Lustre User Group 2013 | China and Japan
Hosted by OpenSFS
Beijing - October 15 **Tokyo** - October 17
Sponsored by: intel

Lustre*
Fastforward to Exascale

梁震
英特尔
2013年10月15日
Liang.zhen@intel.com

美国能源部(DOE) – FastForward的挑战

- 美国能源部FastForward的需求方案说明书(RFP)提出对百亿亿次运算(exascale)的研发工作提供资助
  - 解决目前exascale运算上各种难以逾越的障碍，在2020年实现exascale运算集群的目标
  - 资助方向：CPU, memory 以及 I/O系统
- 美国7所最主要的实验室发起
- Whamcloud与EMC, Cray, HDF的共同方案赢得了I/O的竞标
  - 由上至下重新设计与构造HPC I/O栈
  - HDF组：HPF5的修改与扩展
  - EMC：突发缓存的管理以及I/O调度器的设计与实现
  - Whamcloud：Distributed Application Object Storage(DAOS)
  - Cray：测试环境
- 完成并购后的一些变化
  - DDN（版本化的OSD），英特尔（任意连接图ACG应用）

Exascale I/O 的技术驱动

	2012/2013	2020
节点数	10-100K	100K-1M
每节点线程数	~10	~1000
全局并发度	100K-1M	100M-1B
对象创建性能需求	**100K/s**	**100M/s**
总内存	1-4PB	30-60PB
文件系统总空间	10-100PB	600-3000PB
平均故障时间	1-5 Days	6 Hours
内存转储	< 2000s	< 300s
峰值 I/O 带宽	**1-2TB/s**	**100-200TB/s**
持续 I/O 带宽	**10-200GB/s**	**20TB/s**

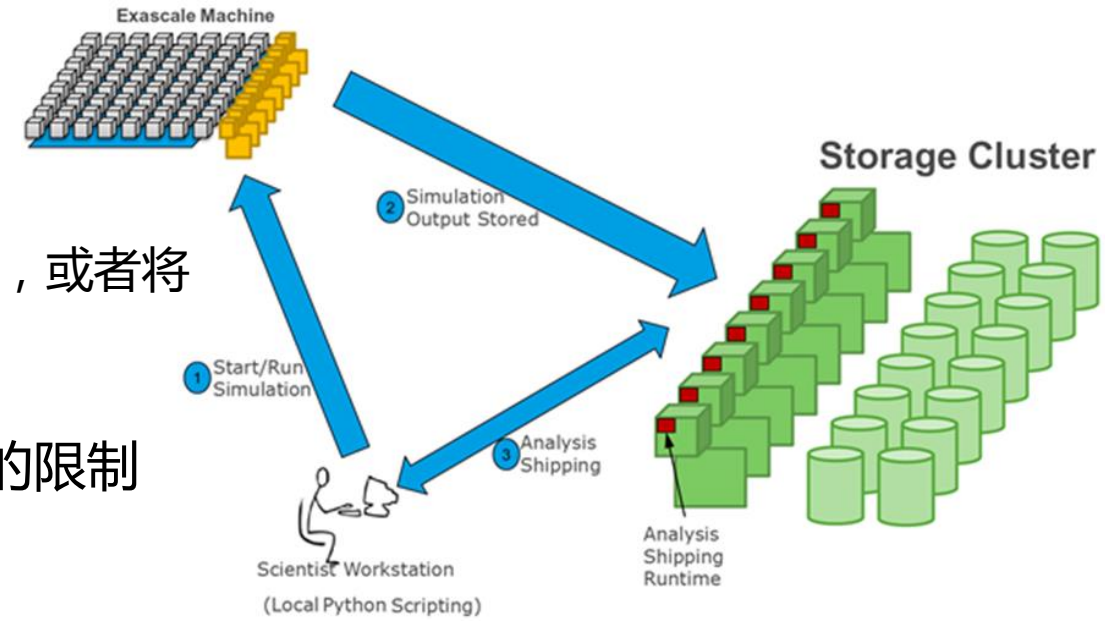
Exascale I/O 应用驱动

- 数据以及元数据的爆炸式增长
  - 更复杂庞大的模拟程序
    - 不确定性量化(Uncertainty Quantification)的同时运行
  - 数十亿的数据条目
    - 网格元素 / 图节点
  - 极其复杂的关系

