中山大學 SUN YAT-SEN UNIVERSITY

国家超级计算广州中心 NATIONAL SUPERCOMPUTER CENTER IN GUANGZHOU

The 2020 China Lustre User group

纵向可扩展元数据管理

陈 志 广

Zhiguang.chen@nscc-gz.cn

中山大学数据科学与计算机学院
国家超级计算广州中心

报告内容

一、文件系统的元数据瓶颈

二、文件系统数据结构优化

三、面向元数据的计算加速

四、总结

元数据访问瓶颈

针对文件系统的IO请求中，一半以上是元数据访问

	Corporate	Engineering
Clients	5261	2654
Days	65	97
Data read (GB)	364.3	723.4
Data written (GB)	177.7	364.4
R:W I/O ratio	3.2	2.3
R:W byte ratio	2.1	2.0
Total operations	228 million	352 million
Operation name	%	%
Session create	0.4	0.3
Open	12.0	11.9
Close	4.6	5.8
Read	16.2	15.1
Write	5.1	6.5
Flush	0.1	0.04
Lock	1.2	0.6
Delete	0.03	0.006
File stat	36.7	42.5
Set attribute	1.8	1.2
Directory read	10.3	11.8
Rename	0.04	0.02
Pipe transactions	1.4	0.2

元数据访问特点

- ✓ I/O大小较小（元数据大小通常只有几百Byte），使得元数据服务器CPU的负载重
- ✓ 许多元数据操作包含多次子操作，例如打开文件需要进行多次的路径解析，使得元数据操作会触发多次网络I/O

Leung, A. W., et al. Measurement and Analysis of Large-Scale Network File System Workloads (ATC'08)

分布式元数据解决方案

- 当前，主流并行文件系统大多采用分布式解决方案
  - Lustre、CephFS
    - ✓ 子树划分+目录条带化
  - BeeGFS
    - ✓ 根据名称的Hash值分布到多节点
  - GlusterFS
    - ✓ 无中心架构
  - IndexFS
    - ✓ 元数据以KV存储
    - ✓ 通过GIGA+算法分布到多节点

