

「大模型/AIGC与智能化基础软件」论坛

面向智能应用的算力硬件调度与管理

陈榕

上海交通大学

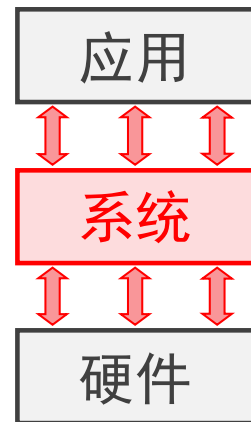
南京 · 2024. 1

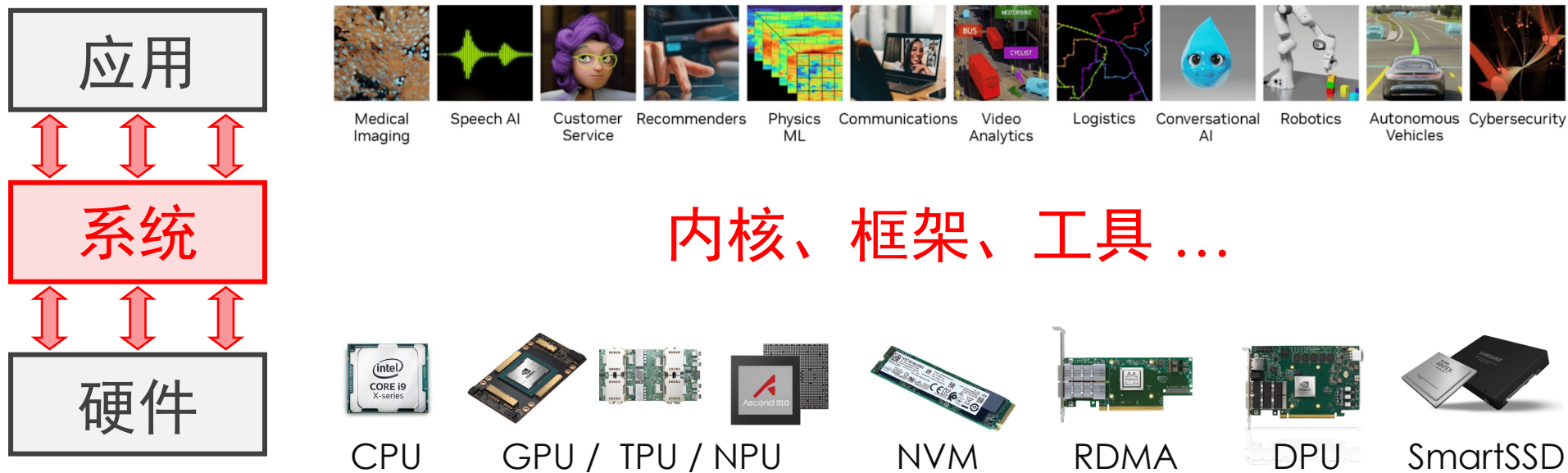
陈榕

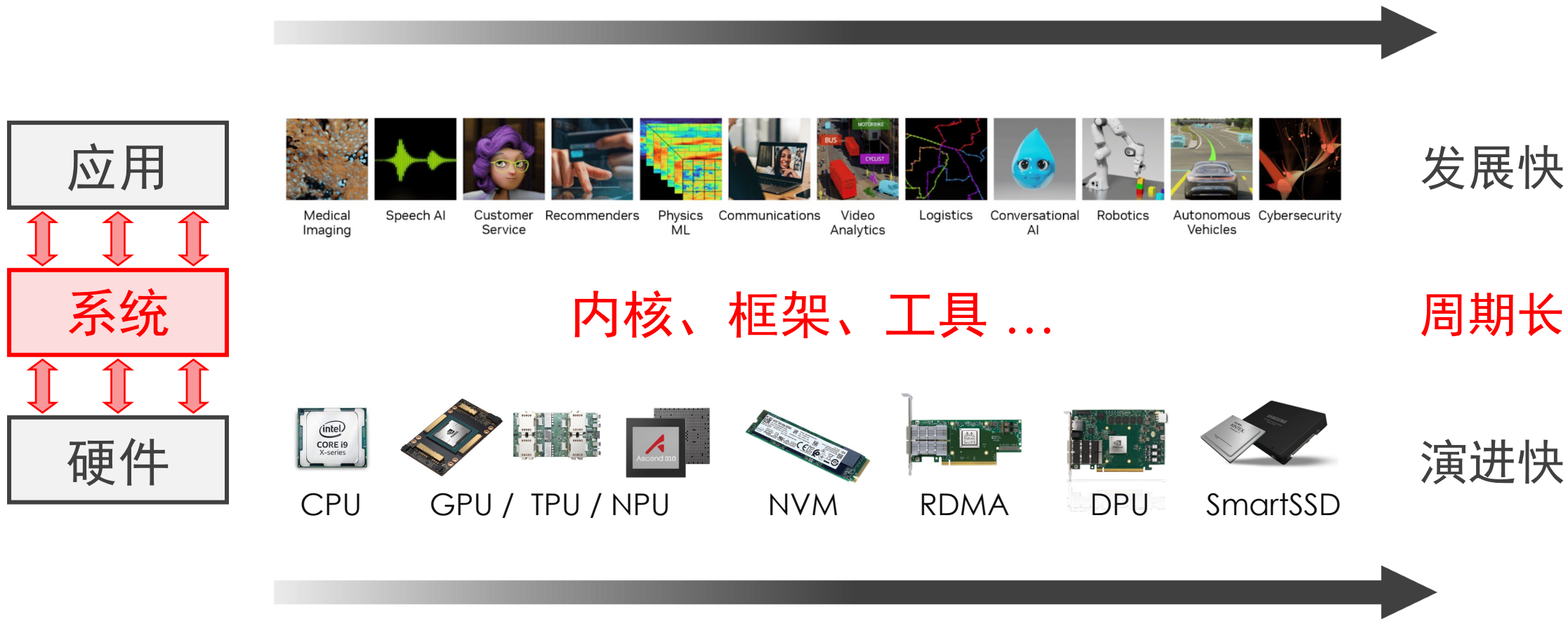
- ▶ 并行与分布式系统研究所（IPADS），上海交通大学（2012）

研究领域：**基础系统软件（操作系统、分布式系统等）**

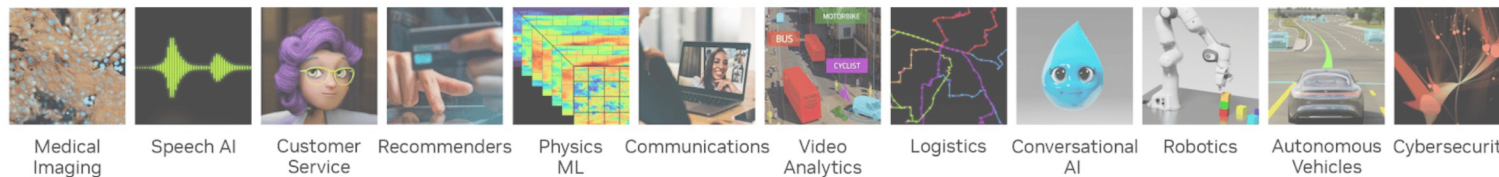
- ▶ OSDI/SOSP（12篇）、EuroSys/ATC（12篇）
- ▶ 最佳论文奖：EuroSys 2015、ICPP 2007
- ▶ 2020年华为“奥林帕斯先锋奖”（第一完成人）







应用



发展快

需求

高吞吐、低时延、可扩展、大规模 ...

发展趋势

系统

内核、框架、工具 ...

源自“共性”

能力

算力、存力、带宽、持久、隔离 ...

演进趋势

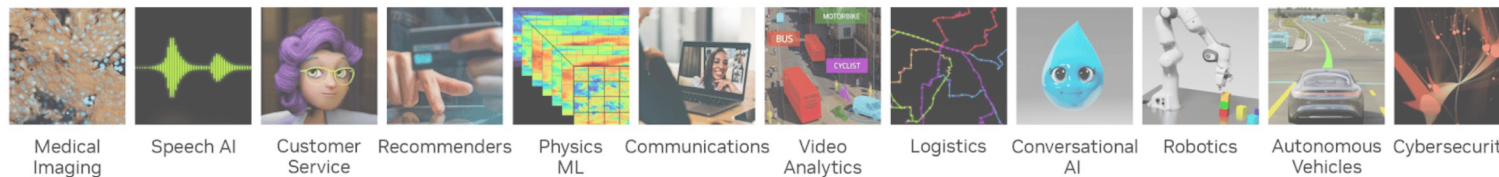
硬件



演进快



应用



发展快

需求

高吞吐、低时延、可扩展、大规模 ...

发展趋势

系统

顺势而为

内核、框架、工具 ...

源自“共性”

能力

算力、存力、带宽、持久、隔离 ...

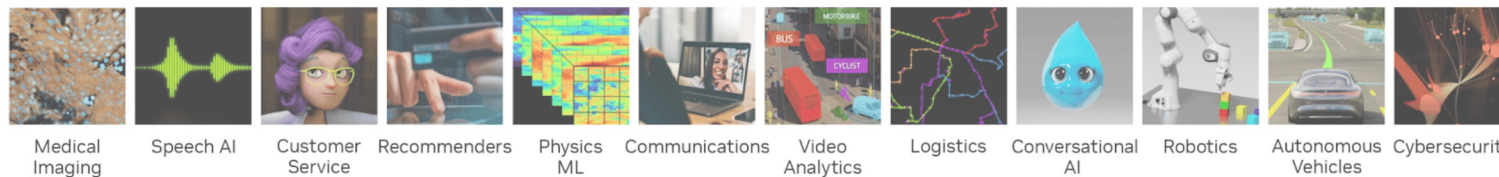
演进趋势

硬件



演进快

应用



发展快

需求

高吞吐、低时延、可扩展、大规模 ...

发展趋势

系统

顺势而为

内核、框架、工具 ...

迎难而上

源自“共性”

能力

算力、存力、带宽、持久、隔离 ...