- 面向对象的数据库(OODB)
  - 读写 -> 实例化/存储
  - 快速/即席查找： “百年一遇的大浪将在哪里发生”
    - 多索引
    - 分析传送

项目的目标

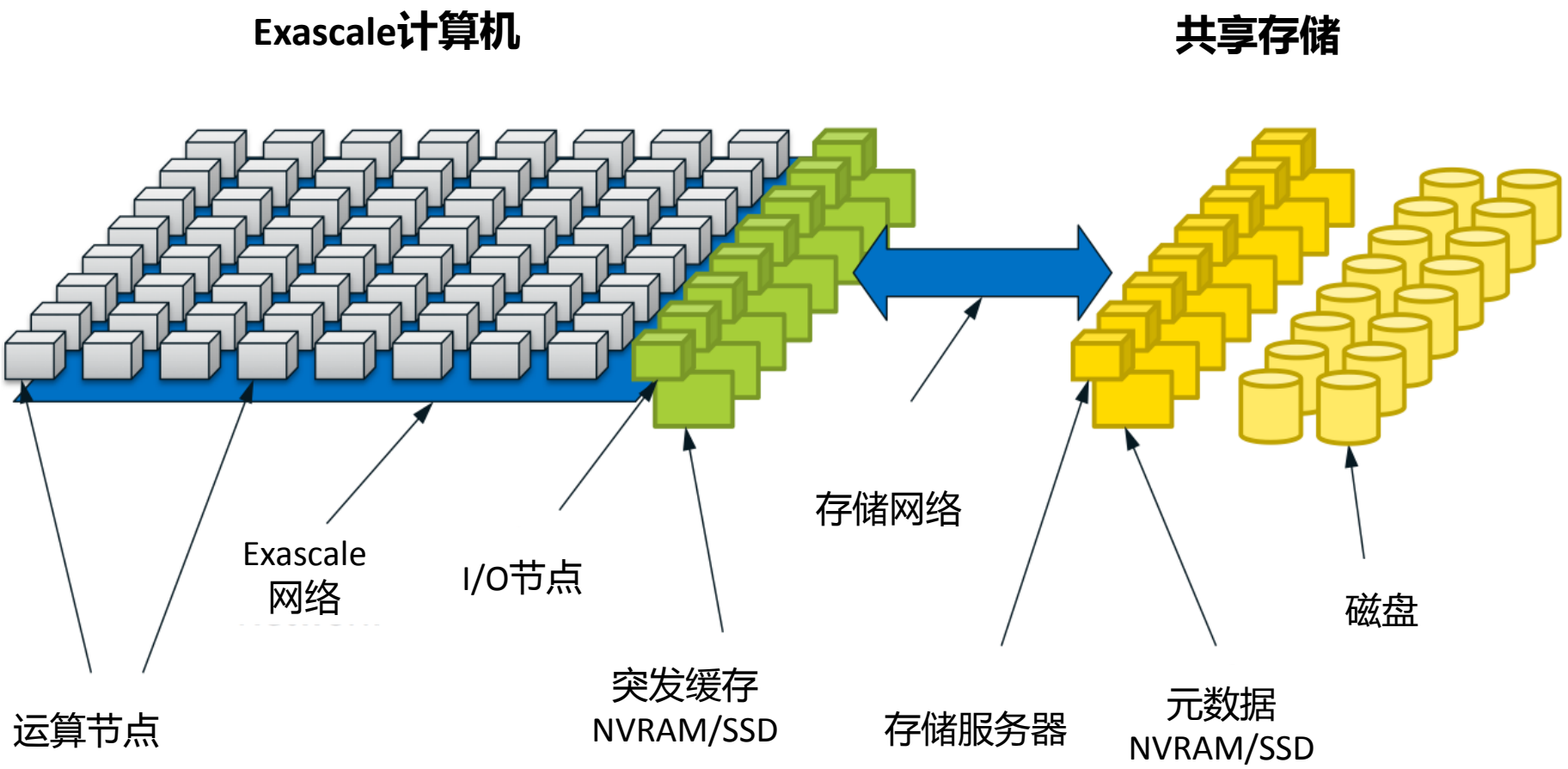
- 存储将成为科研人员的工具
  - 易于管理，全面清晰的交互
  - 能根据需求将运算移至存储，或者将存储移至运算