分布式元数据解决方案

分布式解决方案普遍面临的问题

分布式本身的问题

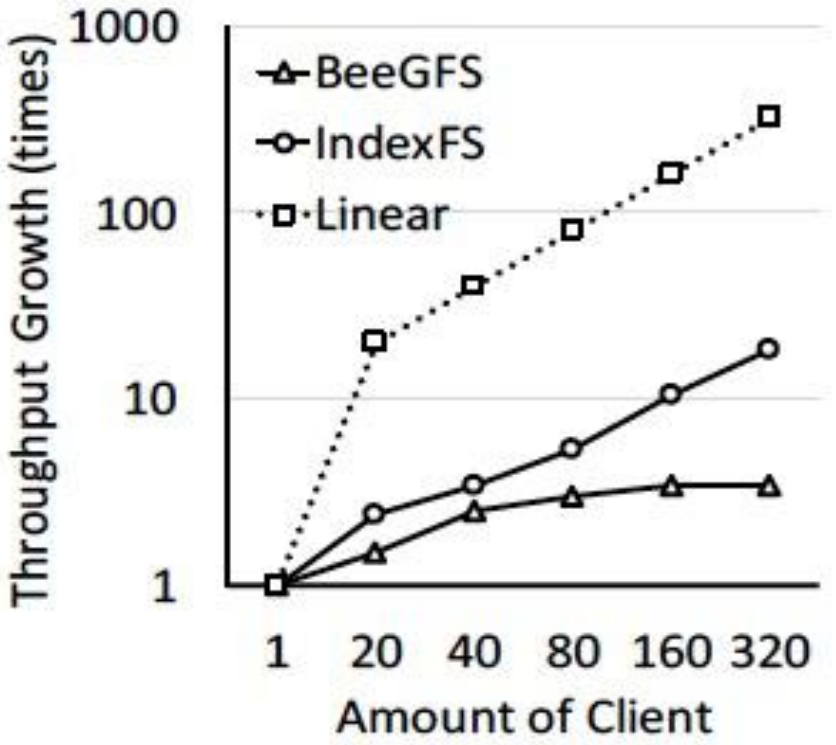
- ✓ 跨节点通信
- ✓ 节点间的一致性

树形名字空间的问题

- ✓ 一个节点失效，会传导到其它节点，可靠性降低

元数据服务扩展性不足

- ✓ 需要大规模的元数据集群来支撑系统峰值
- ✓ 扩容难度大



单节点上元数据性能仍有提升空间

- 现有文件系统元数据管理的潜在缺陷
  - POSIX的语义限制了元数据性能提升
  - 数据结构不利于并发处理
- 元数据性能提升面临的机遇
  - IO设备跨越式发展
    - ✓ NVMe SSD充裕的IOPS和带宽
    - ✓ 多通道并发能力
  - 计算部件性能显著提升
    - ✓ CPU向量指令
    - ✓ GPU众核结构

报告内容

一、文件系统的元数据瓶颈

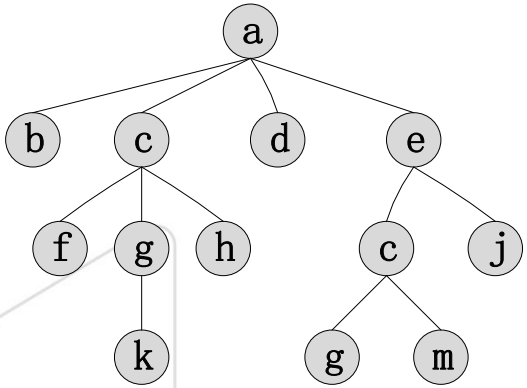
二、文件系统数据结构优化

三、面向元数据的计算加速

四、总结

树状名字空间的限制

- **树状名字空间组织方式限制了并发处理能力**
  - **VFS的实现要求逐级解析目录路径**
    - ✓ 路径解析a/c/g/k：依次解析a、c、g、k各个分量
    - ✓ 随着路径层次加深，性能线性下降
  - **在分布式元数据解决方案中，还导致跨节点交互问题**
  - **一些优化方案：学术研究多，业界采用少**
    - ✓ 全路径哈希
    - ✓ 路径前缀的Key-Value存储



树状名字空间下的并发路径解析

将树状名字空间看作图

顶点存储

- ✓ 每个父目录保存所包含的子目录：维护POSIX语义

边存储

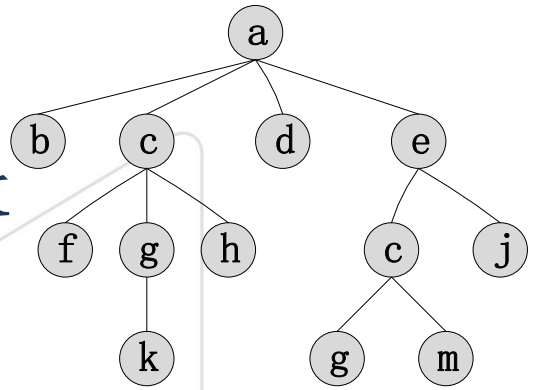
- ✓ 记录所有的父子关系：并发解析加速

(a,b)
(a,c)
(a,d)
(a,e)
(c,f)
(c,g)
(c,h)
(g,k)