演进趋势

硬件



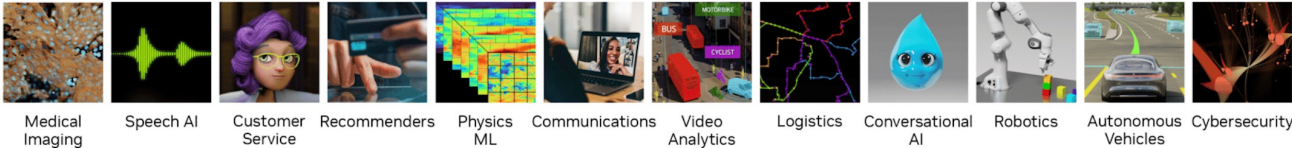
演进快

系统软件研究——智能时代



应用

智能应用



需求



系统



能力

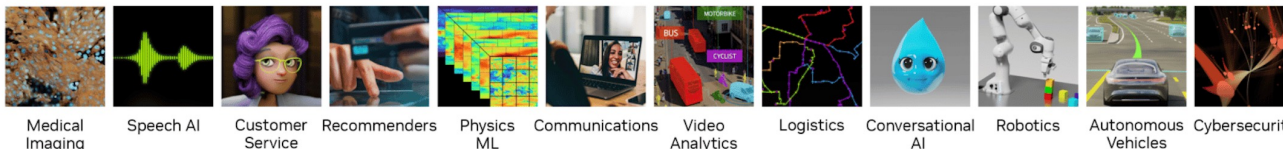
硬件

系统软件研究——智能时代



应用

智能应用



需求

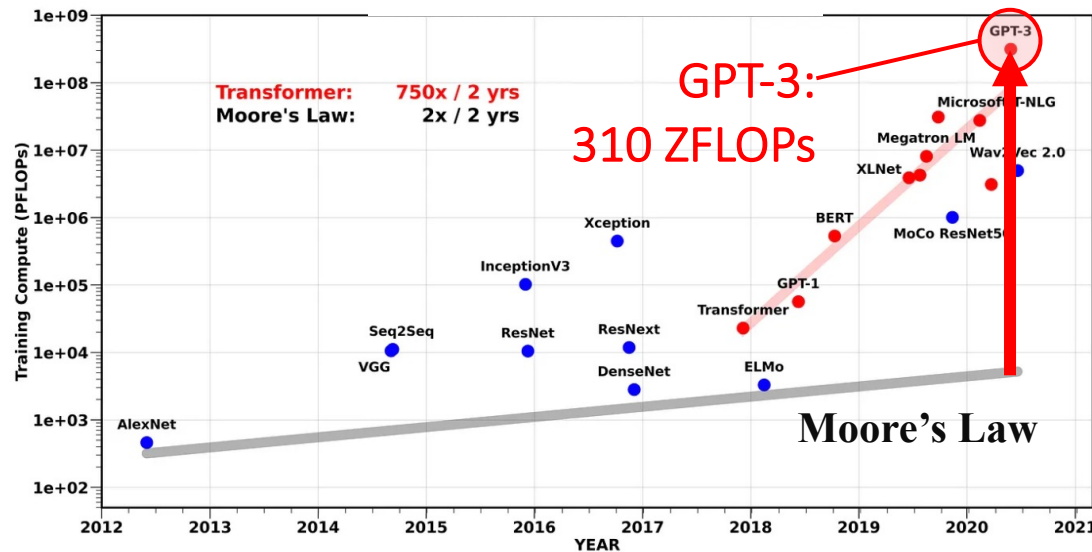
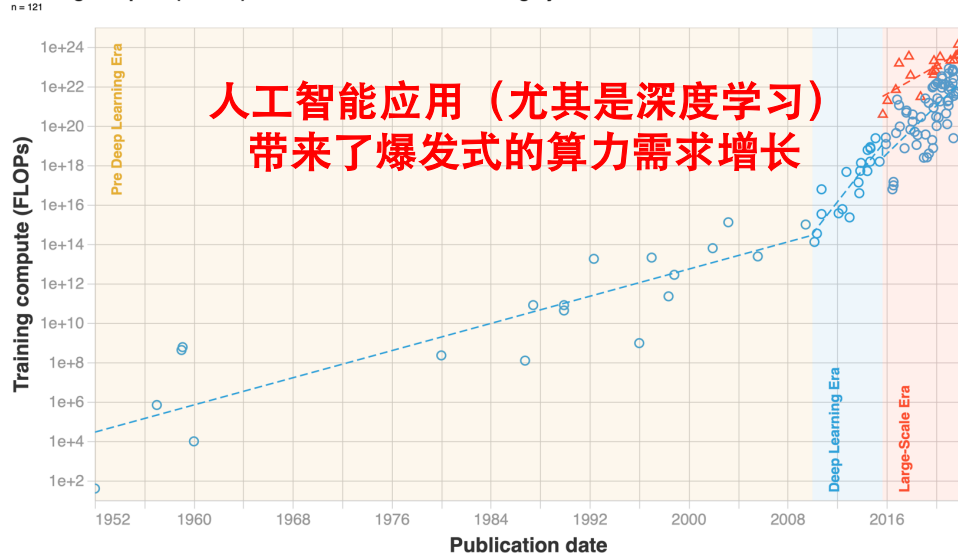
大算力

系统

能力

硬件

Training compute (FLOPs) of milestone Machine Learning systems over time

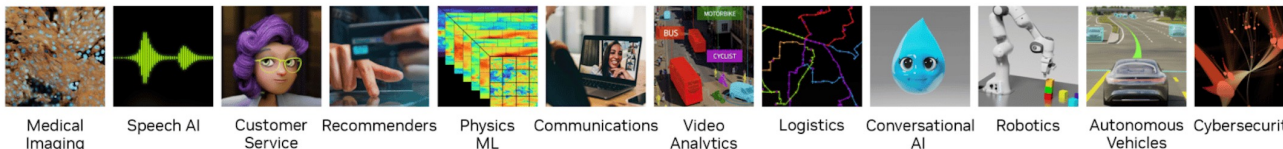


系统软件研究——智能时代



应用

智能应用

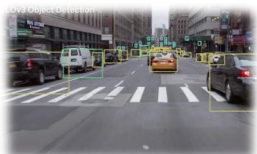


需求

大算力 + 强实时、高性价比、大内存、..

↑

系统



智能车

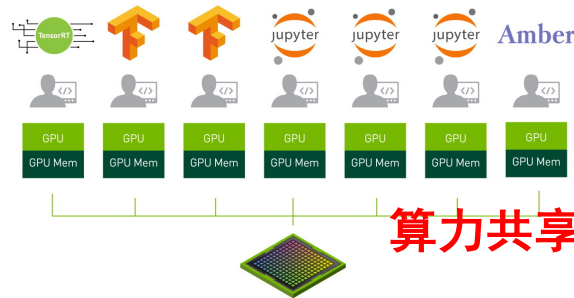
1. 障碍物检测
2. 红绿灯识别
3. 语音助理
4. 疲劳监测
5. ...



↓

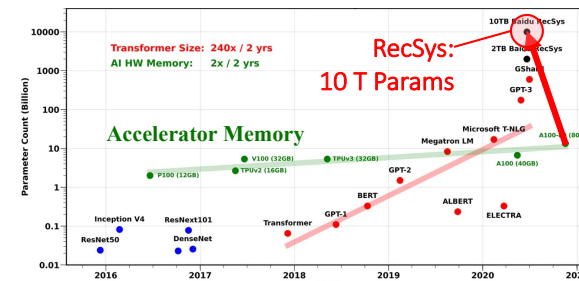
能力

硬件

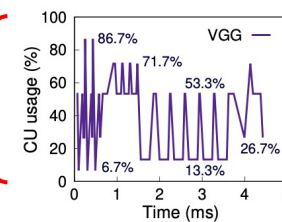


典型DNN任务: 3.5~13.6ms

Model	ResNet	DenseNet	VGG	Inception	Bert
#Kernels	307	207	55	146	205
Exec. Time	13.6	3.5	4.4	8.3	5.4



内存墙



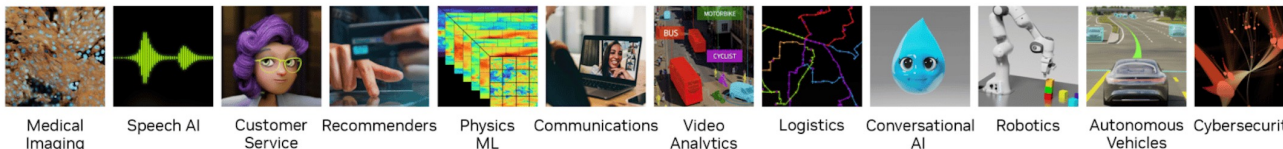
算力需求
亚毫秒级
剧烈波动

系统软件研究——智能时代



应用

智能应用



需求

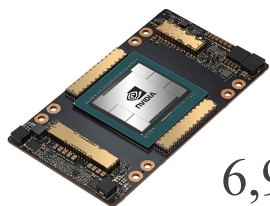
大算力 + 强实时、高性价比、大内存、..

系统

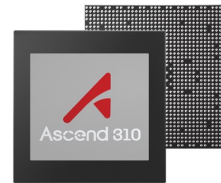
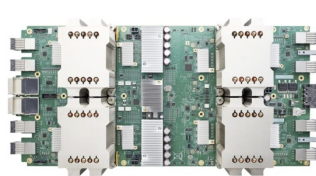
GPU

TPU / NPU / XPU ...

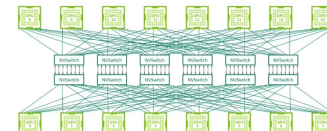
NVLink / NVSwitch



A100
108 SMs
6,912 cores



900GB/s total



3.6TB/s BW

能力

大算力、领域加速、高速互联

硬件

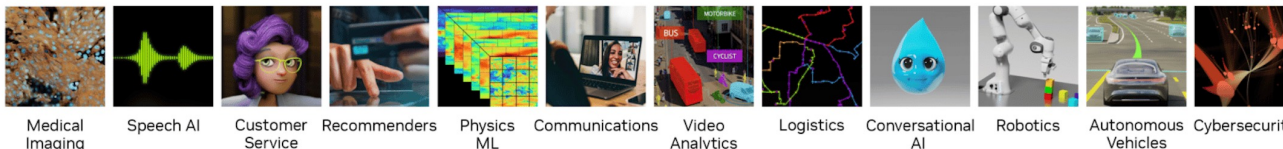
GPU、NPU/XPU、NVLink/NVSwitch/CXL

系统软件研究——智能时代



应用

智能应用



需求

大算力 + 强实时、高性价比、大内存、..

