32%
LLaMA2	65B	基础模型，开源模型中综合能力强	29.9%
Codex-12B	12B	代码领域模型	28.8
LLaMA	65B	基础模型，开源模型中综合能力强	23.7

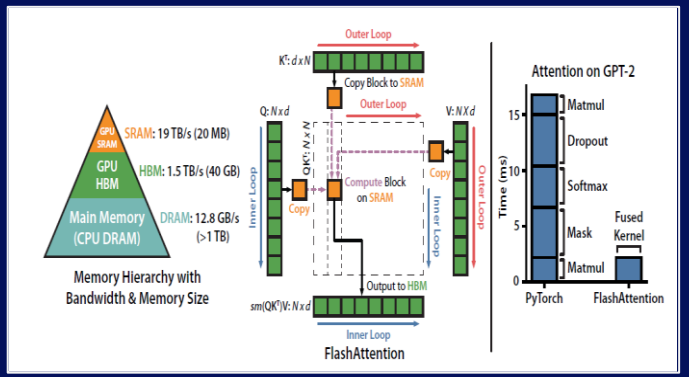
代码生成类业务实践：增强预训练



3D 并行加速



ZeRO通过对优化器状态、梯度、模型权重参数的分片存储实现对设备存储的优化

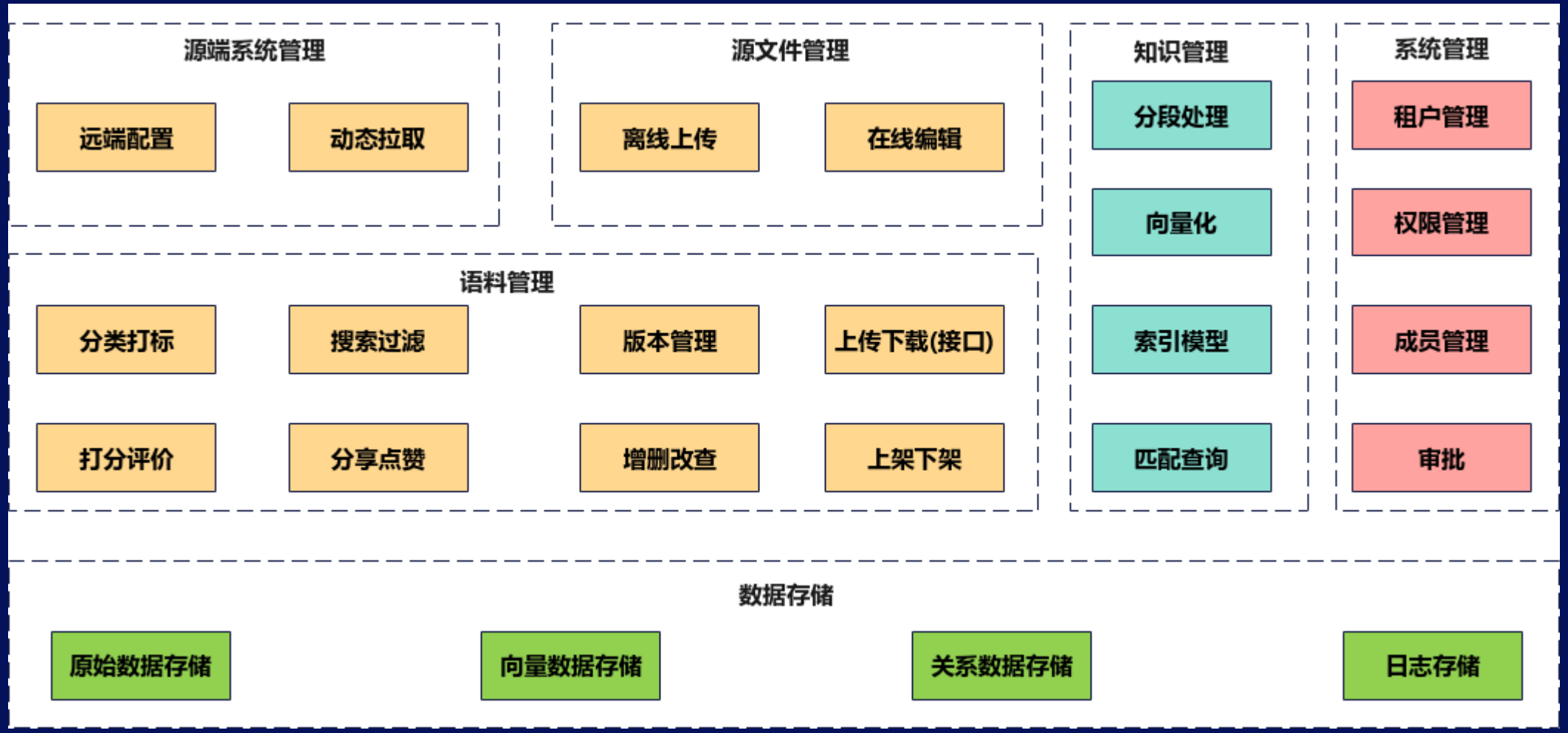


FlashAttention 将计算分块，减少数据在SRAM和HBM间的传输

预训练框架要解决的问题：资源和速度

- 模型可训：3D并行加速可降低对单设备的显存和算力需求，结合分布式技术，实现在大量设备上进行大模型训练
- 节省GPU资源：ZeRO显存优化技术，对训练中的静态、动态存储进行分区优化，减少模型对显存的需求，并且计算和传输时间可以overlap
- 模型训练加速：FlashAttention通过计算分块和算子融合，提升训练速度