边存储

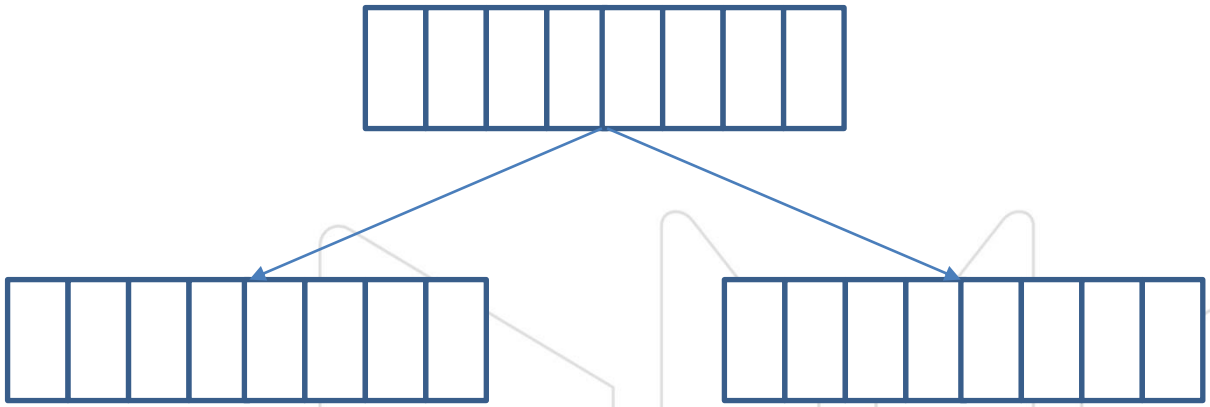
并发目录路径解析

- ✓ 并发解析a⊕c、c⊕g、g⊕k等各个边，然后拼接起来
- ✓ 若能拼接成功，则路径解析成功
- ✓ 理论上：无论路径深度多长时间为常数



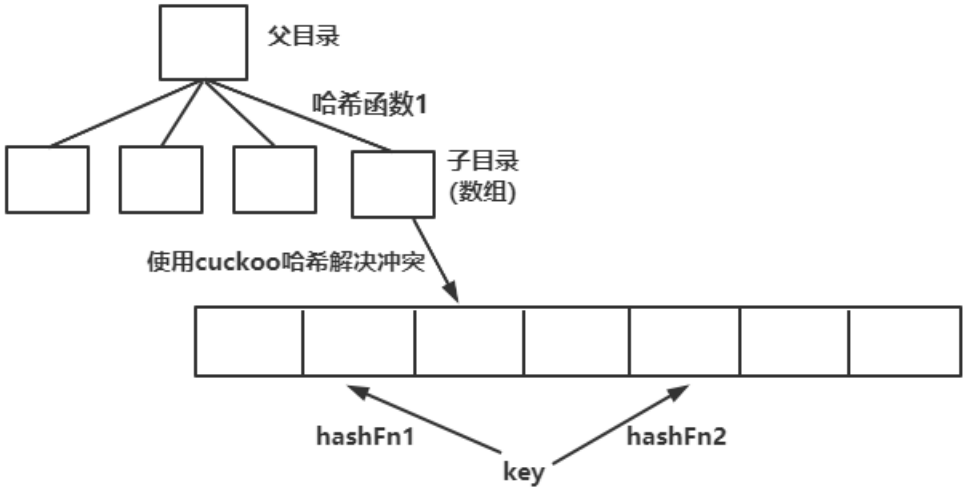
大目录面临的挑战

- 大目录内部的Entry索引方式导致大目录性能低下
  - EXT4：采用两级HTree索引所有的子目录
    - ✓ 一次文件访问至少两次IO操作
  - BtrFS：一切对象采用B+树索引
    - ✓ 一次文件访问多次IO操作
  - 多次IO操作之间存在依赖关系，必须串行执行