系统

系统框架/服务

操作系统

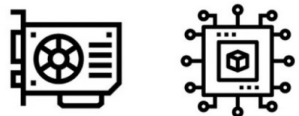
顺势而为

“人工智能”操作系统
关键技术

能力

大算力、领域加速、高速互联

硬件



GPU、NPU/XPU、NVLink/NVSwitch/CXL

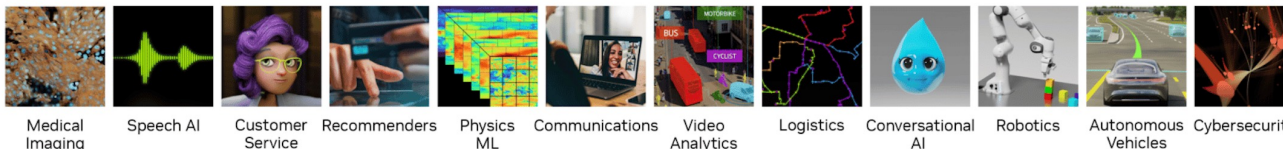


系统软件研究——智能时代



应用

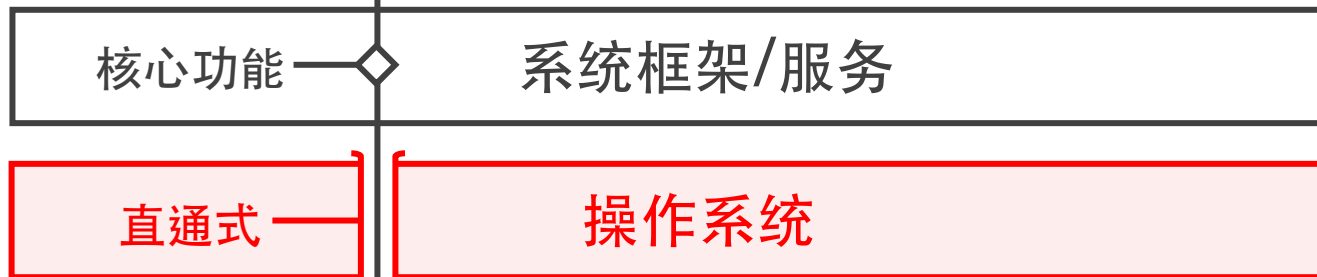
智能应用



需求

大算力 + 强实时、高性价比、大内存、..

系统



顺势而为

“人工智能”操作系统
关键技术

能力

大算力、领域加速、高速互联

硬件



GPU、NPU/XPU、NVLink/NVSwitch/CXL

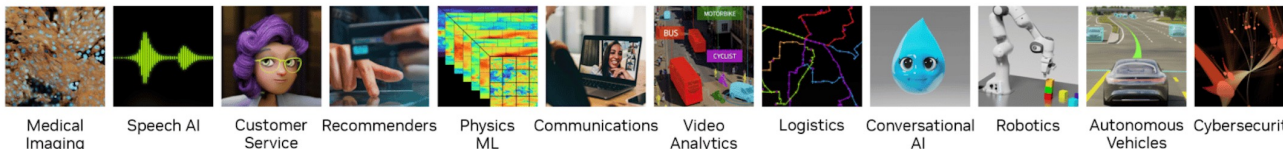


系统软件研究——智能时代



应用

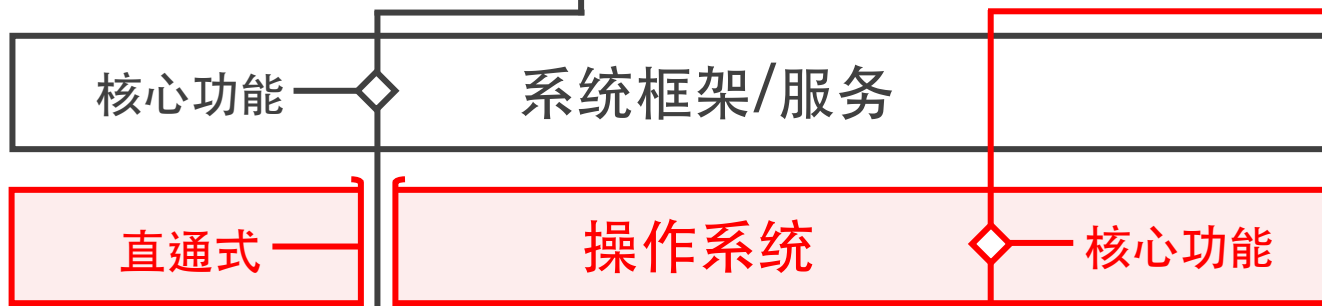
智能应用



需求

大算力 + 强实时、高性价比、大内存、..

系统



迎难而上

面向“大算力+X”需求
调度与管理

能力

大算力、领域加速、高速互联

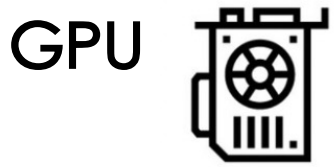
硬件



GPU、NPU/XPU、NVLink/NVSwitch/CXL



GPU调度与管理——大算力+X



高并发算力



GPU Architecture (Nvidia Ampere)

1 GPU
x 8 GPCs / GPU
x 8 TPCs / GPC
x 2 SMs / TPC
= 128 SMs*

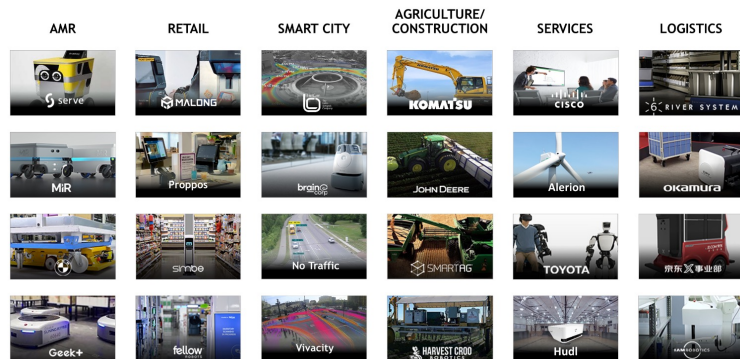
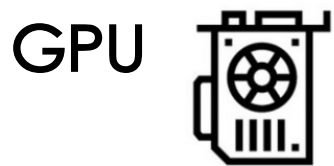
Streaming Multiprocessor



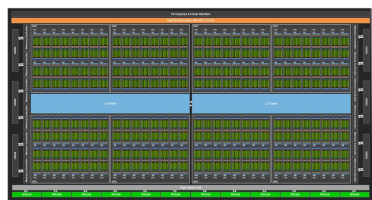
2048 Thds/SM

* GPU调度单元 NVIDIA使用SM、ARM使用CU

GPU调度与管理——大算力+X



高并发算力



GPU Architecture (Nvidia Ampere)

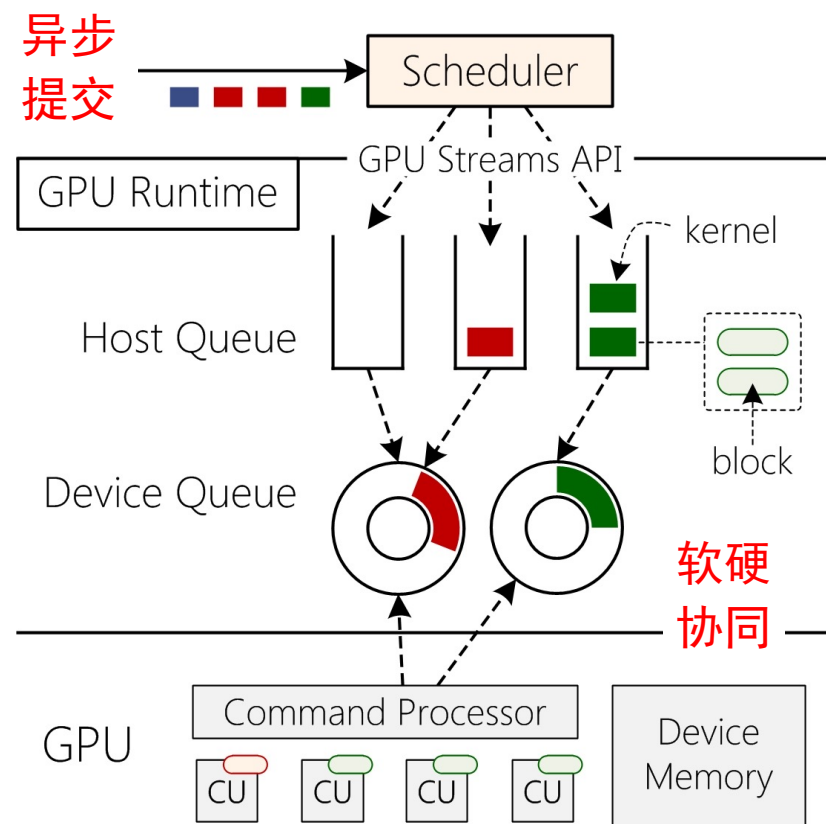
1 GPU
 x 8 GPCs / GPU
 x 8 TPCs / GPC
 x 2 SMs / TPC
 = 128 SMs*

Streaming Multiprocessor



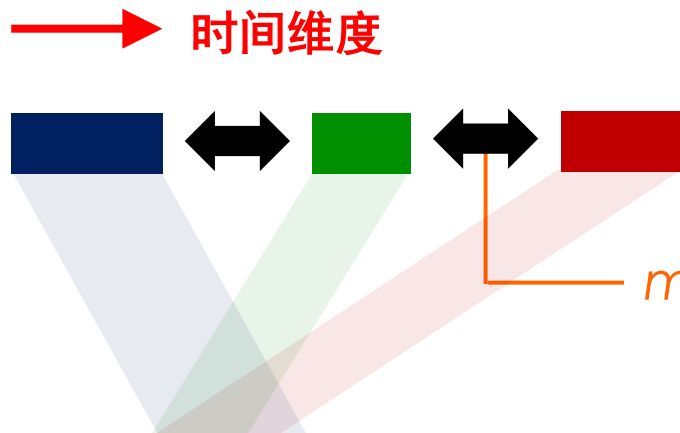
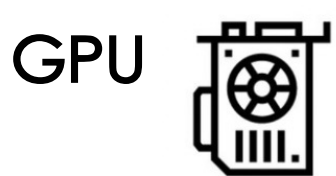
2048 Thds/SM

