Key notes

Session 3: databases

本科没有开设数据库课程，心有余而力不足。。

朱镕 阿里巴巴

paper 1:

现有很多端到端的查询规划器，但是有一个挑战：

time prediction is very difficuly and inaccurate.

我们是不是一定要知道每个plan所需的具体执行时间？我们可能不需要？我们只要一个order，直到谁比较好就可以了。

From learning-to-predict to learning-to-rank

相对于learning-to-predict来说，rank是一个比较自由的东西，可以降低learning difficulty，更新速度与学习速度也会变快。

可以通过预训练的方法，让它能够避免冷启动问题。

paper 2:

PilotScope，解决对于数据库和AI算法之间部署困难的工作

真正把AI算法放到数据库系统中，存在很多问题。

这个工作是希望能够做解耦，让ML和DB程序员不必互相了解。

郝宗寅 厦门大学：

DBMS逻辑错误，结果上出现差异，但是并不会出现崩溃。那我们应该怎么去检测这种逻辑bug呢？

蜕变测试：不需要在多个DBMS中运行输入的SQL。

Session 4: Operating System

曲虹亮 深先院：

目前的服务器采用NUMA架构，对于本地内存速度快，对于远端的内存访问速度慢，这个现象被称为NUMA现象。

2020年对于页表的访问也出现了NUMA现象，因为内存很大了，导致TLB miss增多，去内存中查找页表的次数也变多。

为了缓解，有一个PTSR技术，叫做页表复制，保证程序使用本地页表副本。

页表复制的收益其实会收到其他程序的干扰影响，NUMA也会出现阻塞现象。在本地内存控制器阻塞情况下，本地延迟比远端延迟还要大。

在2 NUMA节点服务器上做实验发现这个现象也存在，即在本地延迟比远端延迟要大时，就还是用远端延迟可能比较好。

总结，不是所有的页表复制都是好事，而且可能需要用户手动控制。

目标：提出一个自动配置页表解决方案。

首先记录相对延迟最低的节点记录下来，根据MAR和PTL以及TLB miss rate等指标来确定一个决策机制。

确定阈值，做好PTL测试等等。

WASP用户工具因此被设计出来，持续测量三个性能参数，并周期性地进行决策。

问题：

多NUMA节点的pagetables的一致性怎么确定

- （可能是NUMA的特性）有一个循环链表，自动能够实现更新，并且overhead也不是很高

CXL-based

一旦CXL被使用，那就能直接在它上面构建高效的分布式共享内存

把CXL_MALLOC扩展到多线程场景上，构建一个远程共享的系统

但是基于CXL来做的也有挑战，对于部分结点意外崩溃，希望他们不会影响到其他节点。

如何进行分配和回收，他们可能没有正常地释放。

拆分引用计数计数为两个部分：

- ModifyRefCnt: 引用计数
- ModifyRef: referd or NULL

但是这两个是独立的，可能还会引发新的问题。

常用的方式可能是看LOG，但是没有办法去重构，不太清楚具体报错的位置。

方法一：

利用锁来做，全局阻塞，太蠢了，我们希望容忍部分故障。

想法：

应该怎么去做？应该如何确定modifyrefcnt过程中是否被执行了？

=> 原子性地对某一个字段进行原子性维护。

构建一个矩阵，进行错误维护，以获得时间戳和存储时间位置戳，这里还是有比较复杂的细节。大致通过这个来判断，是否值得我们redone。

Session 5: Storage System

董明凯 IPADS

既不用去关心持久化，又有较好的性能，如果能做到，那是最好的。

checkpoint要与下面的内存保持一致

在微内核下，整个操作系统变成了一个树。

1. use persisting memory
2. use microkernel

这篇文章的pre讲的很好，感觉确实有必要去看看chcore和这个项目

沈逸凡 阿里巴巴

完了，网络又不会了。

前端网络的部署有一些问题？

kernel tcp的perforemance和throughoutput都是巨大的开销，没法满足产品的需求。

Session 6: 产业论坛

华为云存储

- 分离式内存，找到杀手级应用
- 单卡TB级互联带宽催生紧耦合系统，松耦合系统和紧耦合系统深度结合
- 新的体系架构下，云存储系统通过DSA的思想打通极简路径，构建极致性能/减少开销，消减软件税

杀手级应用还是蛮重要的，分离式内存发展的时间确实已经很长了。

AI系统中对分离式内存的研究持续增加

产业论坛的总结：

- ai4sys主题，较为关注新AI对于系统资源调度上的研究，产业界认为这个比较有价值
- 大模型虽然更倾向于在服务器端做部署，但是端侧其实还是有点想做的
- sys4ai的挑战与需求普遍存在，算力的发展远远落后于需求的提高。端侧资源难以部署大模型，存在AI的存储墙问题

具体可以参考产业论坛手册中的问题总结