- 克服现今文件系统可扩展性的限制
  - 数个petabyte的数据集
  - 共享文件以及元数据性能
  - 系统的水平扩展性 & 偏差或抖动(jitter)

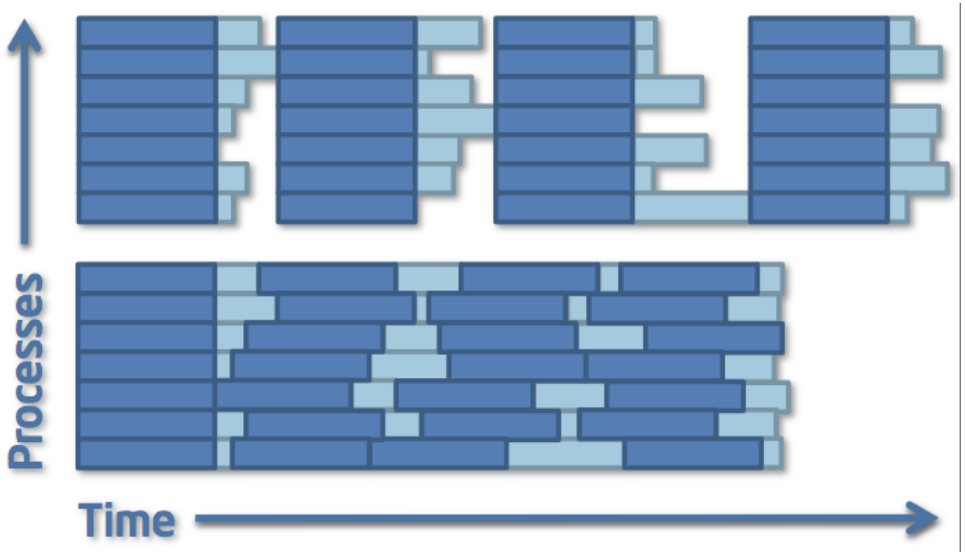
- 提供前所未有的容错性
  - 以失效将时常发生而不是偶然发生作为目标设计的系统
  - 保证数据以及元数据的一致性和完整性

Exascale I/O 的系统架构



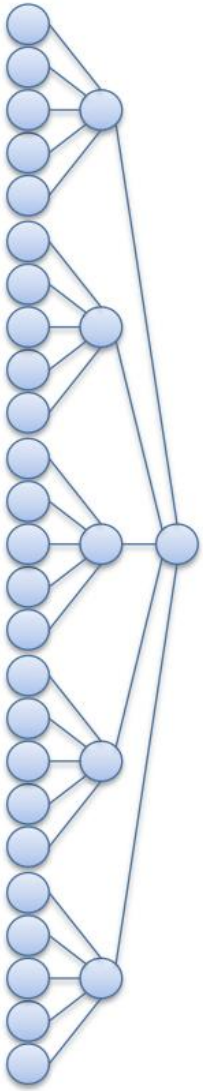
非阻塞式APIs

- 偏差或抖动(Jitter)
  - 进程调度偏差
  - 电源管理
  - 非平衡负载（中断，调度）
- 紧耦合
  - 同步数据传输简化了应用的开发
  - 使得应用更难扩展
- 松耦合
  - 避免了系统的闲置间隙
  - 需要非阻塞的I/O处理机制
- 所有的I/O都是非阻塞的(non-blocking)
  - 初始化请求(initialize)
  - 结束事件(completion event)



协同操作(Collective) 避免N:N的通信

- 服务器之间
  - 协同排除有故障的客户端
  - Open, commit...
- 客户端之间
  - 通信转移至栈的上层
  - 协同操作请求可以附加在上层其他通信内容之中
  - 上层通信机制更加高效
- local2global / global2local
  - 一个进程可以代表进程组执行I/O调用
  - local2global生成不透明可共享的数据内容
  - global2local使用该数据内容生成本地访问句柄
  - 避免open storm