GPU任务调度

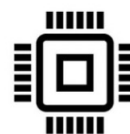


* GPU调度单元 NVIDIA使用SM、ARM使用CU

应用需求: 大算力 + 实时性



vs. CPU

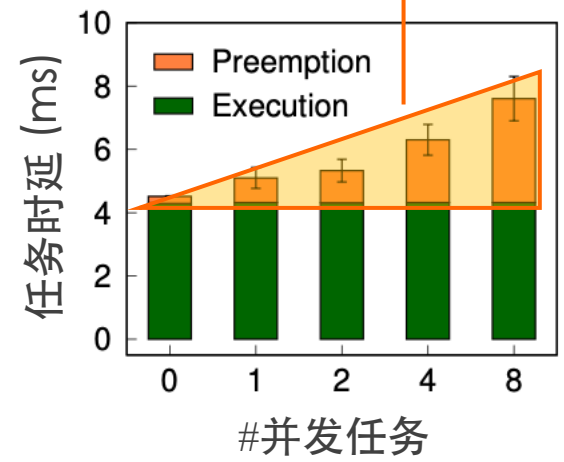


- 执行时间更长 (5~100ms)
- 调度延迟更低 (~5 μ s)

典型DNN推理任务: 3.5~13.6ms

Model	ResNet	DenseNet	VGG	Inception	Bert
#Kernels	307	207	55	146	205
Exec. Time	13.6	3.5	4.4	8.3	5.4

Testbed : AMD Radeon Instinct MI50 GPU




调度开销接近一半*

关键技术: GPU实时任务抢占

* 未考虑GPU内存切换、任务加载等开销

问题/挑战

1. 大算力硬件 **状态多**
任务 **切换慢** } $300\mu s$

GPU A100 

- 128 SMs
- 256 KB regs/SM
- 164 KB shmem/SM

 CPU

$< 1\mu s$
切换延迟

思路/方法

关键洞见：GPU任务多有“幂等性”



重置执行中的任务（不保存状态）

切换延迟 $5\mu s$



问题/挑战

1. 大算力硬件状态多 } 300 μ s
任务切换慢

GPU A100 

- o 128 SMs
- o 256 KB regs/SM
- o 164 KB shmem/SM

CPU 

< 1 μ s
切换延迟

2. 软硬协同异步提交 } > 1ms
任务清理慢

典型DNN推理: 50~300+任务

- o ResNet(307), BERT(205), VGG(55)

思路/方法

关键洞见: GPU任务多有“幂等性”



重置执行中的任务 (不保存状态)

切换延迟 5 μ s

关键设计: 垂直全栈清理

[软] Host Qs: 软件重置队列

[软-硬] Dev Qs: 编译插桩+主动退出

[硬] GPU SM: 硬件指令重置

~30 μ s

清理延迟



GPU抢占调度



“首次在量产GPU上实现了微秒级任务抢占¹”

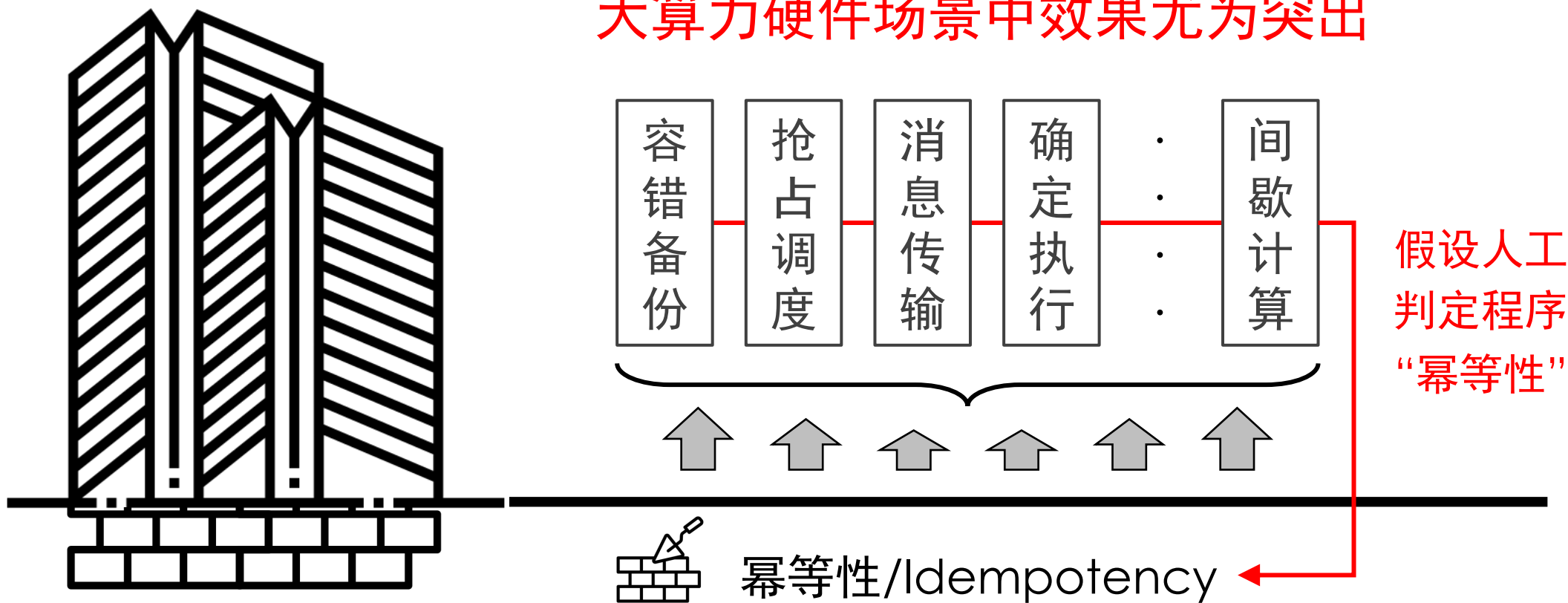
¹ Microsecond-scale Preemption for Concurrent GPU-accelerated DNN Inferences. OSDI 2022.



GPU程序“幂等性”——优化基石



大量优化系统基于幂等性
大算力硬件场景中效果尤为突出



GPU程序“幂等性”——实际情况



幂等/Idempotent

```
__global__ void vectorSet(A)
  idx = bid * blockDim + tid
  A[idx] = VALUE
```

非幂等/Non-idempotent

```
__global__ void vectorInc(A)
  idx = bid * blockDim + tid
  A[idx] = A[idx] + VALUE
```

- vectorSet<<<32,64>>>(x) ★ 幂等
- vectorSet<<<32,16>>>(y) ★ 幂等
- ...

- vectorInc<<<32,64>>>(x) ☆ 非幂等
- vectorInc<<<32,64>>>(y) ☆ 非幂等
- ...

GPU程序“幂等性”——实际情况



幂等/Idempotent

```
__global__ void vectorSet(A)
  idx = bid * bdim + tid
  A[idx] = VALUE
```

非幂等/Non-idempotent

```
__global__ void vectorInc(A)
  idx = bid * bdim + tid
  A[idx] = A[idx] + VALUE
```

条件幂等/Cond-idempotent

```
__global__ void vectorAdd(A,B,C)
  idx = bid * bdim + tid
  A[idx] = B[idx] + C[idx]
```

- vectorSet<<<32,64>>>(x) ★ 幂等
- vectorSet<<<32,16>>>(y) ★ 幂等
- ...

- vectorInc<<<32,64>>>(x) ☆ 非幂等
- vectorInc<<<32,64>>>(y) ☆ 非幂等
- ...

- vectorAdd<<<32,64>>>(x,y,z) ★ 幂等
- vectorAdd<<<32,64>>>(x,y,x) ☆ 非幂等

GPU程序/Kernel 【静态】 【动态】 GPU实例/Instance

GPU程序“幂等性”——实际情况



幂等/Idempotent

```
__global__ void vectorSet(A)
  idx = bid * bdim + tid
  A[idx] = VALUE
```

非幂等/Non-idempotent

```
__global__ void vectorInc(A)
  idx = bid * bdim + tid
  A[idx] = A[idx] + VALUE
```

条件幂等/Cond-idempotent

```
__global__ void vectorAdd(A,B,C)
  idx = bid * bdim + tid
  A[idx] = B[idx] + C[idx]
```

“条件幂等”GPU任务真实存在、且是大多数

GPU Apps	Code	#Kernels	●	○	◐	◑/T
Rodinia [9]	Source	40	7	12	21	2
Parboil [57]	Source	25	4	12	9	0
TVM [60]	Source	308	0	0	308	0
PyTorch [49]	Binary	66	3	1	62	2
TensorRT [44]	Binary	58	0	2	56	0
FT [46]	Binary	50	9	7	34	0
All		547	23	34	490	4

基于官方用例
未发现非幂等实例

89.6%

- vectorAdd<<<32,64>>>(x,y,z) ★ 幂等
- vectorAdd<<<32,64>>>(x,y,x) ☆ 非幂等



大厦将倾

GPU程序/Kernel 【静态】 【动态】 GPU实例/Instance

GPU任务“幂等性”——实际情况



条件幂等/Cond-idempotent

```
__global__ void vectorAdd(A,B,C)
  idx = bid * blockDim + tid
  A[idx] = B[idx] + C[idx]
```



【静态】GPU程序/Kernel

【动态】GPU实例/Instance

vectorAdd<<<32,64>>>(x,y,z) ★ 幂等 (大多数)

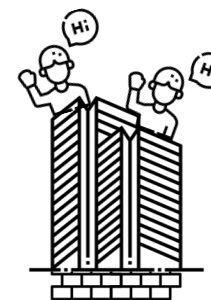
vectorAdd<<<32,64>>>(x,y,☒) ☆ 非幂等

“条件幂等”GPU任务的大多数实例是“幂等”

GPU Apps	#Instances	●/★	○/☆	◐/★	◑/☆
Rodinia [9]	4,527	85	78	4,334	30
Parboil [57]	1,033	103	738	192	0
TVM [60]	609	0	0	609	0
PyTorch [49]	1,570	151	1	1,131	287
TensorRT [44]	478	0	8	465	5
FT [46]	10,000	229	988	7,119	1,664
All	18,217	568	1,813	13,850	1,986

87.5%

关键技术：GPU实例的“幂等性”判定



问题/挑战

1. 如何安全判定“幂等性”？

现有方法：~~“读后写”~~ → “幂等”