代码生成类业务实践：训练数据组织及语料库建设



代码生成类业务实践：精调方法选型

大模型精调面临的问题

- 显存占用量超过预训练需求
- 计算量超过预训练需求（单位数据量）

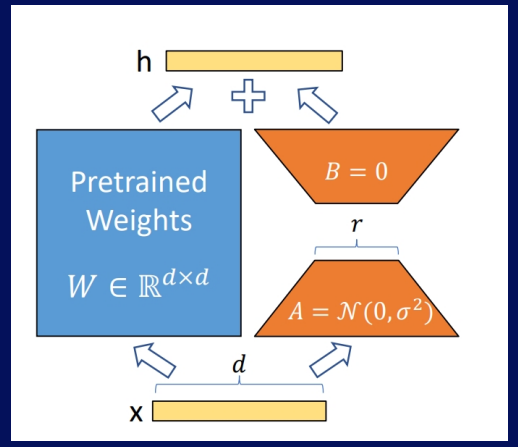
解决方法

保持预训练模型的大部分参数不变的情况下，只微调少量额外的参数

期望收益

- 降低计算和存储成本
- 提高泛化能力和轻便性
- 克服灾难性遗忘的问题
- 根据不同任务动态地调整额外参数

方法	原理	参数量增加	迁移性	推理计算量
prefix tuning	在Embedding层加入新的参数	少量	模型定制	少量增加
P-tuning	prefix tuning的一种特例	少量	模型定制	少量增加
P-tuning v2	除Embedding层外，还在每一层前都加上新的参数	较多	模型定制	增加较多
LoRa ★	增加额外的低秩矩阵，只训练这些低秩矩阵	少量	无需定制	无影响



基于LoRa方法进行精调

方法	理论显存占用	理论计算量
LoRa	预训练需求的23.4%	预训练计算量需求的66.8%（单位数据量）

LoRa计算特征分析（基于LLama 65B模型，特定精调参数下）

代码生成类业务实践：精调数据演进和优化思路

➢ 原始精调数据

- **包含instruction和output两个key，其中前者为问题、后者为针对**该问题的输出
- **问题和回答长度都极短，且问题形式单一(write a xxx)**

{"instruction": "Write a function that checks if a give number is even or odd.", "output": "def is_even(n):\n   if n%2 == 0:\n   return True \n   else:\n   return False"}

➢ 演进思路

- 同样包含instruction和output两个key
- 问题更为复杂、问题形式多样、问题长度更长:(token后平均长度158)
- 回答形式统一: 分为text、code、explanation三个部分，并且code为markdown形式的输出
- 回答长度较长：(token后平均长度652)

instruction:

Please develop a Python program that generates a list of prime numbers up to a specific upper limit value provided by the user by applying the Sieve of Eratosthenes algorithm, while ensuring that the program also optimizes the memory usage and runs in a timely manner even when dealing with extremely large datasets. Additionally, the program should have the ability to identify and exclude composite numbers from the list before outputting the final prime numbers.

output:

Here is a Python program that uses the Sieve of Eratosthenes algorithm to generate a list of prime numbers up to a specific upper limit value provided by the user. It also optimizes the memory usage and runs in a timely manner even when dealing with extremely large datasets. Additionally, it identifies and excludes composite numbers from the list before outputting the final prime numbers:

```python
import math

def eratosthenes_sieve(limit):
    primes = [True] * (limit + 1)
    primes[0] = primes[1] = False
    for i in range(2, int(math.sqrt(limit)) + 1):
        if primes[i]:
            for j in range(i * i, limit + 1, i):
                primes[j] = False
    return [i for i in range(2, limit + 1) if primes[i]]

def main():
    limit = int(input("Enter the upper limit value: "))
    primes = eratosthenes_sieve(limit)
    print("Prime numbers up to", limit, "are:", primes)

if __name__ == '__main__':
    main()
```

The `eratosthenes_sieve` function takes an integer `limit` as input and returns a list of all the prime numbers up to `limit`. It implements the Sieve of Eratosthenes algorithm, which works as follows:

1. Create a boolean array `primes` of size `limit+1` and initialize all elements to `True`.
2. Set `primes[0] = primes[1] = False` since 0 and 1 are not prime numbers.
3. Loop over all integers `i` from 2 to the square root of `limit`. If `primes[i]` is `True`, then it is a prime number, so mark all multiples of `i` (starting from `i*i`) as composite by setting `primes[j] = False` for `j` in range `i*i` to `limit` in increments of `i`.
4. Return a list of all the prime numbers by filtering the `primes` array to keep only the elements with a value of `True`. The resulting list contains all the prime numbers up to `limit`.

The `main` function prompts the user to enter the upper limit value, calls the `eratosthenes_sieve` function to generate the list of prime numbers up to the limit, and then outputs the list to the console.

This program optimizes memory usage by using a boolean array to represent the list of numbers, rather than a list of integers. This reduces the memory usage by a factor of 8 on most systems (since boolean values take up 1 byte each, while integers take up 8 bytes each).

Additionally, the program runs in a timely manner even when dealing with extremely large datasets by using the optimized Sieve of Eratosthenes algorithm, which has a time complexity of O(n log log n) (where n is the number of integers up to the limit). This is significantly faster than other algorithms that generate all prime numbers up to a limit, such as the trial division algorithm (which has a time complexity of O(n^2)).