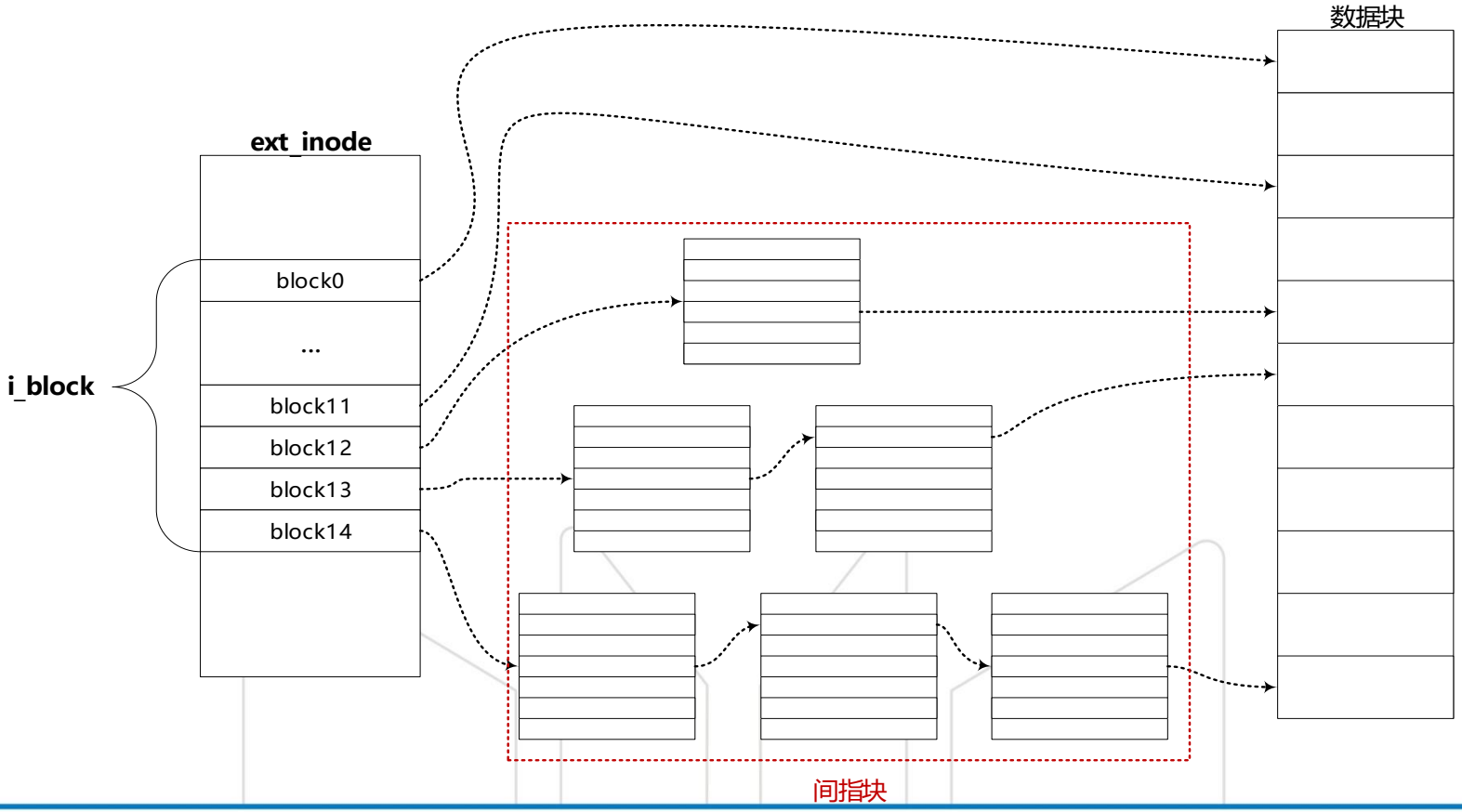
基于Cuckoo哈希的大目录组织

- 采用Cuckoo哈希索引同一目录下的所有子目录
  - 本质上是哈希
    - ✓ 查询的时间复杂度是常数
  - 每次文件访问对应两次IO
    - ✓ 设计合理哈希函数，两次IO距离足够近，可合并成一个大IO