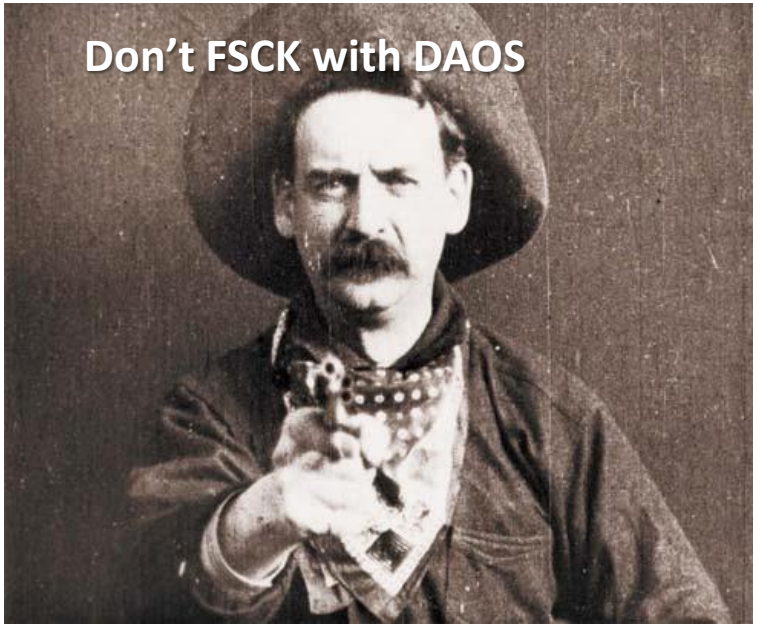


锁以及缓存 (Locking & Caching)

- 顺序化操作和强一致性cache是扩展性的瓶颈
  - 伪共享(false sharing)和locking storm
  - 存储系统应该避免参与客户端操作的顺序化
- 存储系统不是消息传递机制
  - 紧耦合的进程之间应该直接通信，而不是通过存储系统
- 底层I/O系统尽量避免预测上层缓存的需求
  - 聚合写(Aggregation device)
  - 预读(Readahead)
- 写操作应尽可能的避免阻塞读操作
  - 反之，读也应该避免阻塞写

原子性 (Atomicity)和可恢复性

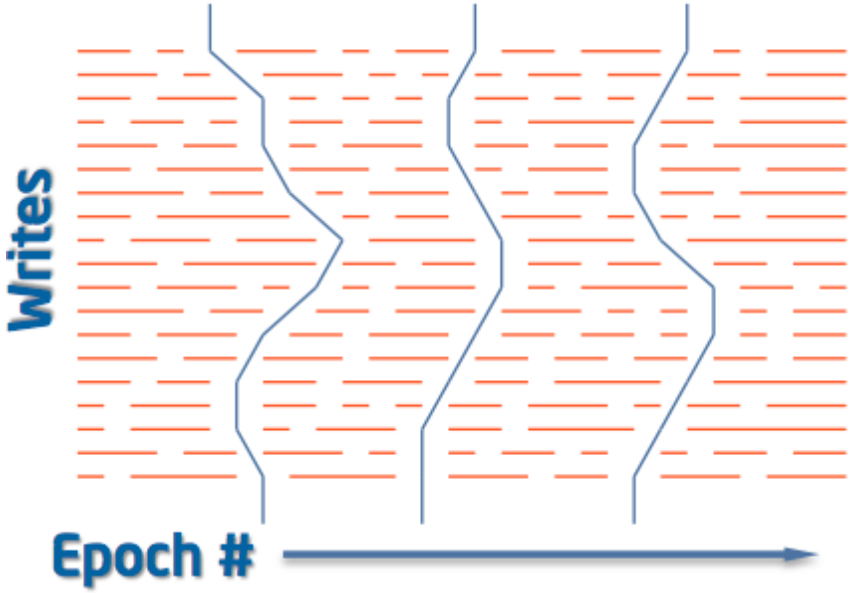
- 完整性和一致性的保证
  - I/O栈所有层的基本需求
  - 数据和元数据
    - 某层的元数据对于下层是数据
  - 避免 O(n) 恢复机制



- 事务机制
  - 存储系统的变化总处于一致性状态中
  - 恢复 == 回卷到最新永久状态
    - O(0) v.O(transaction size) 的恢复时间
  - 整个I/O栈支持可嵌套的事务机制
  - 异步的事务机制
  - 通过Scrub用来检查bitrot