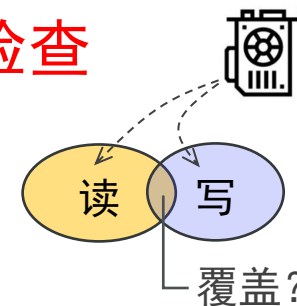
- × GPU执行访存高度并行
- × 无法确定线程间访存顺序

思路/方法

关键设计：“读-写域”覆盖检查

~~读写覆盖~~ → “幂等” ★

- 无误判/no false positive
- 高精度/less false negative



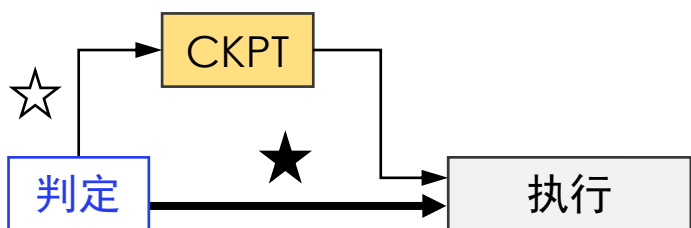
问题/挑战

1. 如何安全判定“幂等性”？

现有方法：“读后写” → “幂等”

- × GPU执行访存高度并行
- × 难以确定内存读写顺序

2. 如何在执行前完成判定？



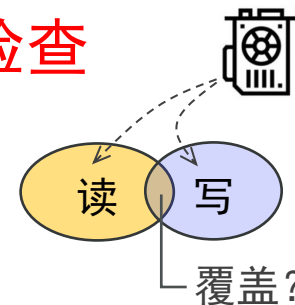
典型基于“幂等性”的系统架构

思路/方法

关键设计：“读-写域”覆盖检查

读写覆盖 → “幂等” ★

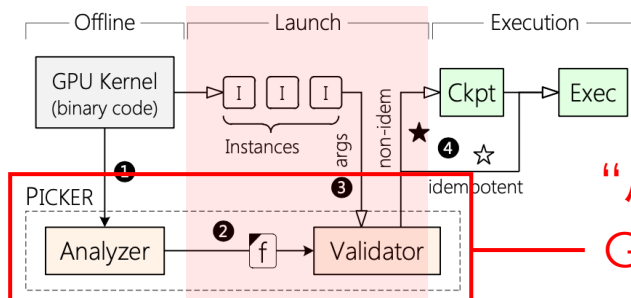
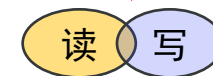
- 无误判/no false positive
- 高精度/less false negative



关键洞见：“启动参数”确定“读-写域”

```
vectorAdd<<<32, 64>>>(x, y, z)
```

[LAs]



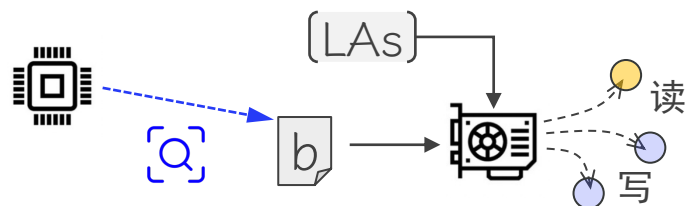
“启动期”

GPU实例分析工具

问题/挑战

3. 如何计算GPU程序读-写域?

× CPU无法执行GPU访存指令



思路/方法

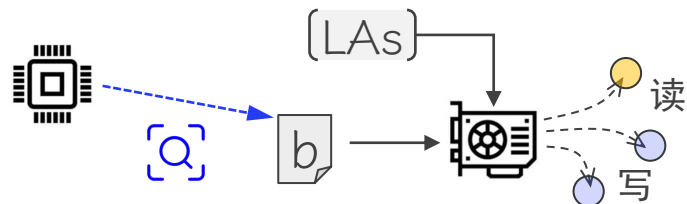
关键设计：基于符号地址的模拟计算



问题/挑战

3. 如何计算GPU程序读-写域？

× CPU无法执行GPU访存指令



4. 如何快速判定实例“幂等”？

× 访存次数多 (GBs)

× 启动时间短 ($<5\mu s$)

执行时间 (3-13ms)

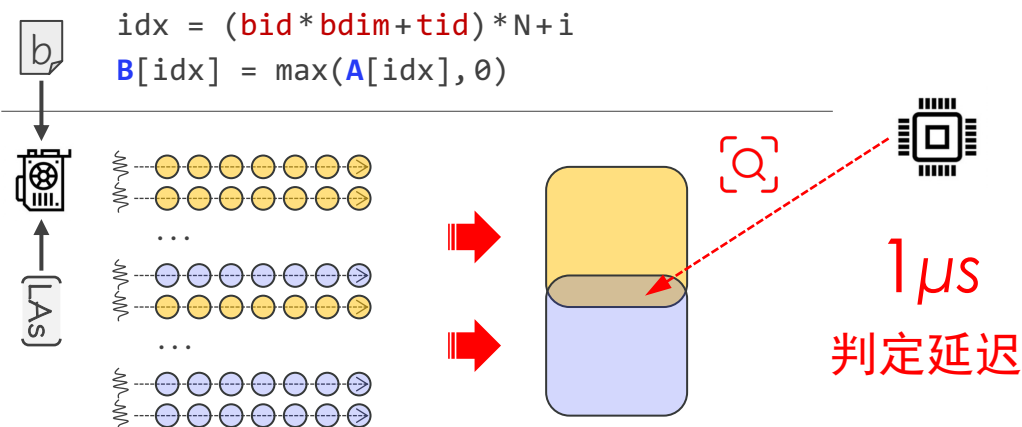
~50ms
判定延迟

思路/方法

关键设计：基于符号地址的模拟计算

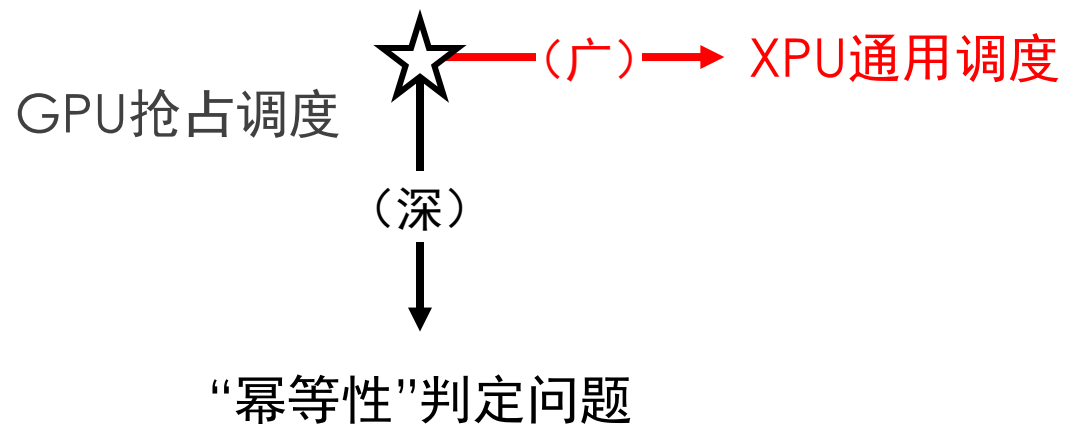


关键观察：访存地址连续性、单调性



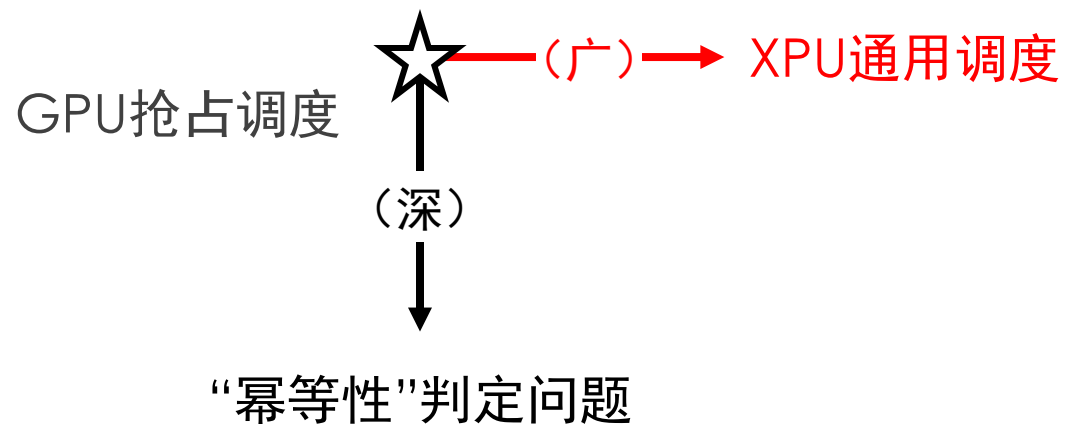


“首次实现了GPU实例的幂等性动态判定、且做到了安全、快速、精准”





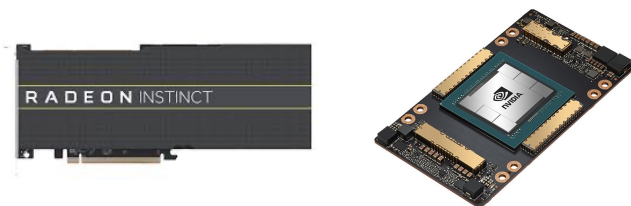
“支持了GPU/NPU/DSA等算力硬件和各类调度策略”



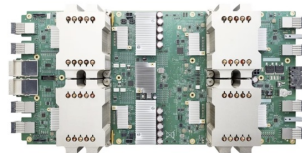
算力硬件（XPU）繁荣



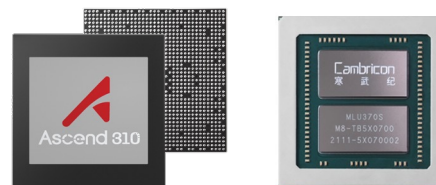
GPU



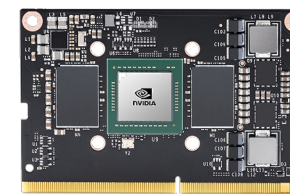
TPU



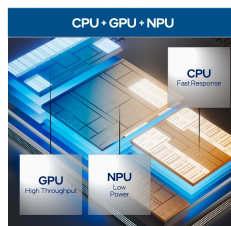
NPU



SoC



AI·PC



What is an AI PC?

A PC with new NPU silicon that brings new AI experiences in productivity, creativity, and security through a combination of the CPU, GPU, and the NPU.