多次间址带来的延迟

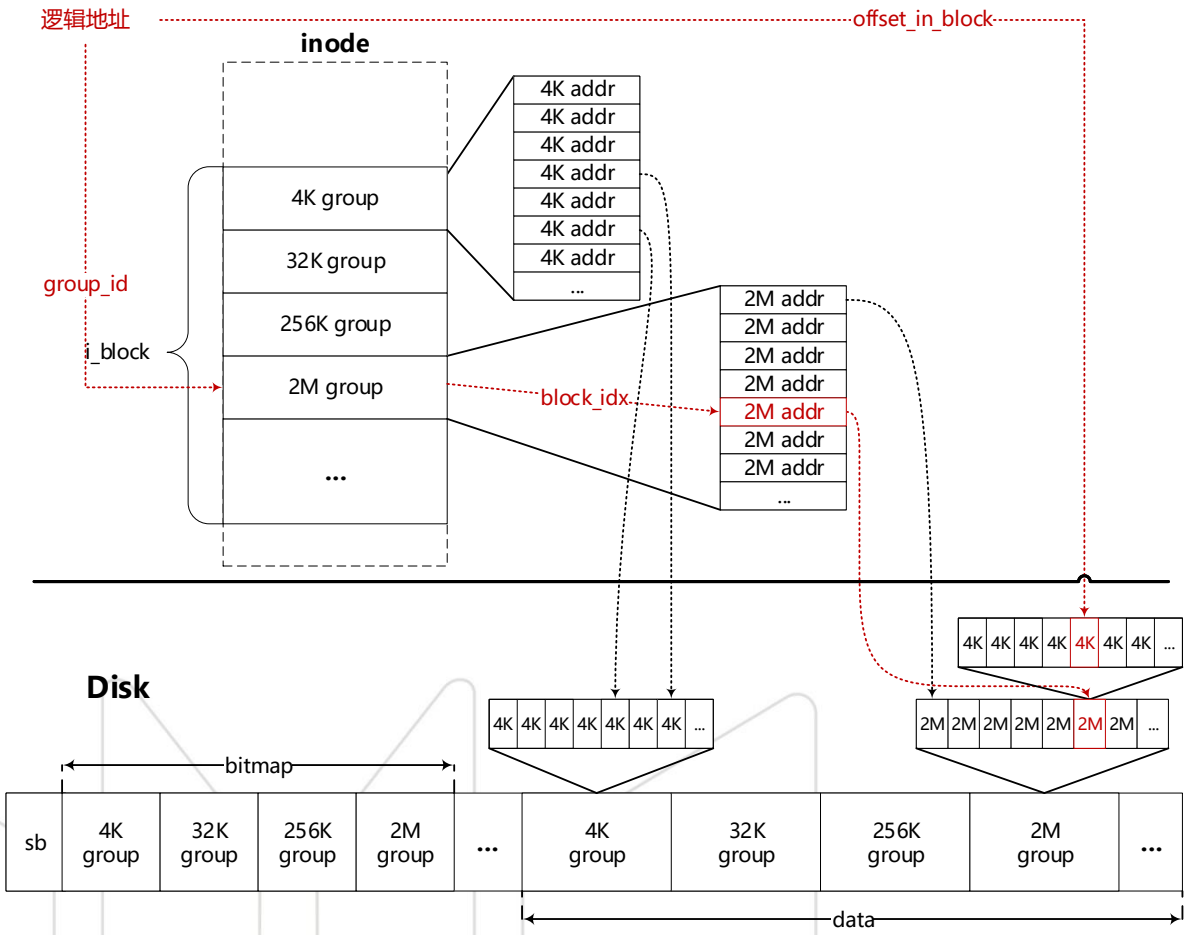
- 属于同一文件的大量数据块采用复杂索引结构
  - EXT4：三次间接寻址
  - BtrFS：B+树索引