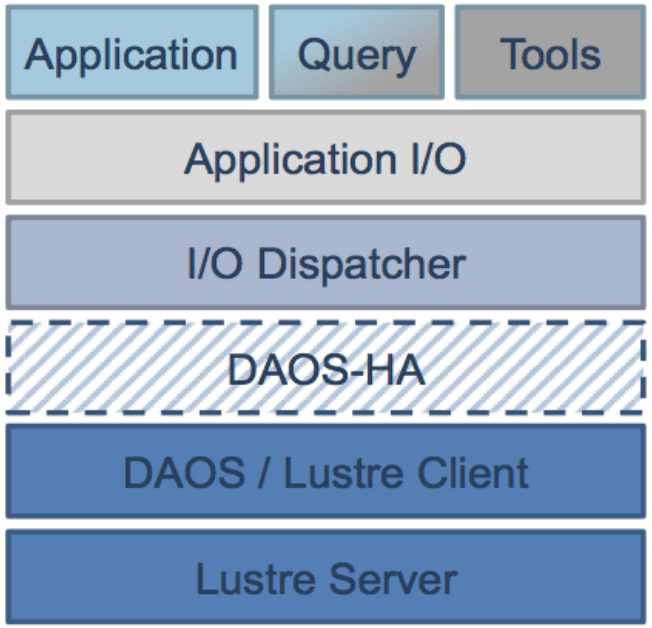
事务模型和Epoch

- 事务以epoch排序
  - 写操作按照epoch顺序展开
  - 对同一个epoch的所有写操作将被原子的提交
  - 对同一个epoch的所有读操作将得到一致的数据
- Epoch和事务的作用域(scope)
  - 以容器位单位(Container)
  - 多进程同时对多对象实施读/写操作
  - Epoch可以标识snapshot
- 无限制的事务pipeline
  - 系统可以聚合epochs
  - Query/Wait/Slip分别用来查询、等待以及回收一致的epoch