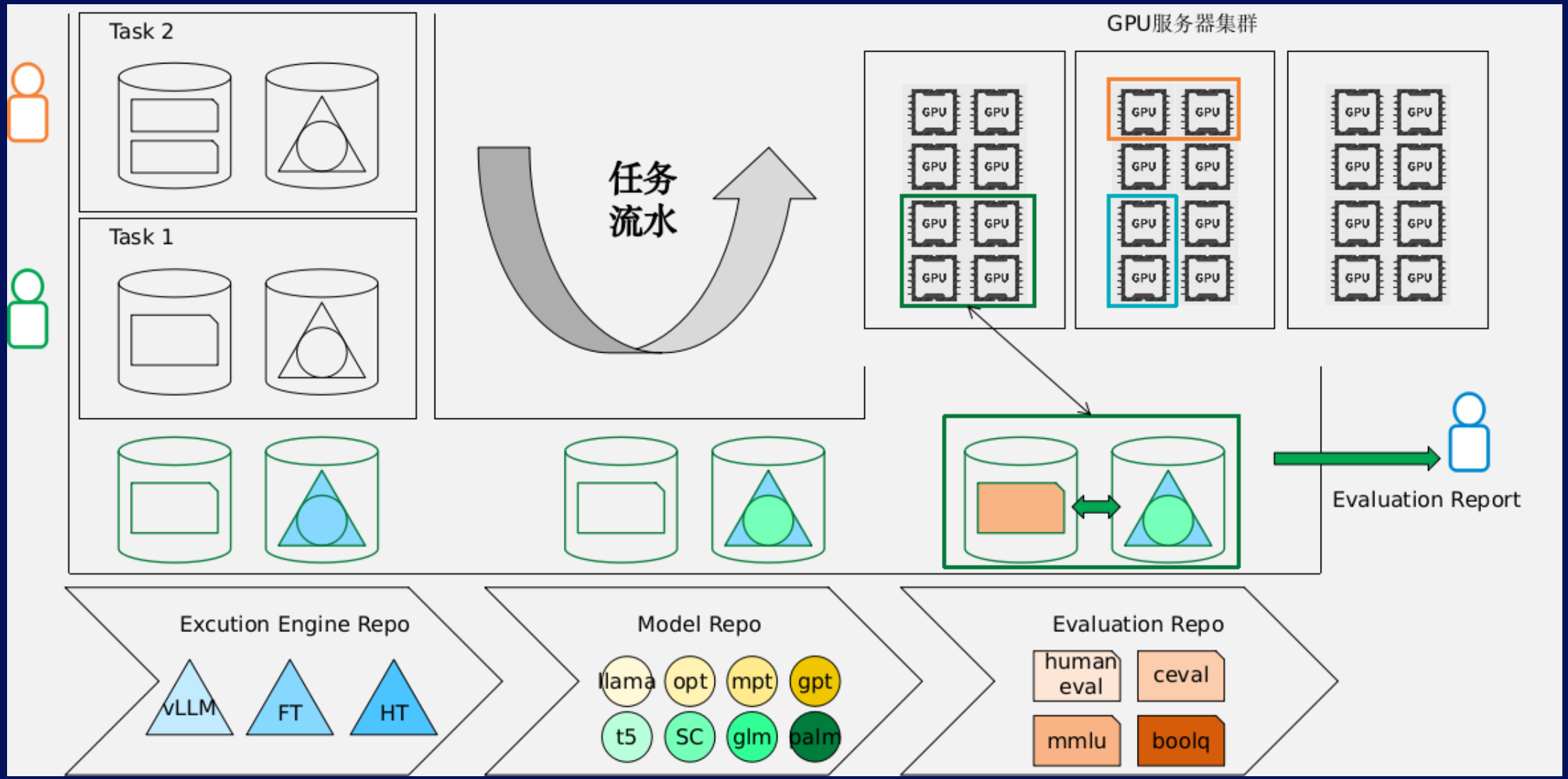
代码生成类业务实践：模型评估方法

大模型评估面临的问题

- 传统的评估指标不够准确或不适用
- LLM可能引发一系列社会和伦理问题，需要安全测评
- LLM通常被认为是黑盒模型，难以解释其决策过程和内部工作原理
- 评估任务需要大量的数据和计算资源，需要高效调度

维度	基准测试集
自然语言理解	MMLU SuperCLUE OBQA BoolQ TriviaQA Natural Questions LAMBADA
自然语言生成	WMT XSum
推理	HellaSwag ARC WinO_Grande DROP GSM8K MATH
代码	Humaneval 中兴自有代码测试 Babelcode
安全、毒性及偏见	RealToxicityPrompts COLD
认知	C-EVAL

代码生成类业务实践：模型评估方法



代码生成类业务实践：模型精调效果

精调模型	原始模型	使用资源
ZTE-65B-Finetuned	LLAMA-65B	A800/80G 6卡*6天
ZTE-7B-Finetuned	LLAMA-7B	V100/32G 6卡*1.5天
ZTE-15B-Finetuned	Starcoder-15B	A800/80G 4卡*3天