基于变长块的磁盘地址空间管理

采用变长的数据块减少数据块索引所占的空间

- 磁盘数据块按一定递增比例划分为大小不同的块组，数据指针按各块组一定数量分布
- 通过数据指针分布直接将逻辑地址换算成物理地址，只需1次IO直接读取数据块
- 大数据块有较好空间局部性



报告内容

一、文件系统的元数据瓶颈

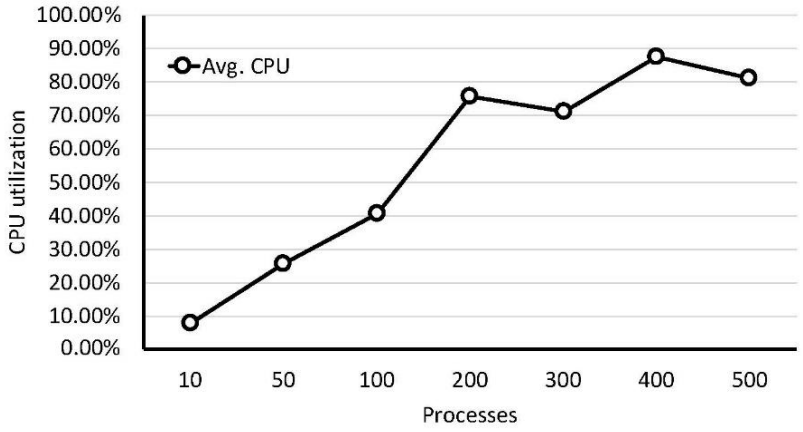
二、文件系统数据结构优化

三、面向元数据的计算加速

四、总结

元数据瓶颈由IO向计算转移

- 新型存储设备的IO性能显著提升
  - NVMe SSD： IOPS 50万以上，读写带宽2GB/s以上
  - 3D-Xpoint：读写延迟接近DRAM，IOPS 200万以上
- SSD上的元数据性能与内存中元数据性能相当
  - Ramdisk：192.437K IOPS
  - NVMe SSD：207.683K IOPS
- CPU成为重要的元数据瓶颈