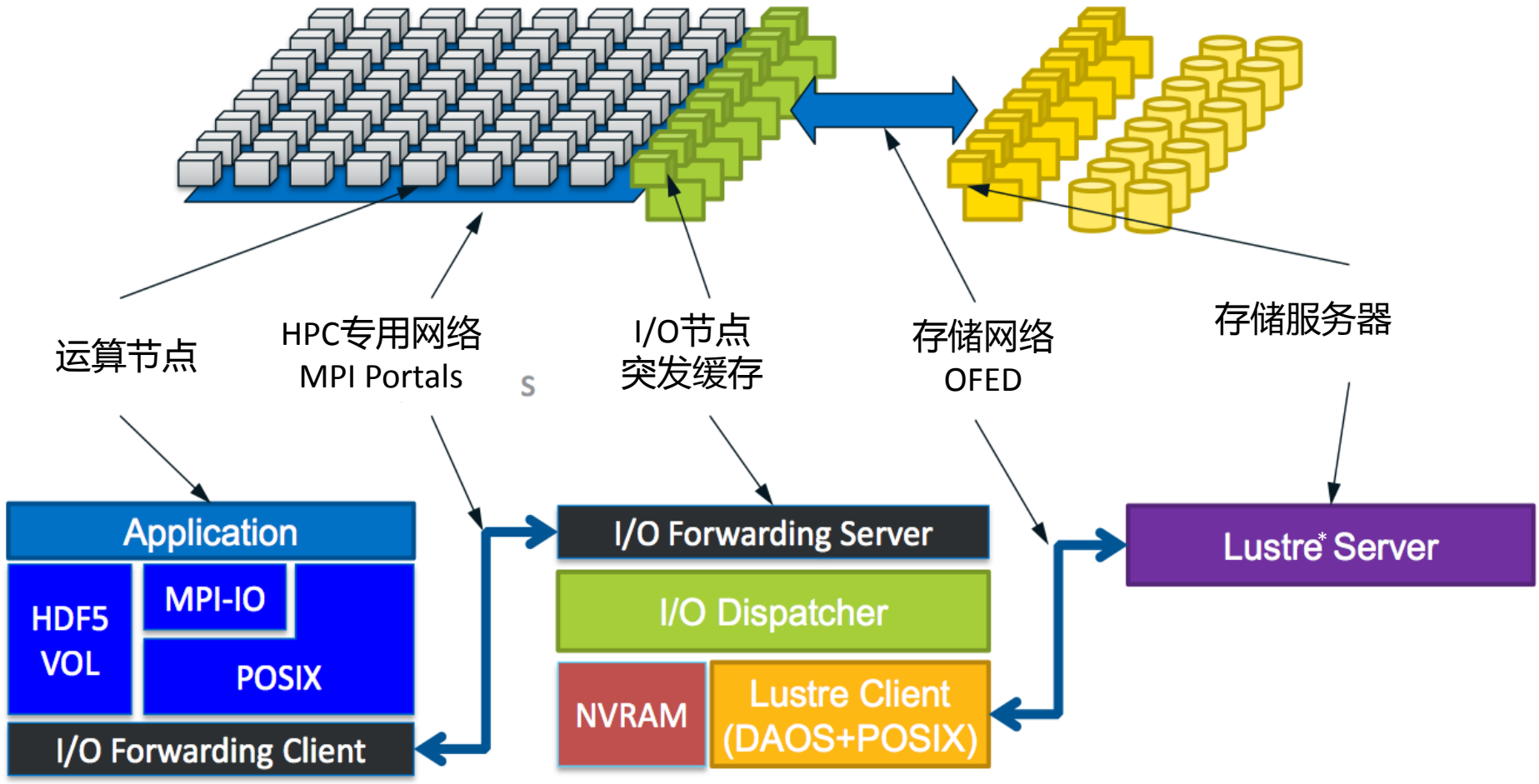


I/O 栈结构

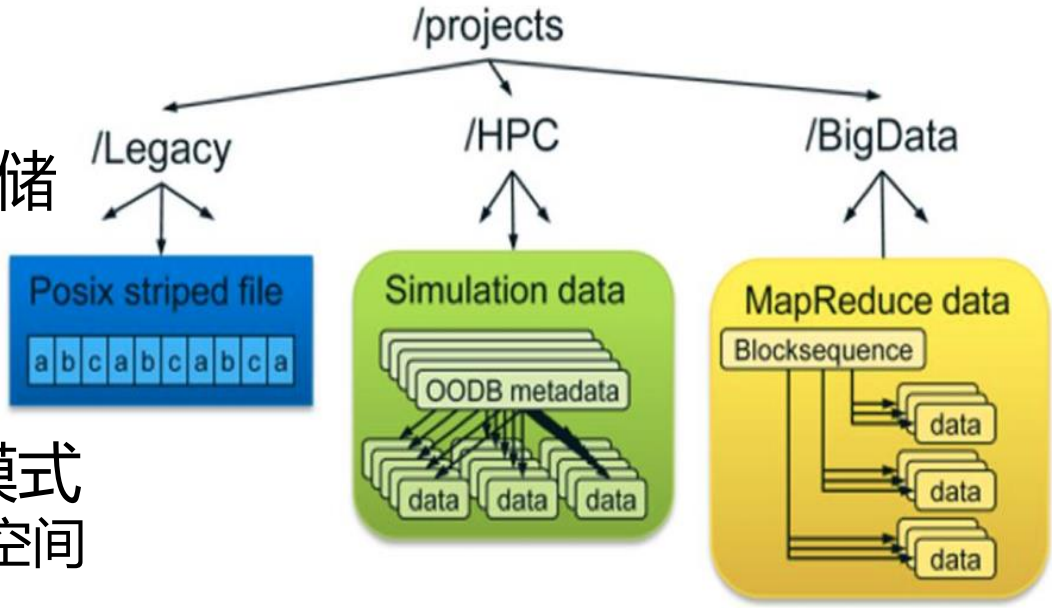
- 应用和工具
  - 查询、查找、分析、浏览数据
  - 分析传送
  - 分析和可视化In-transit数据
- 应用I/O API
  - 多样化的特定域API和对象模式
- I/O调度器(I/O dispatcher)
  - 理解存储结构
  - 管理突发缓存，缓冲峰值负载以匹配存储性能
- DAOS-HA
  - 高可靠、可扩展的对象化存储
  - Fast Forward的后续项目
- DAOS (Distributed Application Object Storage)
  - 可扩展的，事务化的全局共享的对象化存储
  - 避免POSIX