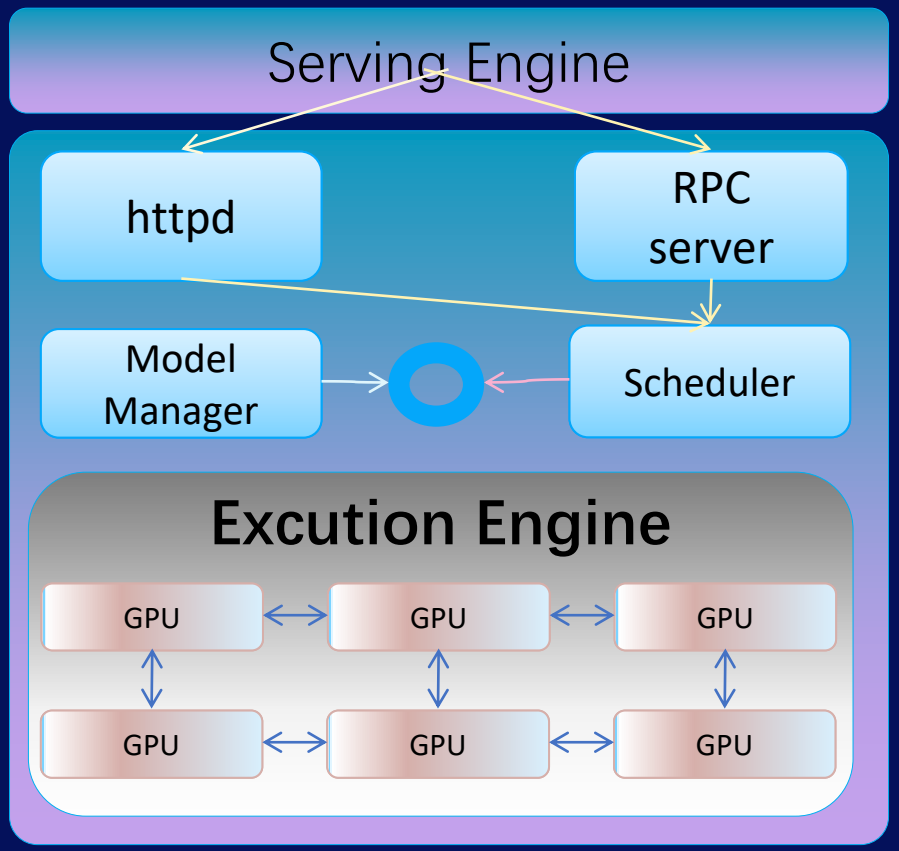
代码能力精调效果，基于HumanEval@Pass1指标：

- **7B模型代码性能提升验证：6 %--> 16.26%**
- **65B模型代码性能提升验证：23.7%-->34.75%**
- **15B模型代码性能提升验证：32%-->50.5%**

Model	Size	HumanEval@1
GPT4	1.8T	67%
Pangu-Coder	15B	62%
ZTE-Coder ★	15B	50.5%
GPT-3.5	175B	48.1%
PaLM-Coder	540B	36%
Codex-12B	12B	28.8
LLaMA	65B	23.7

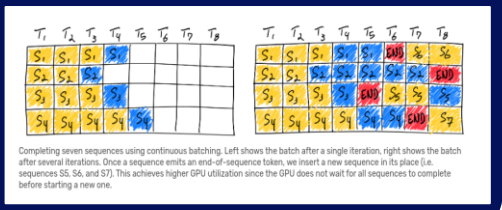
代码生成类业务实践：模型部署优化

挑战	说明
模型大	15B模型推理需要显存超过30 GB，需要40G显存的显卡才能单卡推理
推理资源需求大	初始版本上线单卡吞吐量30tokens/s。系统需求为4k tokens/s，推理资源需求超过100张卡，因此亟需大幅提升系统吞吐量，降低部署成本。