AI·PC 是一个“混合体”

在硬件上集成了**混合AI算力单元**
且能够本地运行**个人大模型**
创建个性化的**本地知识库**
实现**自然语言交互**

——《AI·PC产业(中国)白皮书》

- Comes with new NPU, CPU, and GPU powered silicon
- Comes with Copilot
- Has the Copilot key

- 8-core CPU
- 8-core GPU
- Secure Enclave
- 16-core Neural Engine

- Leads the AI PC era
- World's first x86 processor with integrated NPU
- AI PC
- 荣耀MagicBook Pro 16
- 发布时间: 2024年3月18日 19:30

- Lenovo AI PC
- Smarter technology for all
- Lenovo CES 24

算力硬件任务调度现状

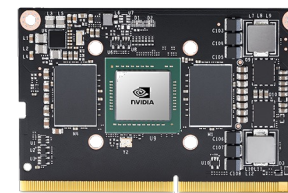
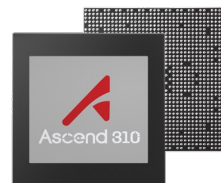
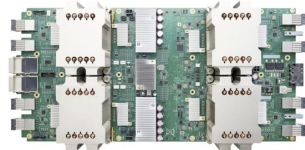
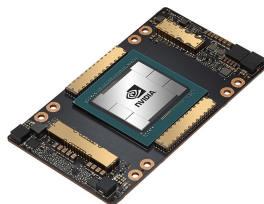
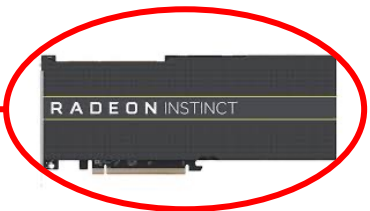


GPU

TPU

NPU

SoC



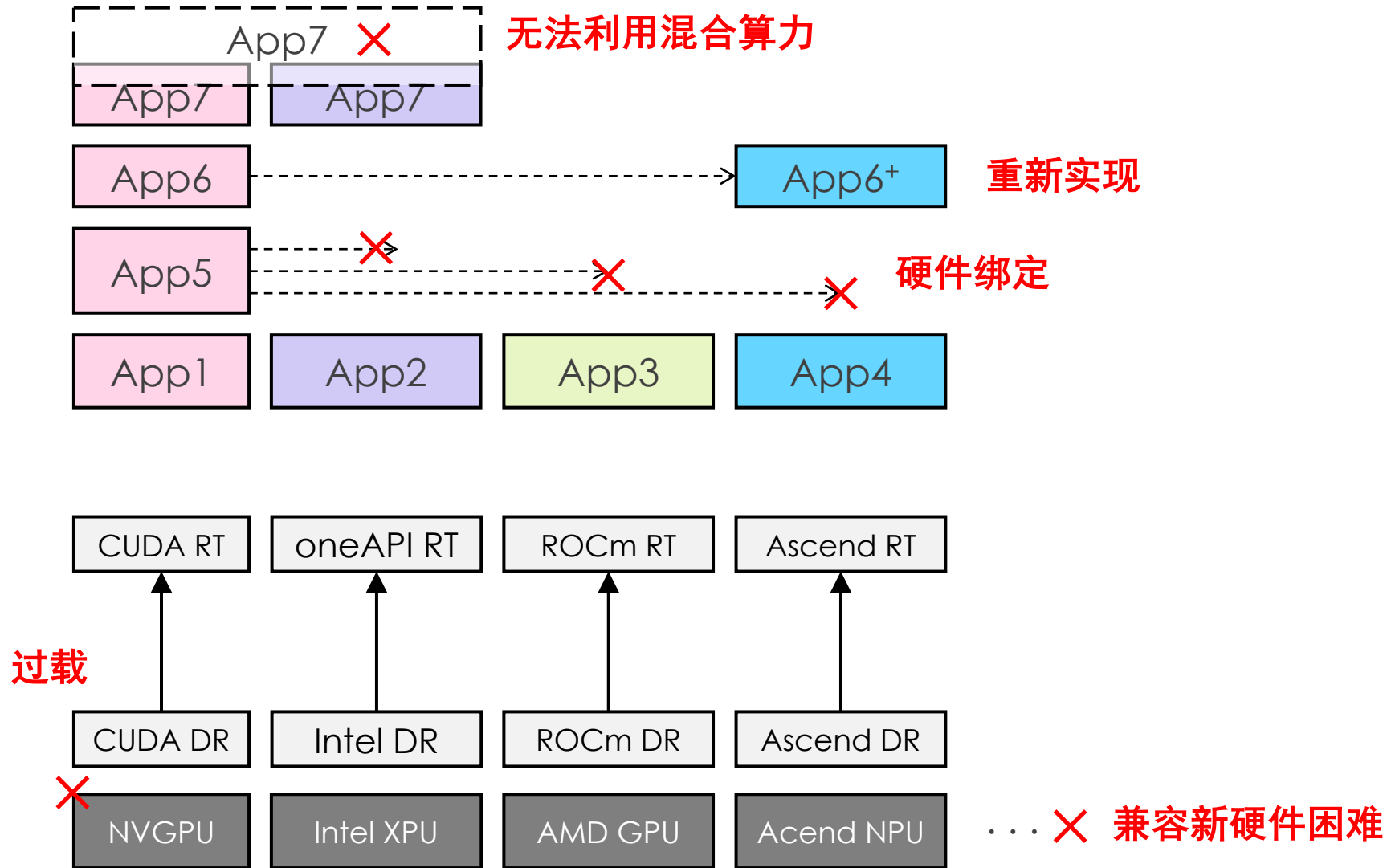
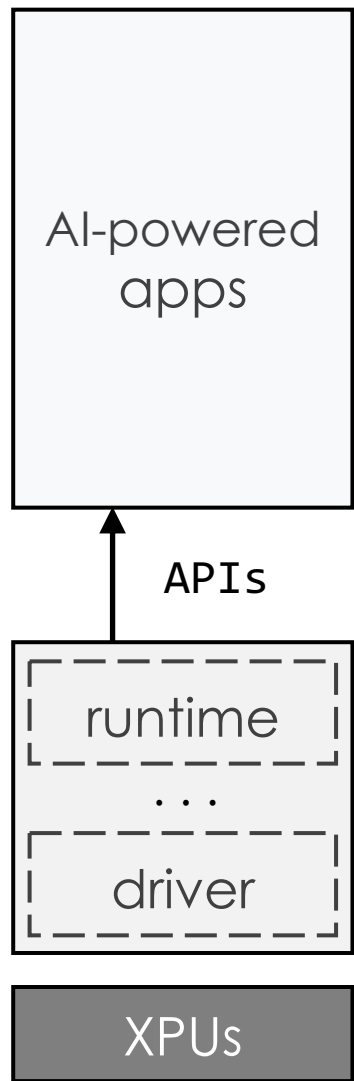
现有算力硬件（XPU: GPU/TPU/NPU/DSA/...）任务调度方法

- × 硬件调度：功能有限（First-Come-First-Serve）、仅适合大算力需求
- × 软件调度：绑定特定硬件（修改）、实现工作量大、方法迁移困难

我们的工作：面向开源GPU（修改驱动/运行时/应用）、5,500 LoC（C++）

关键技术：XPU通用调度框架

算力硬件任务调度现状



多任务调度问题演示



Linux-Fake-Background-Webcam
Star 1.6k

App #1
Linux Fake Background Webcam
Google mediapipe

whisper.cpp
Star 34.1k

App #2
whisper.cpp
OpenAI whisper

OpenVINO/oneAPI

intel CORE ULTRA 9
Intel® Core™ Ultra 9
CPU GPU NPU



ASUS NUC 14 Pro Plus

App#1: <https://github.com/fangfufu/Linux-Fake-Background-Webcam>

App#2: <https://github.com/ggerganov/whisper.cpp>

FPS: 3.98
[+1.59s] [BLANK_AUDIO]

FPS: 15.23
[+2.45s] This is my world. I've been coaching...

FPS: 21.69
[+1.56s] sure that your audience is really connecting.

Intel NPU Hardware Scheduling

Intel NPU XSched Scheduling

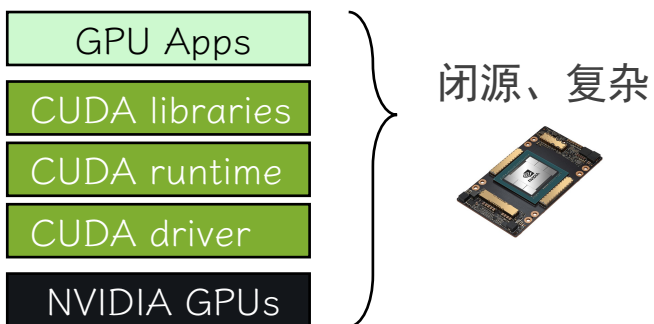
Intel GPU+NPU

问题/挑战

1. 如何提供XPU通用调度?

× XPU软件栈闭源、复杂

包括：驱动、运行时、应用

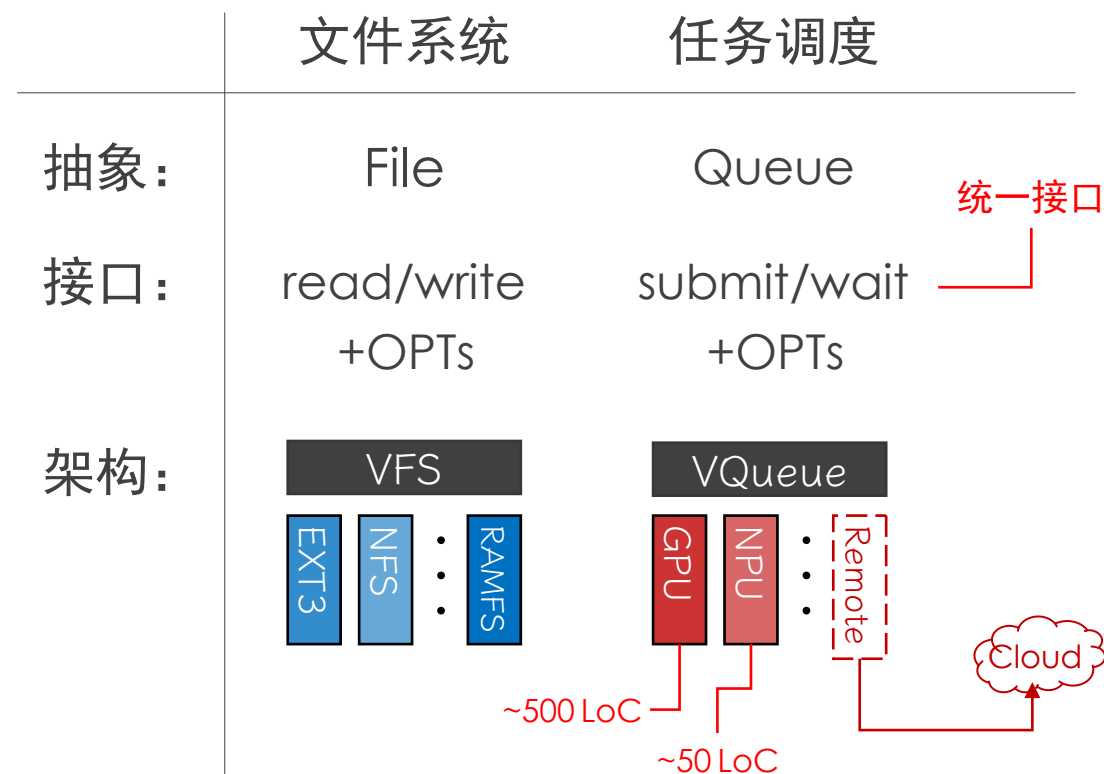


× XPU软件栈不兼容



思路/方法

关键思路：借鉴内核的统一抽象设计



问题/挑战

2. 如何支持不同种类XPU?