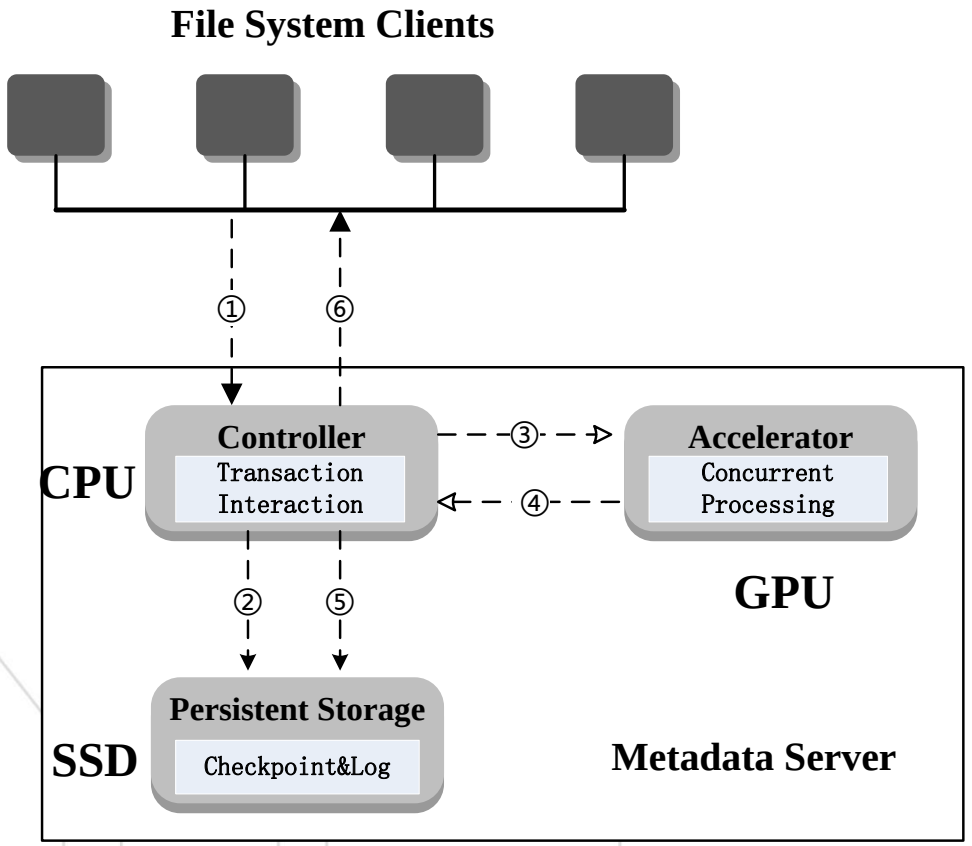
文件系统没有享受计算红利

计算部件性能提升的主要手段

- **CPU主频提升：2.2GHz→2.8GHz**
  - ✓ 定点计算能力提升，元数据性能可相应提升
- **CPU向量指令：128bit →512bit**
  - ✓ 文件系统没有利用向量指令
- **GPU的众核处理：V100 5376 INT32 cores**
  - ✓ 文件系统没有利用GPU加速

基于GPU的元数据加速

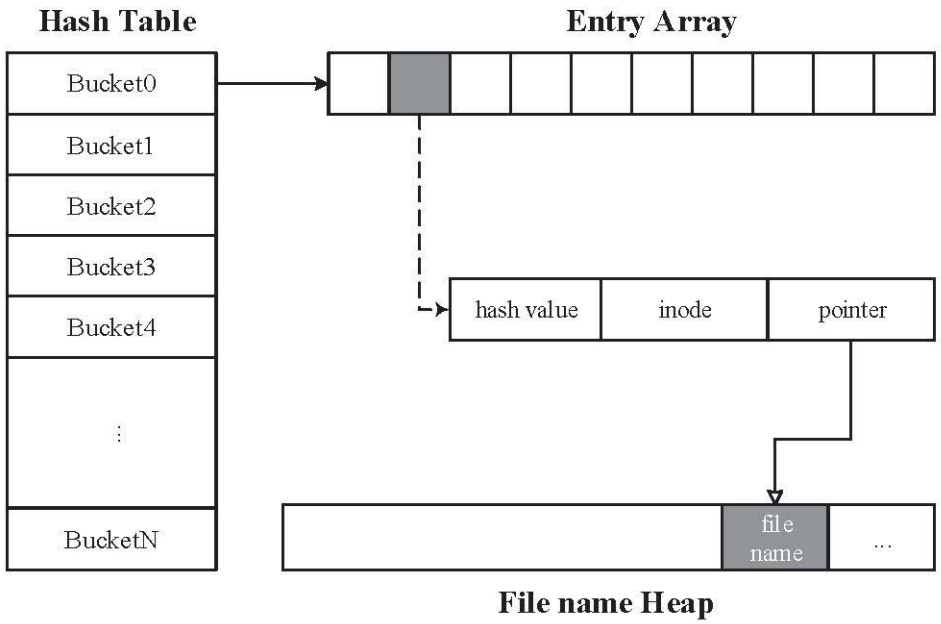
- 基于CPU的控制器
  - 与客户端交互
  - 元数据请求打包发送到GPU
- 基于GPU的加速器
  - 并发处理元数据请求
- SSD持久存储
  - 日志
  - Checkpoint