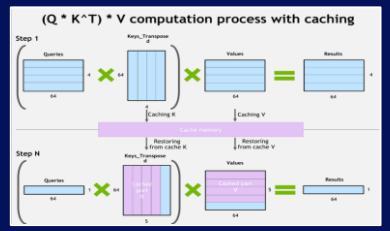
服务层

Continuous Batching

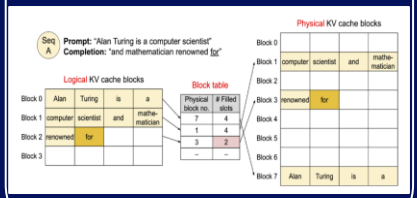


引擎层

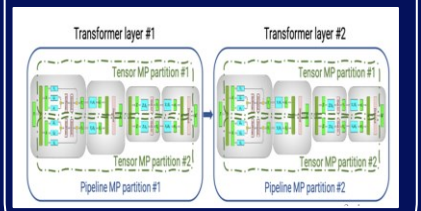
KV Cache



Paged Attention

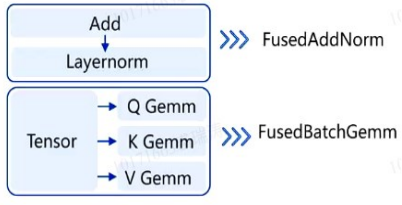


TP/DP/PP

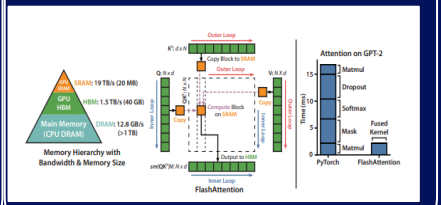


算子层

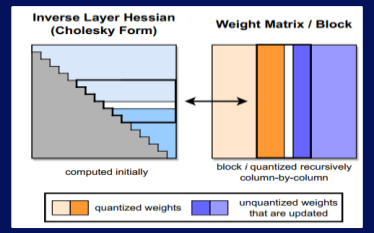
算子融合



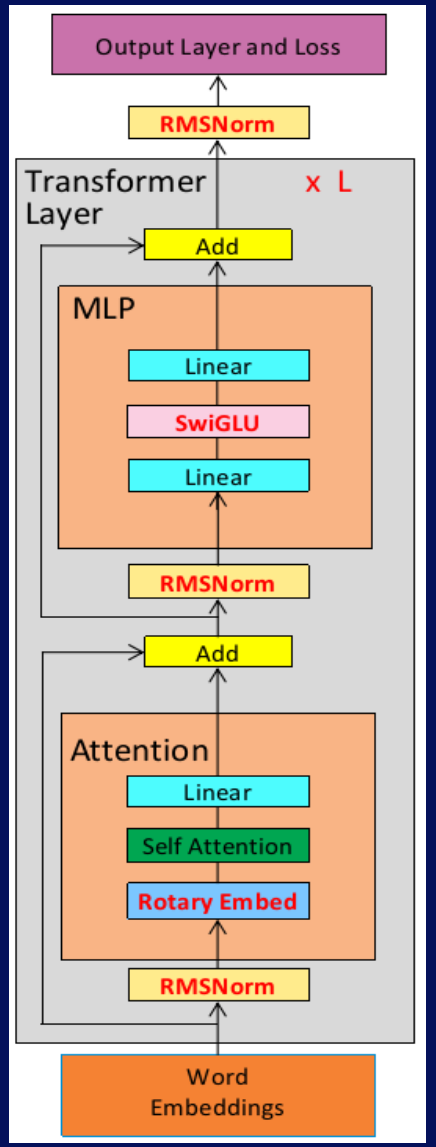
Flash Attention



AWQ/GPTQ



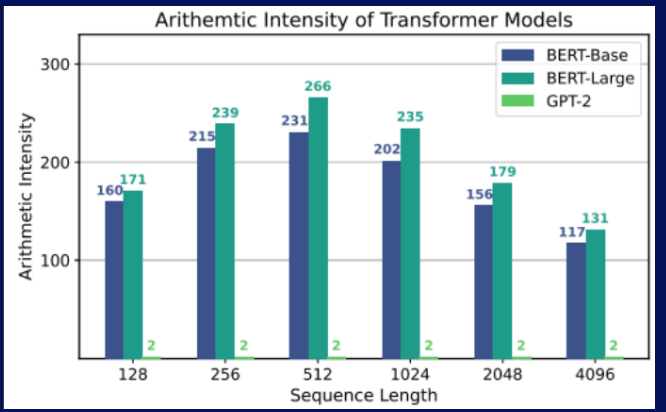
模型部署优化：LLM模型计算特点



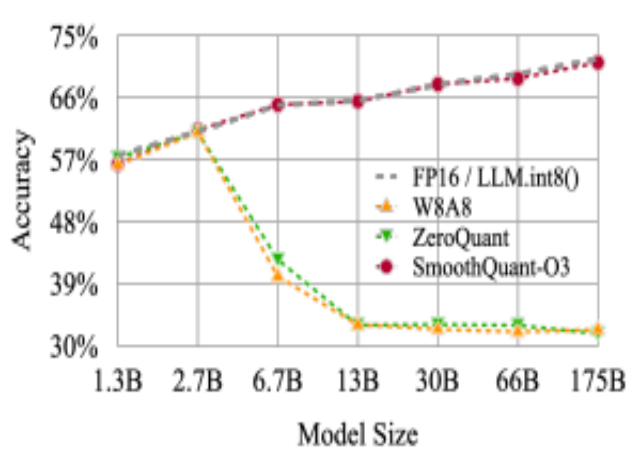
LLaMA -7B	Total	Attention-QKV	Attention -Score	Attention-Projection	MLP	Embedding	Logits
计算量	13.48 GFLOPs	3.22 GFLOPs	0.34 GFLOPs	1.07 GFLOPs	8.59 GFLOPs	0	0.26 GFLOPs
显存占用量	13.50 GB	3.56 GB	0	1.07 GB	8.59 GB	0.28 GB	0
访存量	14.11 GB	3.56 GB	0.17 GB	1.24 GB	8.59 GB	0.28 GB	0.26 GB

计算特点：

- LLM模型一般是Decoder-only结构，计算强度会小于1，属于访存受限模型
- 存在较多非线性计算，如SwiGLU激活函数，及归一化操作，旋转位置编码操作等



模型部署优化：LLM 量化特点



量化方法	适用模型规模	内存	加速效果	量化bit	精度损失
LLM.int8()	175B	均可降低内存消耗	无加速	8bit	无精度损失
ZeroQuant	20B以下		有加速效果	支持低比特	损失可忽略（低比特需蒸馏）
SmoothQuant	175B		有加速效果	8bit	损失可忽略
GPTQ	175B		有加速效果	支持低比特	损失可忽略
AWQ	100B以上		有加速效果	支持低比特	损失可忽略

开源框架	量化支持	稀疏支持
Transformers	LLM.int8()	-
FasterTransformer	SmoothQuant	-
DeepSpeed	ZeroQuant	head pruning，sparse pruning

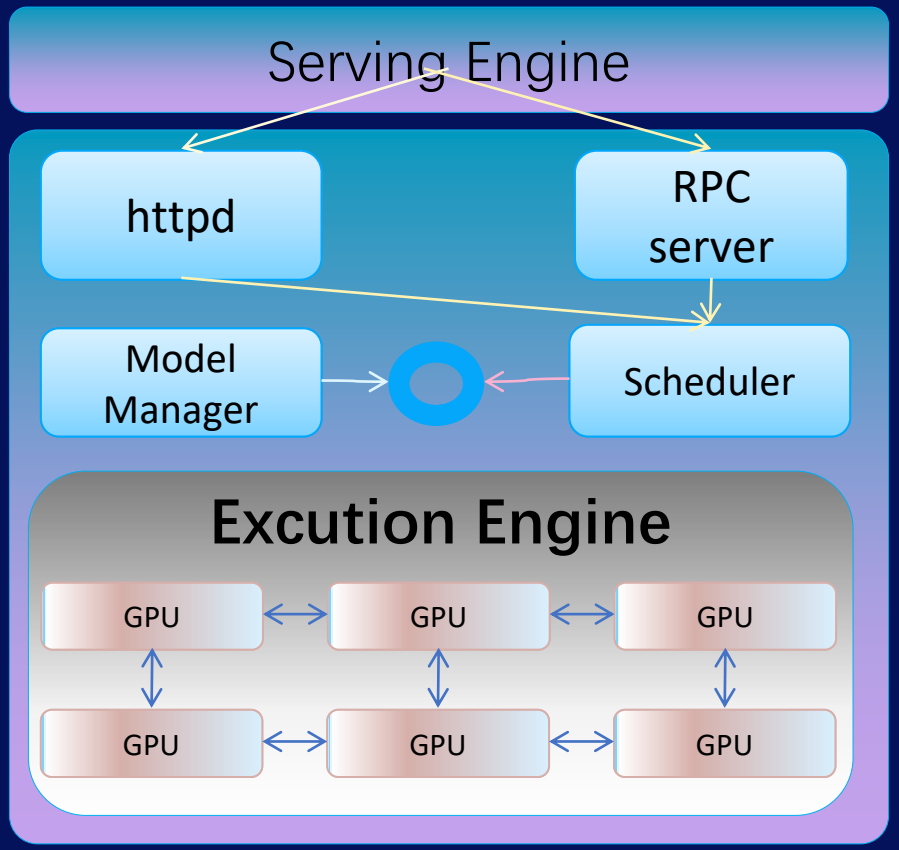
对参数量超过2.7B的大模型使用普通量化方法，模型的精度会出现断崖式下降，因此针对大模型需要专有的量化算法。