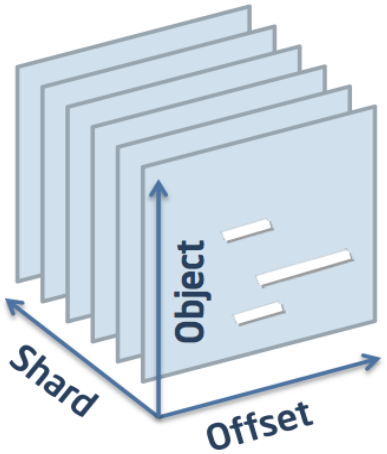
I/O系统架构和I/O栈结构



DAOS容器(Container)

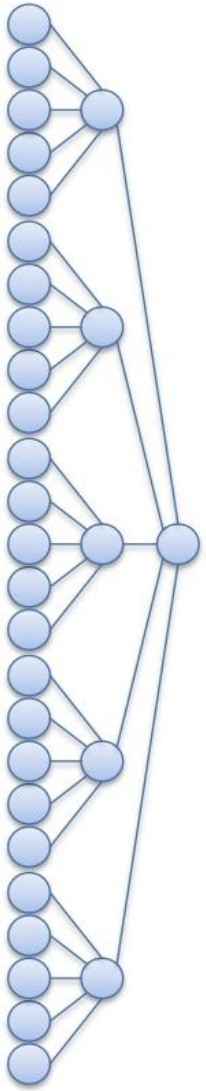


- 虚拟Lustre*底层的对象存储
  - 容器之间无共享
  - 容器内部可以有数十亿的对象，数千计的存储服务器
- 私有的对象名空间/对象模式
  - 避免污染文件系统的全局名空间
- 事务化的PGAS
  - Baseline : addr = <shard.object.offset>
  - HA : addr = <layout.object.offset>
- 读与写
  - 没有创建与删除
  - Punch == 高效的刷写 “0”



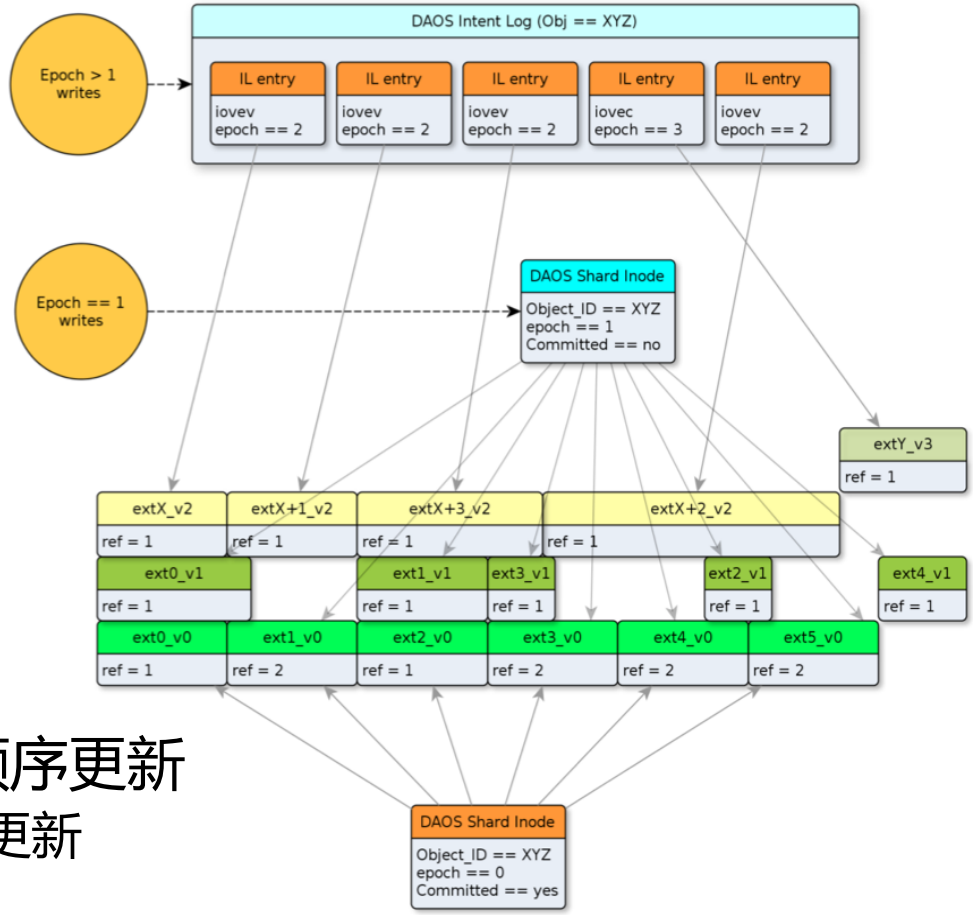
服务器的协同操作(Server Collectives)

