Keynotes:

session 1: DNN related

记不得了，精彩的比赛，这个session我确实没有什么前置知识

可能还是要去修习一下陈云霁老师的智能计算系统

上交

Nearest Neighbor，这个东西似乎在任奎的云计算相关有点关系，在云计算领域和数据库领域有一些提及

这边似乎是打算做加速器，以面向LLM的新的应用场景

工作概览：IVF + PQ的设置下调优

把高维向量分解成多个低维向量，clustering化后，对其进行残差处理

效果：原来的一个D * 32位向量，在每一个子空间上，只需要D / M log2 * 32位向量表示

session 2: 存储

文宇鸿 华中科技大学

研究动机：存储元数据上的开销

消除重删系统映射表和Dedup-GC开销

设计目标：希望能够在消除重删系统存储管理模块开销的同时支持阵列级别重删

将FTL逻辑地址空间从设备级别扩展到阵列级

实现了一个阵列级别的跨盘重删方案

还有一个问题，怎样才能解耦重删系统？这方面可能存在一个旧的指纹无法同步更新的问题，可能会造成指纹与数据的不一致问题，可能导致错误重删与数据丢失

问题

FTL扩展了表的大小，会占据更大空间，可能会导致更多的FTL Miss，会不会影响性能？

- 这边的工作默认DRAM的大小是足够的，消除了主机表，虽然从单个区域来看，性能降低，不过从整个系统来看，性能有所提升

1T的SSD可能只有1G的内存，把FTL扩大了，这样是不是一个资源浪费

- 还是假设固态盘这边能够扩大大小，资源空间足够，不然确实工作的性能提升可能确实不明显

宫佳薇 人民大学

数据压缩，减小数据规模，缓解存储与处理的开销

在压缩领域的趋势：先解压缩再计算 -> 使用新的算法和结构，最好在能够直接处理、直接在压缩后的数据上计算

目前的工作可能还是有很大限制

1. 缺乏对于压缩的共识与理解，需要做算法性质相关的筛选
2. 系统层面，可以支持计算实现的，面对的场景不一而足，需要深入理解每个压缩算法的深入机制
3. 算法层面，可能和1差不多

借用了同态加密的概念，（其实感觉还是直接是同态概念，我咋感觉和加密没啥关系？）

场景：文本结构

哦哦，还是用了同态加密的一些原语

在文章中尝试了三种压缩方法，构建了一个叫做HOCO的同态压缩系统

算法利用cpp面向对象来进行组织，并同时设计Evaluation模块评估当前同态压缩状态

问题

现在的同态压缩，支持文本的正则匹配吗？

- 代价可能比较大，计算代价可能比较大，懂的都懂，不然要搞同态加速器干嘛

向一文 兰州大学

通过文件迁移和预加载来提高效率

问题：

1. 文件预取机制有一些问题，即便是小取，也得取大的块
2. 可执行文件启动的读地址是分散的

blktrace工具值得学习，简单验证了一下上面的问题

结果发现碎片文件的延迟高于顺序文件的延迟，可不可以把碎片化文件预先放到一个顺序区域，从而提高运行效率，减小延迟

标记迁移应用程序文件访问位置，将其顺序化

利用好F2FS

问题

离散的问题，有没有考虑可执行文件，现在的linker可能已经能够用更大粒度去组织了？这靠谱吗？

- WIP，会去考虑

离散的问题，占到了怎样的一个开销或者比例

- 正在测试

吴镕龙 厦门大学

首先，NVM是一个好东西

但是其有一个Write Disturbance问题

MinHash，二元局部性哈希算法，表示数据相似性

session 3: 加速器

这个我是真不会

简要记录一下新词：

GPGPU是用于科学计算，AI训练，推理任务矩阵计算的，和用于图形渲染。图形计算的GPU存在差异

热设计功率：TDP等等

session 4: Severless & Quantunm

李一鸣 天津大学

serverless：服务器无感知计算

把服务器运维的责任交给云服务器开发商，开发者只需要搞应用程序就行

Serverless = FaaS + BaaS

function as a service

backend as a service

Serverless的好处

- 程序员没有运维的责任
- 弹性伸缩
- 用时付费
- 按需分配资源

因为这些优点Serverless目前发挥着较好的作用

Serverless的问题:

1. 冷启动问题：生成容器，冷启动开销是秒级，不可承受
2. Serverless似乎在函数的通信方面需要第三方转发，会带来很大的延迟

Motivation:

- High Latency: one-to-one 一对一，每个函数分配单个沙箱，的调度与交互开销巨大；多对一一样有很大问题，主要编程语言是python和nodejs，对于多进程计算抽象仍然很大，尤其是对于并行度高的应用；那么m-to-n是否可以？似乎有一点性能提升，但也不是一个很好的方案，但凡用上多线程，似乎没法避免资源浪费

提出了一个wrap-based deployment system -> chiron系统

工作流程：

参考python GIL机制，使用strace，使用系统调用阻塞来模拟GIL阻塞

使用预测器根据指标分配资源

使用特殊算法进行调度

采用MPK给线程做好隔离，以缓解安全问题

应该是一篇不错的文章，可以考虑阅读

实验结果，在没有GIL锁的时候，依旧有较好的吞吐率

这篇文章提出了一个M-to-N的部署系统

问题：

调度时间是怎么测量的？

- 根据日志，和AWS static function有关，它的日志记录的非常详细

吴明瑜 上海交大

对于高级语言虚拟机部署在Serverless的现象的研究

Serverless更偏向于用户不感知

FaaS更偏向于轻量级部署，小段代码片段的使用

一般认为执行单元是一个容器或者虚拟机

执行单元的创建往往会带来秒级的开销，是很大的

冷启动现有缓解方法：重用，缓存（商用平台较多），checkpoint

保留也会带来问题，后台的函数不管，只管了前台的函数情况，这样可能会偷到云平台的算力。

为了避免这个问题，现有FaaS平台会采用冻结已缓存执行单元以管理器CPU资源占用

上述冻结方式观察：

1. 无用冻结内存：执行长时间Java函数，每次执行的时候只是看一下实际上有多少内存是被用上的，结果发现只有40%是用上的，大部分内存被垃圾对象给占据了。这样可能会造成更多的冷启动。
2. 冻结内存不局限于JVM，还存在其他类型的语言虚拟机，python一样有类似的问题。
3. 垃圾回收操作难以有效回收无用冻结内存，不太行。原因1：垃圾回收的目的默认之后我们还要继续进行，但我们这边会冻结，之前垃圾回收的预留内存会浪费。原因2：现有缩容机制与间歇式的执行模式不匹配
4. 资源浪费量越来越多

工作：

做了一个低侵入式内存回收激活，在后台监控内存的使用量，主动去做垃圾回收。相当于把原来后端忽视的地方拿起来用。

被回收单元的选择，去选择高性价比：回收时间越短，回收单元越多，打分回收。回收的效率动态估计

内存回收：调动现有python等语言虚拟机的回收内存的接口

优化：

1. 堆外内存，共享库如果被一个函数独占，则释放器占用的内存
2. V8垃圾回收太激进，影响性能，简单调整了一下

评测

有一个细节可以注意：将微软真实环境的测评函数，在我们测试平台中找一个表征最像的来做。

谭思危 浙江大学

量子计算

量子电路编译框架方向

背景知识：

量子计算，通过量子力学微观原理来做事情

优势： 指数增长的计算能力，但是可能会有量子坍缩问题，导致量子信息读取会有较大的代价

流程： 基于量子电路模型，使用超导量子等来做，和0/1半导体，CMOS管存在本质差异

量子电路实现最终造出来的芯片更像是一个FPGA

编译的流程则是会把一个量子程序编译成波形层，然后再变成板卡指令编译，最后才会在量子真机上跑

任务1：量子酉矩阵分解 任务2：噪声较大，要做保真度优化

量子电路编译方法目前的挑战：速度与质量不可兼得

提出了QuCT方法，实现电路里拓扑和上下文感知的电路向量化

这个方法在传统领域上NLP上已经有了，但是量子领域似乎没有人之前做过

于是照着NLP的方法来做，最后发现能work

观点：你得有比较好的基础和灵活的大脑

整个流程很像是在system领域玩NLP，不过还是需要严格的物理上的证明。

问题：

您认为以后量子计算的存在形式是什么样的？

- 更倾向于是他会是一个类似于GPU的外设，对于特定问题上有良好的加速效果

您认为有成为通用计算机的可能性吗？

- 理论上是有的，不用量子叠加性，通用计算机在理论上可以成为他的子集，不过还是有噪声过大、指令较长、花很长时间的问题。

后记：

交大的同学很擅长提问，大部分提问的都是他们