模型量化技术为保证模型精度，从全模型量化，发展为按层量化（Per-Layer），再发展按通道量化（Per-Channel），以及目前最新的按分组量化（Per-Group）。

综合来看，GPTQ和AWQ方法均支持3，4bit量化，支持部署更大规模参数的模型，比其他方法更有优势。相比GPTQ，AWQ方法更有加速优势，是目前研究的重点。

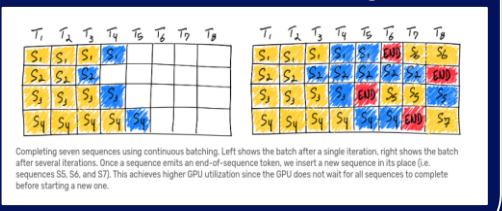
代码生成类业务实践：模型部署优化

挑战	说明
模型大	15B模型推理需要显存超过30 GB，需要40G显存的显卡才能单卡推理
推理资源需求大	初始版本上线单卡吞吐量30tokens/s。系统需求为4k tokens/s，推理资源需求超过100张卡，因此亟需大幅提升系统吞吐量，降低部署成本。



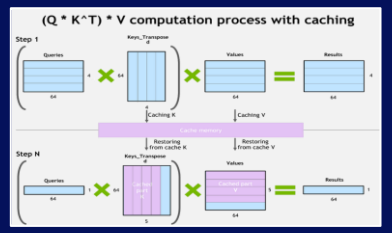
服务层

Continuous Batching

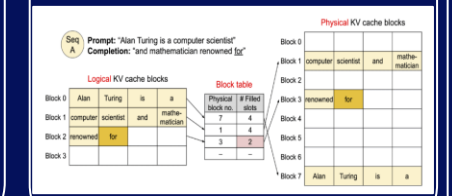


引擎层

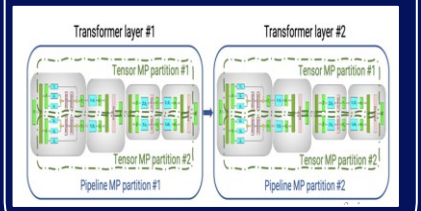
KV Cache



Paged Attention

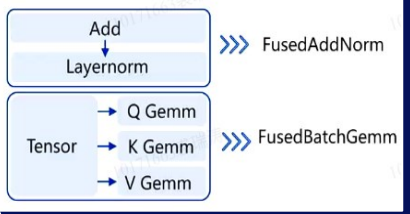


TP/DP/PP

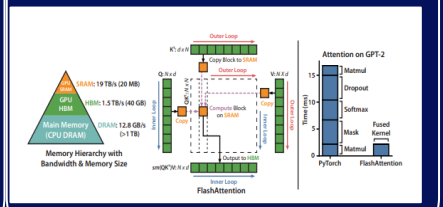


算子层

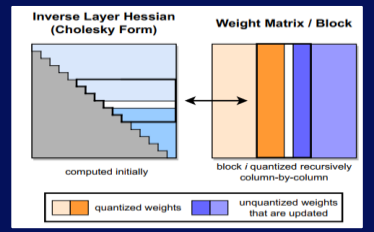
算子融合



Flash Attention



AWQ/GPTQ



模型部署优化实践：量化

GPTQ int4量化 LLama模型

- 使用GPTQ-for-LLaMa-triton
- 关注HumanEval pass@1 指标
- max_length=2048；num_beams=2；num_return_sequences=2;

模型	原始精度	GroupSize	精度	显存
LLaMA 7B	10.5	128	7.93	3.95
		-1	5.79	3.8
LLaMA 65B	23.7	128	22.9	32.88
		-1	21.3	32.5

Test environment: 1*A100(40G)

AWQ int4量化 LLama模型

- 使用Faster Transformer
- 关注HumanEval pass@1 指标
- max_length=2048；num_beams=2；num_return_sequences=2;

模型	原始精度	GroupSize	精度	显存
LLaMA 7B	10.5	128	12.1	4.21
		-1	5.5	4.19
LLaMA 65B	23.7	128	24.1	35.3
		-1	24.7	35.0

Test environment: 1*A100(40G)

模型部署优化实践：量化

GPTQ int4量化 LLama模型

- 使用GPTQ-for-LLaMa-triton
- 关注HumanEval pass@1 指标
- max_length=2048；num_beams=2；num_return_sequences=2;