基于GPU的元数据加速

面向GPU的数据结构设计

- 采用平面化的哈希表，适合SIMT处理
- 文件名从Entry中剥离出来，确保定长数据结构



基于GPU的元数据加速

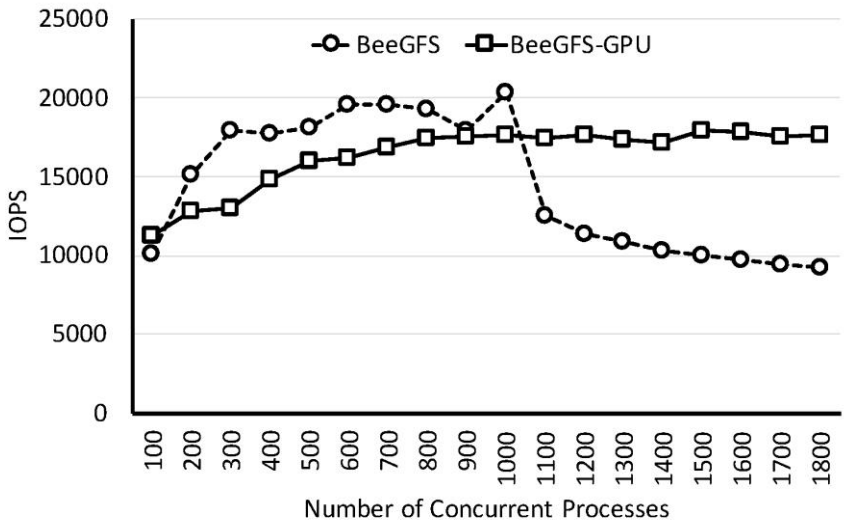
以可控的延迟换取更高的IOPS

缺陷

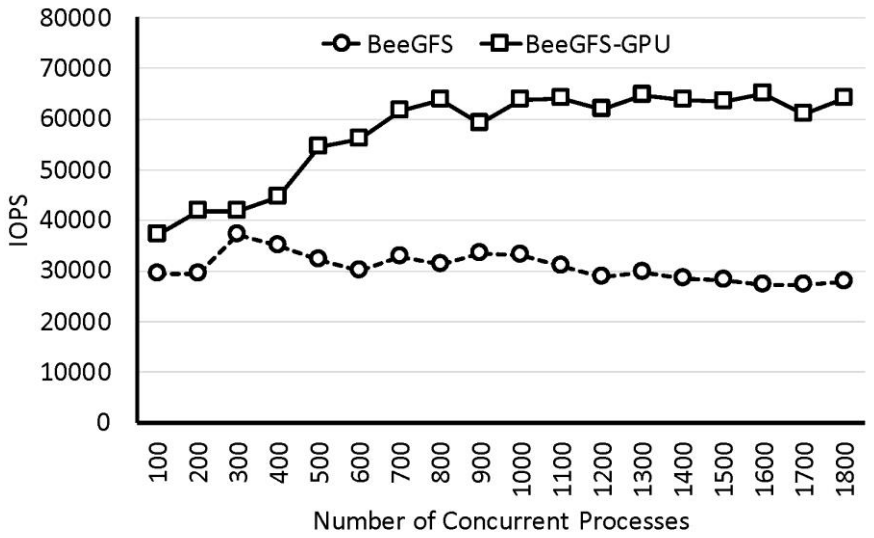
- ✓ CPU打包元数据请求引入额外的毫秒级延迟
- ✓ GPU的内存容量成为潜在的限制因素

优势

- ✓ CPU解放出来专注于处理网络请求
- ✓ GPU的元数据请求并发处理



Create



Stat

报告内容

一、文件系统的元数据瓶颈

二、文件系统数据结构优化

三、面向元数据的计算加速

四、总结

总结

- 元数据一直是文件系统的重要瓶颈
- 单点元数据性能仍然存在很大的提升空间
  - 没有充分发挥IO效能
  - 没有充分发挥计算效能
- 我们的思路
  - 改进数据结构，降低IO之间的依赖，发挥并行IO能力
    - ✓ 降低目录路径解析延迟
    - ✓ 降低大目录访问延迟
    - ✓ 降低数据块访问延迟
  - 改进算法，发挥计算部件的并行计算能力
- 分布式解决方案也能受益于单点元数据性能提升

感谢各位专家

敬请批评指正