× XPU硬件架构差异大

GPU: CUDA/ROCm 程序

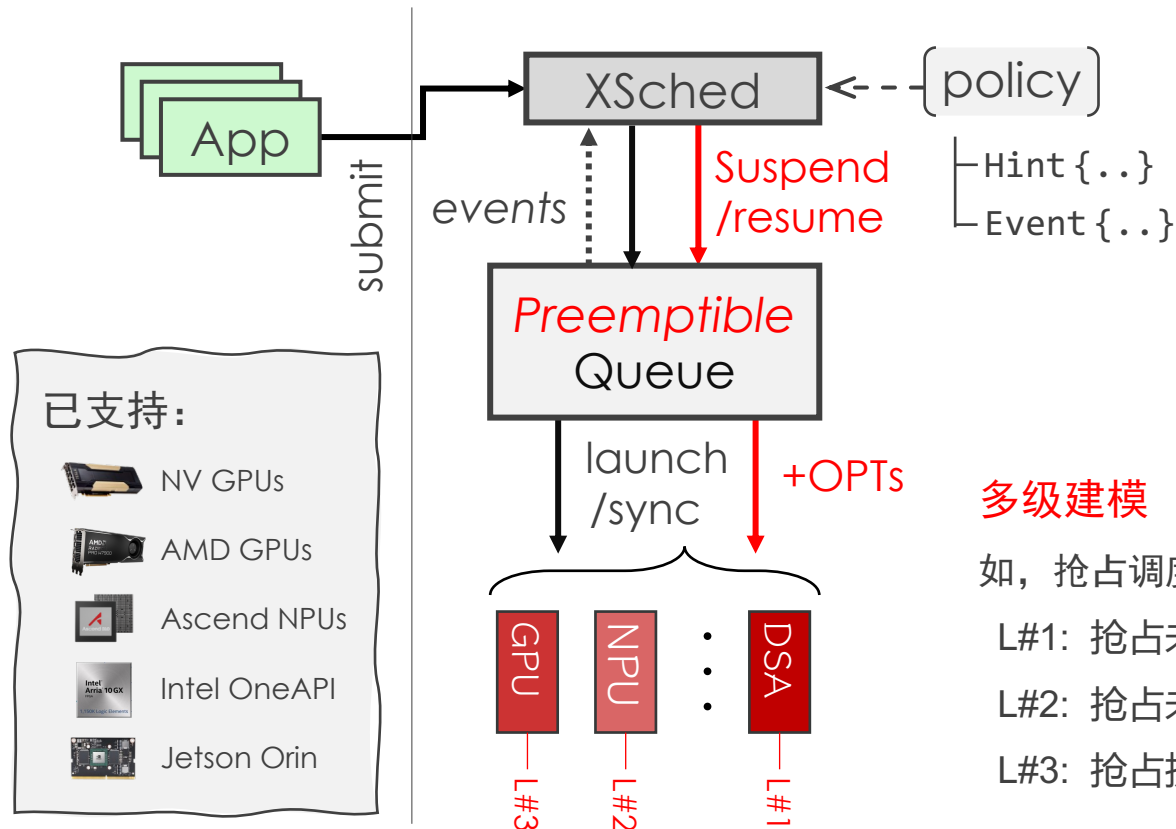
NPU: 预设命令/功能

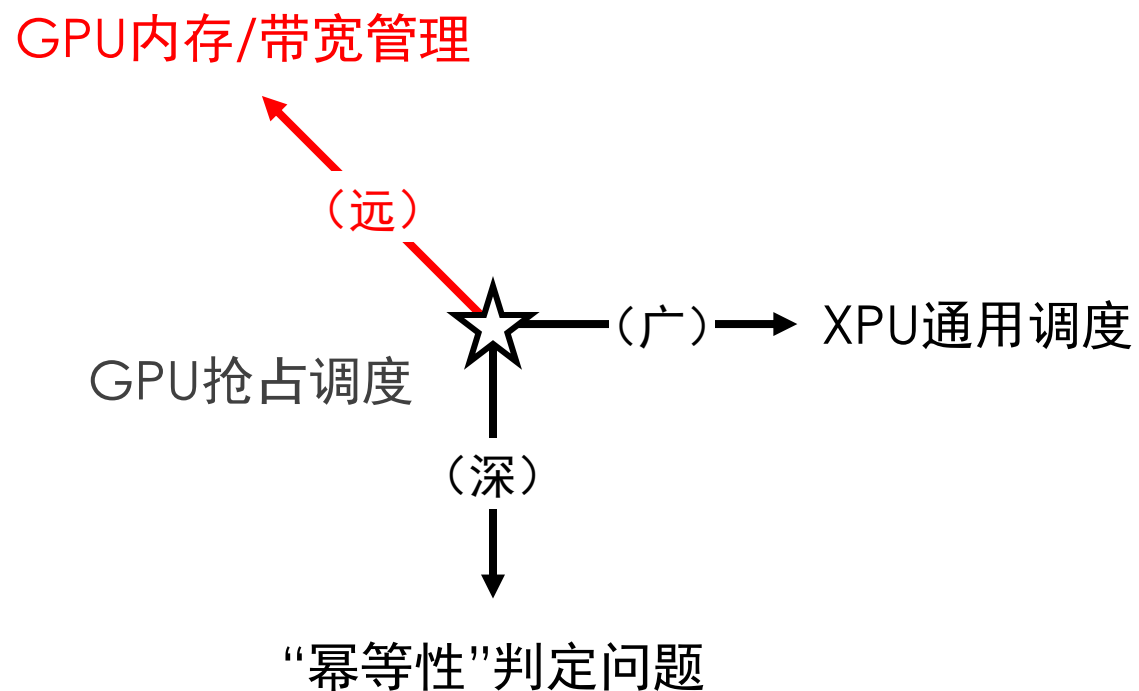
× XPU硬件实现差异大

- 提交/等待 (基础)
- 中断
- 计算单元重置
- 内存刷新
- ...

思路/方法

关键思路：基于能力的多级建模

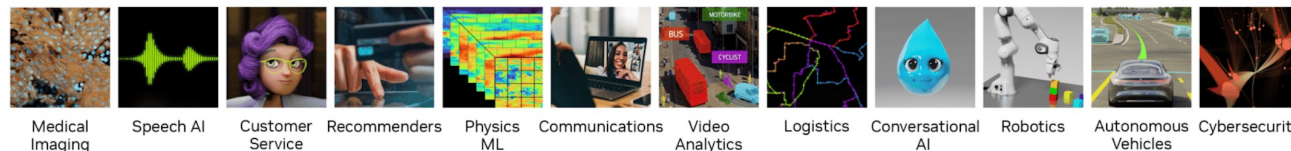




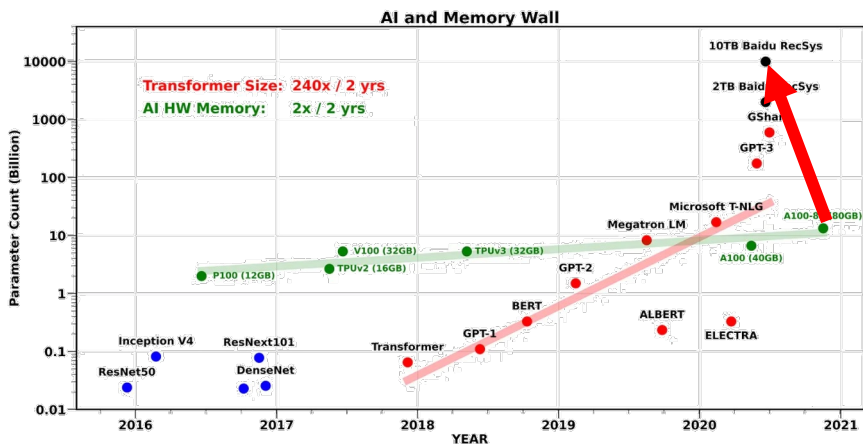
GPU内存/带宽管理



智能应用

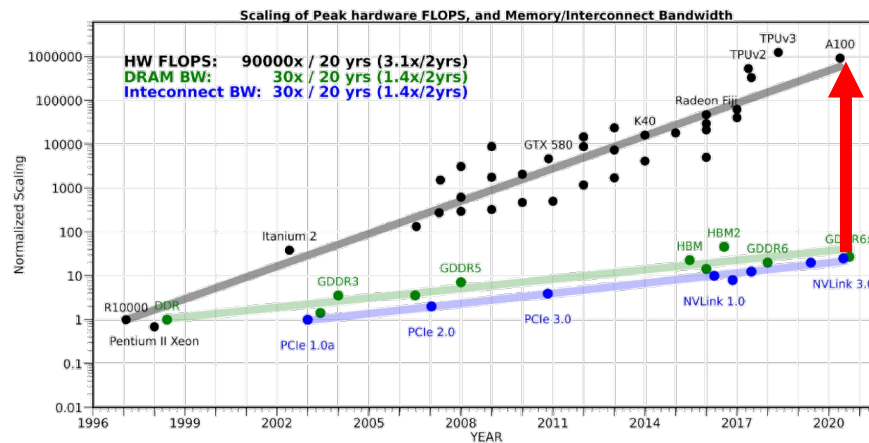


人工智能应用（尤其是DL/LLM）在内存/带宽上同样带来了爆发式需求增长



内存需求: 240X / 2 Yrs

内存容量: 2X / 2 Yrs



算力增长: 3.1X / 2 Yrs

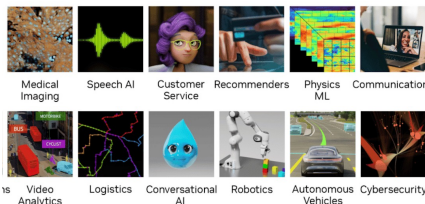
带宽增长: 1.4X / 2 Yrs (内存)