模型	原始精度	GroupSize	精度	显存
LLaMA 7B	10.5	128	7.93	3.95
		-1	5.79	3.8
LLaMA 65B	23.7	128	22.9	32.88
		-1	21.3	32.5

Test environment: 1*A100(40G)

AWQ int4量化 Starcoder模型

- 使用vLLM
- 关注HumanEval pass@1 指标
- max_length=2048；num_beams=2；num_return_sequences=2;

模型	原始精度	GroupSize	精度	显存
Starcoder 15B	33.4	128	33.5	28.9GB->9.3GB

Test environment: 1*A100(40G)

模型部署优化实践：FlashAttention

FlashAttention优化验证

- repeats=30
- batch_size=1
- nheads=16
- seqlen=512
- n=1024
- d=n//nheads
- dropout_p=0.1
- causal=False
- dtype=torch.float16

Attention	Forward pass	Backward pass	Forward + Backward pass
PyTorch Standard Attention	568.90 us	700.48 us	493.66 us
Flash-attention	130.88 us	473.44 us	1.98 ms
accelerate rate	4.35	1.48	4.01

Test environment: 1*A100(40G)

模型部署优化实践：PagedAttention

starcoder 15B吞吐量验证

- 使用vLLM，Offline，gpu_memory_utilization=0.9，temperature=1.0，top_p=1.0，未开启beach_search，其他默认

Input Length	Output Length	Num Requests	Throughput (tokens/s/card)
256	256	1	68.06
256	256	10	551.16
512	512	1	64.07
512	512	10	505.86
1024	1024	1	58.95
1024	1024	10	434.15
2048	2048	1	50.05
2048	2048	10	335.72

Test environment: 1*A100(40G)

starcoder 15B时延验证

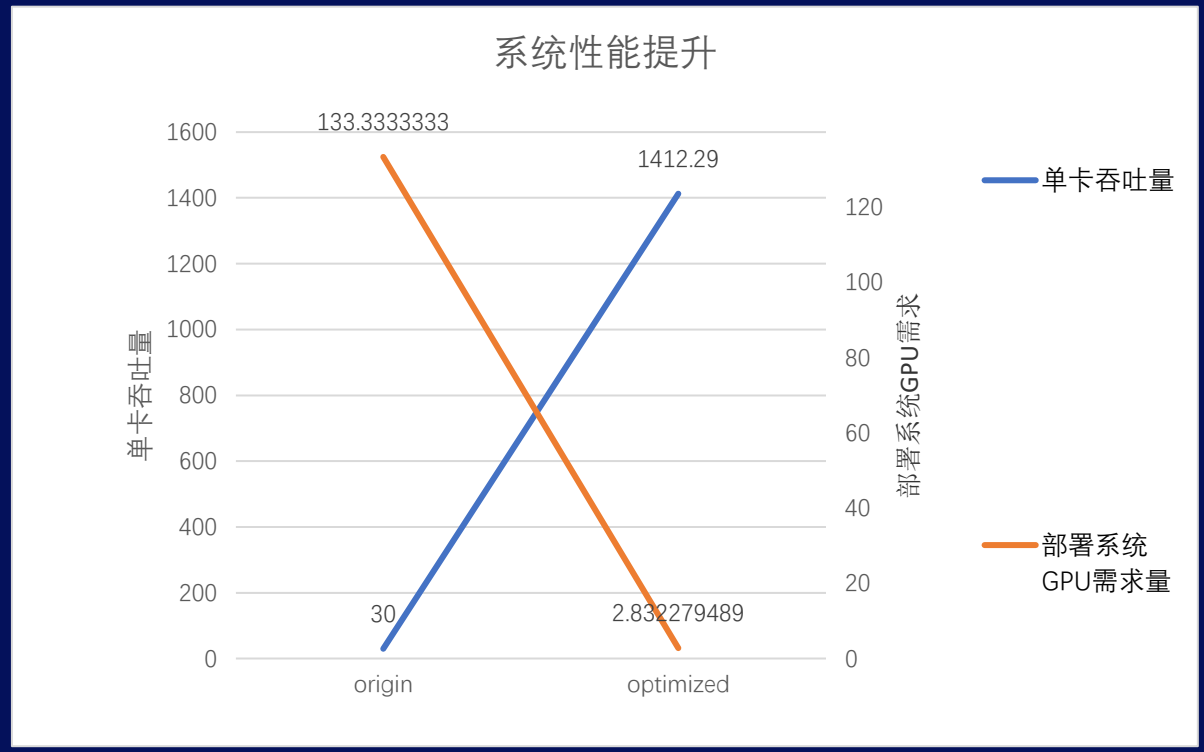
- 使用vLLM，Offline，gpu_memory_utilization=0.9，temperature=1.0，top_p=1.0，未开启beach_search，其他默认

Batchsize	Input Length	Output Length	Total elapsed time(sec)	Latency (ms/output_token/gpu)
1	256	256	7.5586	29.53
1	512	512	15.8277	30.91
1	1024	1024	34.7495	33.94
1	2048	2048	82.0953	40.09
1	4096	4096	215.1392	52.52
10	1024	1024	50.7713	9.92
20	1024	1024	61.0197	2.98
30	1024	1024	74.8276	2.44