- 容器水平扩展性的基础
- 协同监控客户端的健康程度
  - 避免”ping”风暴 (N^2)
  - Gossip协议
  - 累加型失效监测(Accrual failure detection)
- 协同响应某些I/O请求
  - 分布式的事务（全局epoch的提交）
- 生成树
  - 可扩展的O(log n)延迟
  - 建立和恢复



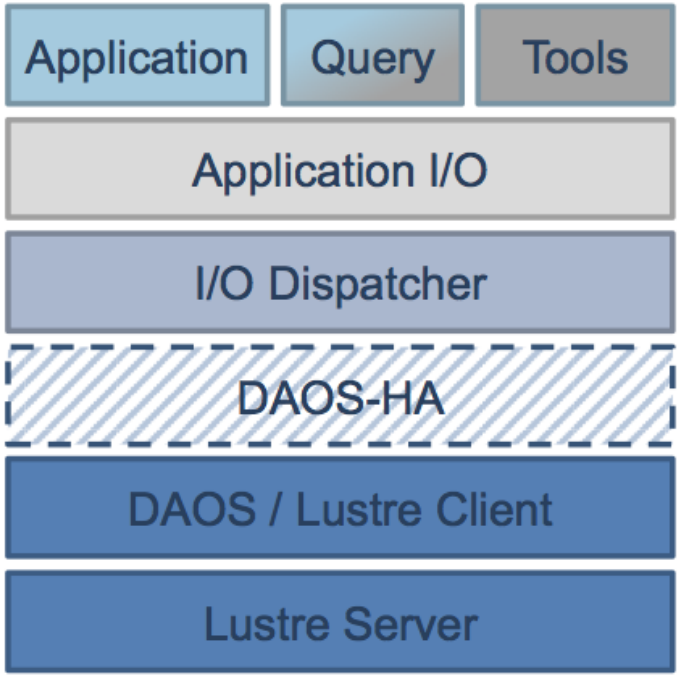
版本化的对象存储(Versioned object storage)

- Shard是VOSD中的对象
- COW & snapshot
- 版本化的意图日志
  (Version intent log)
  - 按照epoch顺序展开写数据
- 实时的写操作
  - 非顺序的实时写
  - 无需拷贝或者移动整块的数据
- Extent的metadata按照Epoch顺序更新
  - 先前的epoch完成之后立刻开始更新
  - 尽可能的实时更新
  - 如果不能实时更新，则根据intent log中的信息进行更新



I/O调度器 (I/O Dispatcher) & HDF5应用I/O

- I/O调度器 (IOD)
  - 突发缓存(NVRAM)和全局存储(DAOS)的抽象
  - 匹配峰值负载：I/O比率/延迟/带宽
  - 根据上层需求进行存储布局优化
  - 以事务为单位传输缓存中的数据
  - 提供给上层的恢复力模型
  - 与任务调度器整合
  - 端到端的数据完整性
- HPF5应用I/O
  - 为HPC构造的对象容器
  - 提供新的应用层功能
  - 扩展的HDF5数据模型
  - 新的存储格式



目前进度以及进一步研发

- 目前进度 (DAOS)
  - 非阻塞式API
  - 容器访问接口
  - 基本的I/O函数
  - 基于ZFS的VOSD
  - 客户端的collective函数
  - LNet层的collective函数
  - 基于epoch的事务机制(in progress…)
  - 源代码: git.hpdd.intel.com (ff/daos_lustre, ff/daos_tools, ff/daos_posix, …)
- 下一步的研发
  - 产品化和系统集成
  - 基于Btrfs的VOSD – in-kernel (GPL)
  - DAOS-HA
  - 多样化的上层APIs

Thanks

Question?