1.4X / 2 Yrs (互联)

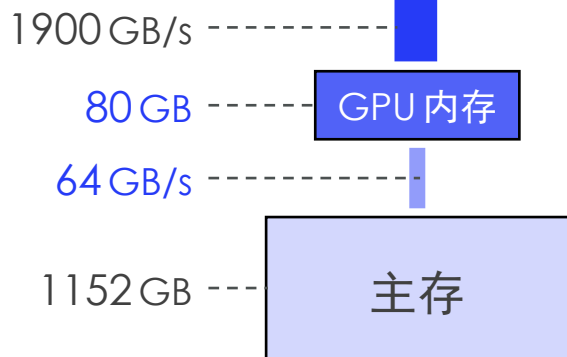
GPU缓存系统



智能应用



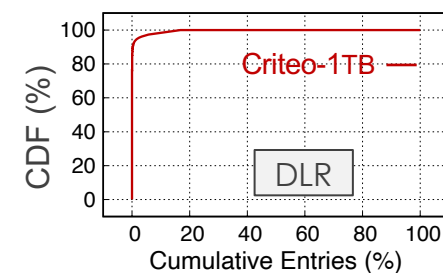
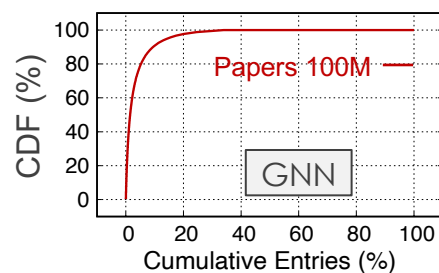
典型GPU存储架构



GPU缓存系统¹

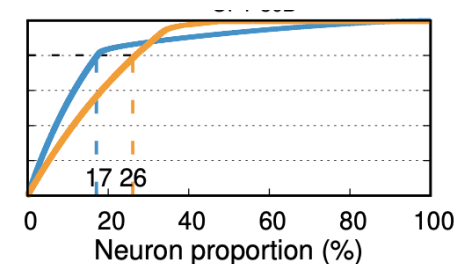
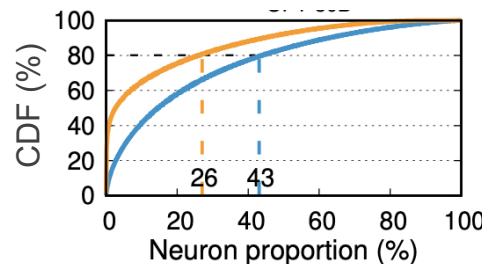
智能应用的数据访问普遍具有偏态分布特征

神经网络、推荐系统的嵌入 (Embeddings)



Source: UGache. SOSP 2023

LLM神经元激活 (Neuron Activation)



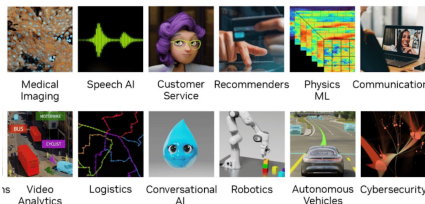
Source: PowerInfer, arXiv 2023

¹ GNNLab: A Factored System for Sample-based GNN Training over GPUs. EuroSys 2022

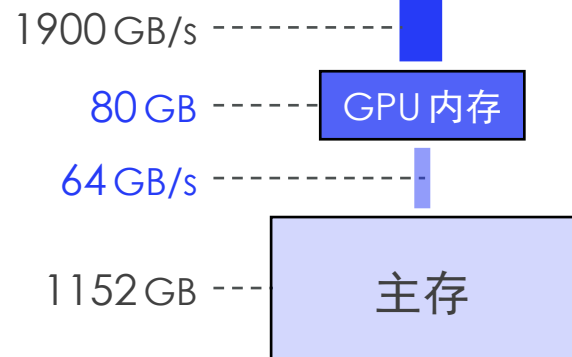
GPU缓存系统



智能应用



典型GPU存储架构

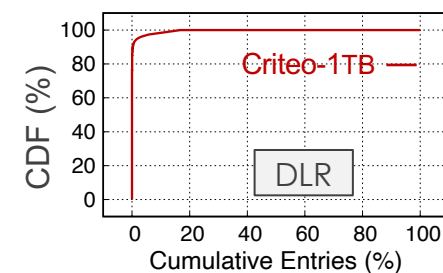
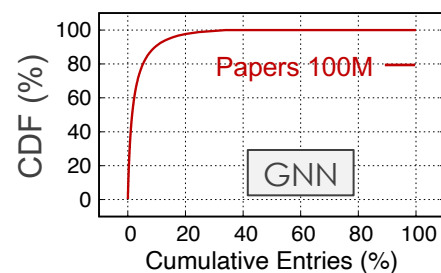


多卡GPU
+
高速互联

GPU
缓存系统¹

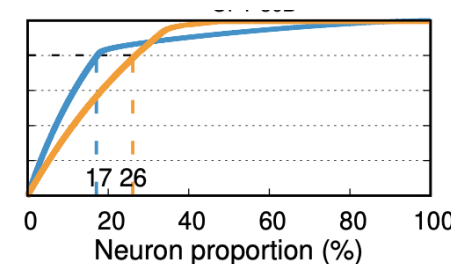
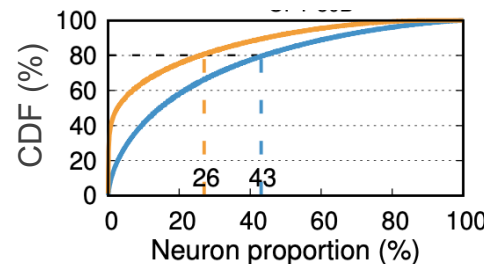
智能应用的数据访问普遍具有偏态分布特征

神经网络、推荐系统的嵌入 (Embeddings)



Source: UGache. SOSP 2023

LLM神经元激活 (Neuron Activation)



Source: PowerInfer, arXiv 2023

关键技术：多卡GPU统一缓存方法

¹ GNNLab: A Factored System for Sample-based GNN Training over GPUs. EuroSys 2022

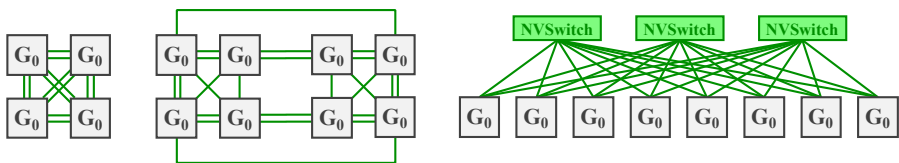


问题/挑战

1. 如何统一、高效的缓存数据？

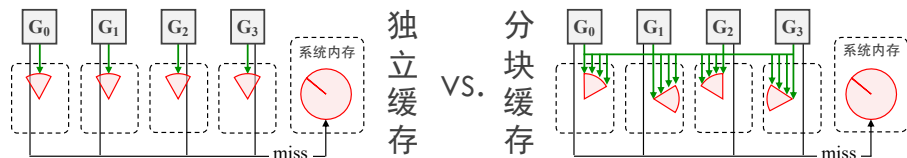
× 多卡GPU系统：互联架构多样

- 互联硬件：PCIe / NVLink / NVSwitch ..
- 访存性能：Local / Remote / Host ..



× 统一缓存抽象：放置策略复杂

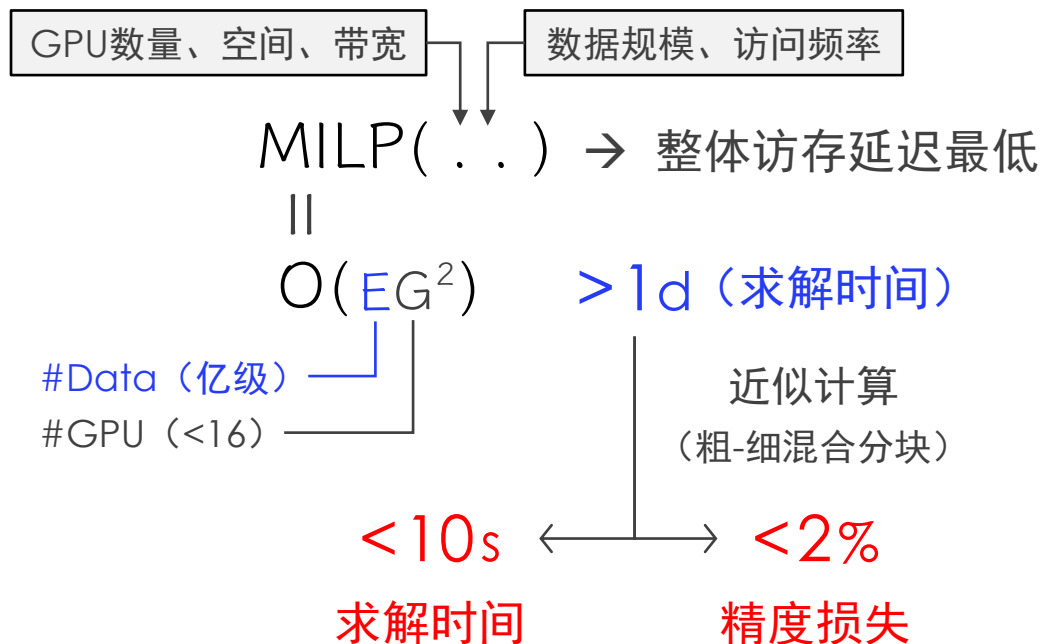
- 缓存什么数据、在哪里、...？



思路/方法

关键思路：对缓存数据问题统一建模

多GPU缓存问题 \Rightarrow 混合整数线性规划/MILP





“构建了面向多卡GPU架构的统一缓存系统、大大提升了缓存空间与访存效率¹”

GPU内存/带宽管理

(远)

GPU抢占调度

(深)

(广)

XPU通用调度

“幂等性”判定问题

¹ UGache: A Unified GPU Cache for Embedding-based Deep Learning. SOSP 2023



应用和硬件的发展演进是系统软件研究的**原动力**

“应用需求”与“硬件能力”是系统软件研究的**重要抓手**

“赋能赋智”带来**算力外需求**，亟需基础系统软件的**关键支撑**

我们的一些初步探索——**算力硬件调度与管理**

感谢！