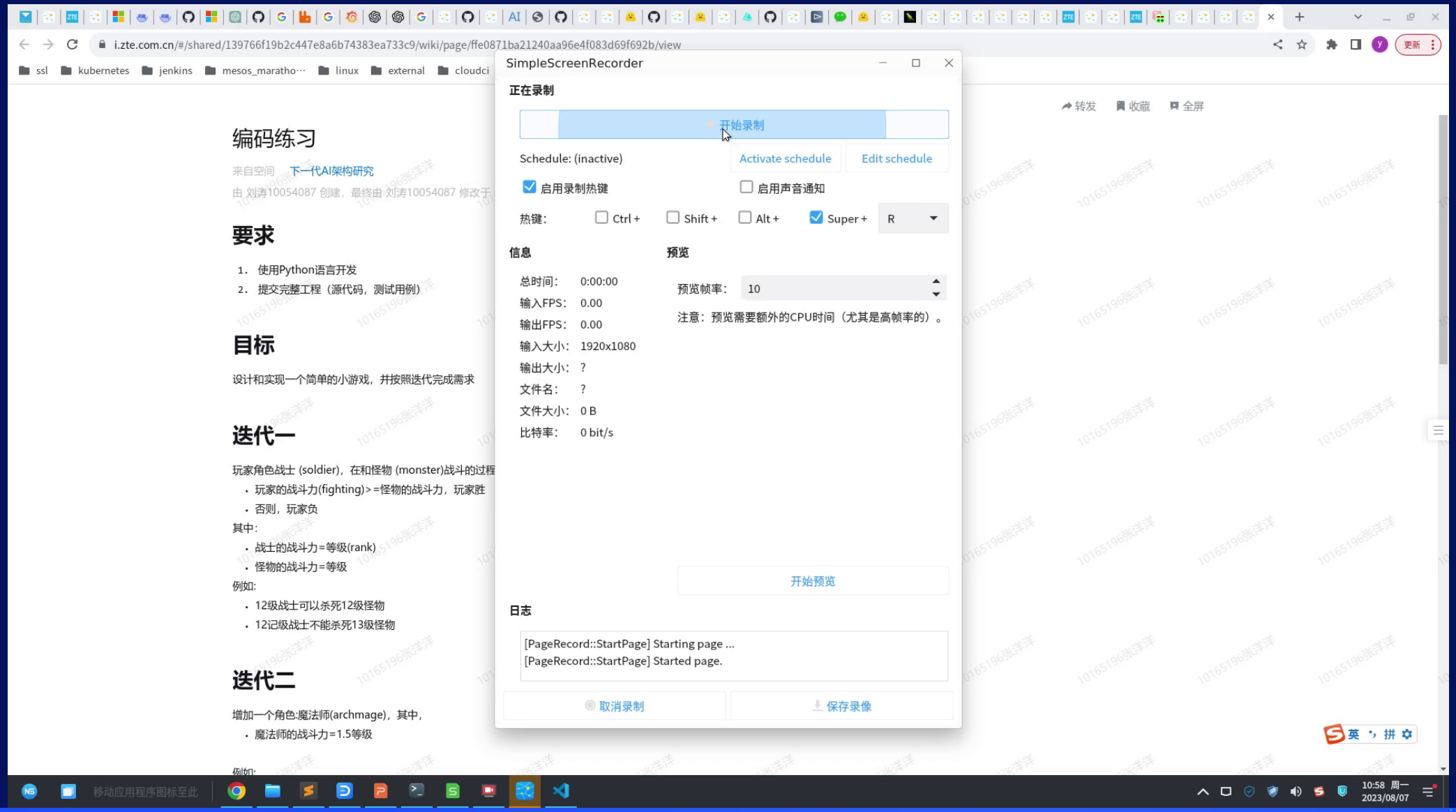
Test environment: 1*A100(40G)

模型部署优化效果

Requests per second	Iutput Length	Output Length	Throughput(req/sec)	Throughput(tokens/sec)
1	256	256	0.51	263.47
10	256	256	3.36	1718.42
1	512	512	0.34	347.57
10	512	512	1.53	1565.82
1	1024	1024	0.18	366.18
10	1024	1024	0.56	1140.26
使用HumanEval真实数据测试				
1	174.64	373.58	0.96	518.01
10	174.64	373.58	2.29	1318.22



辅助编程应用演示


i.zte.com.cn/#/shared/139766f19b2c447e8a6b74383ea733c9/wiki/page/ffe0871ba21240aa96e4f083d69f692b/view
ssl
kubernetes
jenkins
mesos_maratho···
linux
external
cloudci
转发
收藏
全屏
编码练习
来自空间 下一代AI架构研究
由 刘涛10054087 创建，最终由 刘涛10054087 修改于
要求
1. 使用Python语言开发
2. 提交完整工程（源代码，测试用例）
目标
设计和实现一个简单的小游戏，并按照迭代完成需求
迭代一
玩家角色战士 (soldier)，在和怪物 (monster)战斗的过程
• 玩家的战斗力(fighting)>=怪物的战斗力，玩家胜
• 否则，玩家负
其中：
• 战士的战斗力=等级(rank)
• 怪物的战斗力=等级
例如:
• 12级战士可以杀死12级怪物
• 12记级战士不能杀死13级怪物
迭代二
增加一个角色:魔法师(archmage)，其中，
• 魔法师的战斗力=1.5等级
例如:
SimpleScreenRecorder
正在录制
开始录制
Schedule: (inactive)
Activate schedule
Edit schedule
启用录制热键
启用声音通知
热键:
Ctrl +
Shift +
Alt +
Super +
R
信息
总时间: 0:00:00
输入FPS: 0.00
输出FPS: 0.00
输入大小: 1920x1080
输出大小: ?
文件名: ?
文件大小: 0 B
比特率: 0 bit/s
预览
预览帧率: 10
注意：预览需要额外的CPU时间（尤其是高帧率的）。
开始预览
日志
[PageRecord::StartPage] Starting page ...
[PageRecord::StartPage] Started page.
取消录制
保存录像
移动应用程序图标至此
10:58 周一
2023/08/07

AiDD AI+ 软件研发数字峰会
AI+ software Development Digital summit

感 谢 聆 听