Lustre®元数据性能测试

Cheng Shao
cheng_shao@xyratex.com

xyratex

概要

- 关于Xyratex
- Lustre元数据性能测试回顾
- 元数据性能测试方法
- Lustre元数据性能问题分析

The Power of Xyratex

全球解决方案提供商

高品质数据存储硬件产品，软件产品及相关服务

拥有广泛的数据存储解决方案相关经验

存储媒体，存储硬件平台，集群文件系统

拥有370项存储相关专利

解决复杂的数据存储技术之相关需求

ClusterStor™ Scale-Out Storage Solutions

OneStor™ OEM Storage

Capital Equipment

xyratex

HDD Capital Equipment Solutions

为高产量硬盘制造商提供全自动硬盘测试解决方案，servo-track writing技术，以及磁盘处理方案

OEM Storage Solutions

高密度，高性能和高可用性数据存储盘柜，以及完全整合的应用平台

ClusterStor™ Storage Solutions

可横向扩展的数据存储解决方案，为高性能和大数据应用而设计，满足研发，政府和商业应用

HGST a Western Digital company
WD Western Digital
Seagate
TOSHIBA Leading Innovation
HOYA

DELL
hp
HITACHI Inspire the Next
IBM
Microsoft
NetApp
AVID

CRAY
DELL
hp
cea
AWE
NCSA
RUTGERS
U.S. AIR FORCE
MIT

业界领先的数据存储解决方案提供商

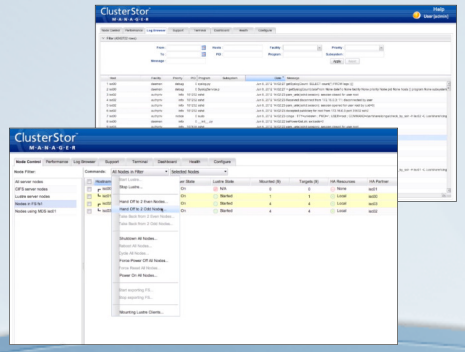
OEM Storage Solutions

- 2012年占有全球交付的外置存储总容量的15%*
- 2012年交付逾3000PB存储设备
- 全球最大的基于硬盘的存储设备提供商

HDD Capital Equipment Solutions

- Xyratex的技术被用来制造全球约50%的硬盘产品*
- 最大的硬盘制造设备独立供应商

ClusterStor™ Manager



ClusterStor™ HPC Storage Solutions

- 全世界最快的数据存储系统
- 2012年交付用户的ClusterStor产品总容量逾90PB
- 广泛支持全球范围科研、能源、国防和生命科学相关的应用

ClusterStor™

xyratex

*Source: Company estimates

关于Xyratex和Lustre®资产

- Xyratex在2013年2月从Oracle购得Lustre的相关资产
- Lustre相关资产包括
  - Lustre商标
  - lustre.org域名
  - 从1999年Lustre诞生至Lustre2.0版本的源代码知识产权
  - 与Lustre 1.8版本相关的sustaining项目
  - 开发和支持的工具链 (变异和测试工具)
  - 社区资源 (lustre.org, wiki, Bugzilla, Mailing lists, etc.)
- Active Lustre support contracts

元数据性能测试

- 过去业界和社区主要关注于Lustre的数据IO性能，对元数据的性能分析较少

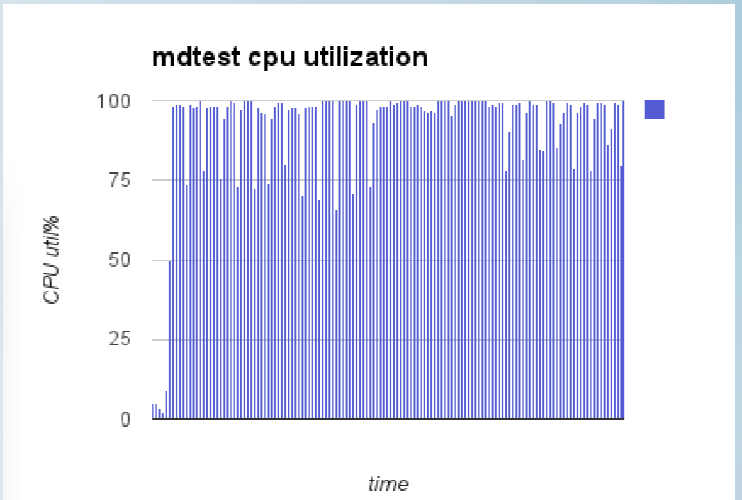
- 最近以来Lustre元数据的性能得到越来越多的重视
  - 一些高性能科学计算的应用需要生成大量临时文件
  - 新的应用模式的出现
    - 审计应用
    - 容灾备份
    - 文件共享服务
  - 面对来自其他分布式／并行式文件系统或数据存储解决方案的竞争

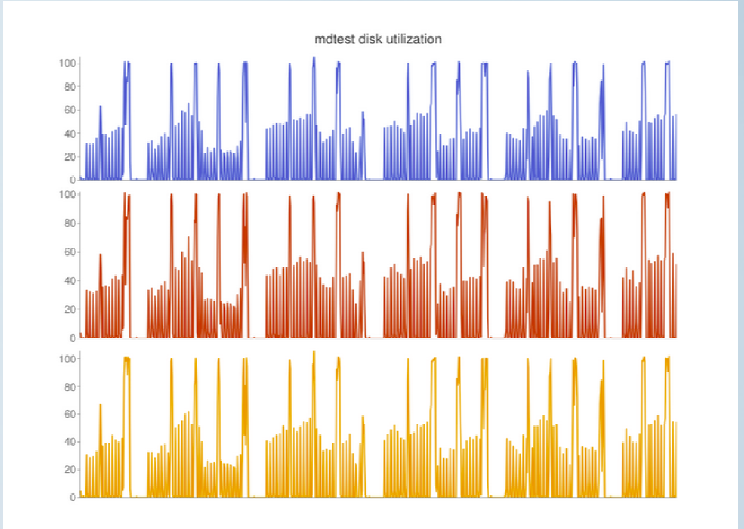
元数据性能测试方法

- 元数据测试工具
  - mdtest
  - mds-survey
  - mdsrate
  - metabench
  - bonnie++
  - SPECsfs
  - 其他一些自制的工具

元数据性能测试方法（续）

- MDS硬件配置
  - –后端存储
  - –ldiskfs日志
  - –CPU和内存
  - –网络 (LST)





元数据性能测试方法（续）

- 与mdtest相关的选项
  - 任务进程数量
  - 文件总数
  - 运行次数
  - shared mount 与 unique mount
  - ‘-a’和‘-o’选项用来避开OST的参与
  - 目录结构

元数据性能测试方法（续）

- 测试设计
  - 增加每个客户节点上任务进程数量
  - 列出影响性能的各种因素，每次测试一种因素
  - 测试进行中进行profiling
  - 对测试结果的差异给出合理解释
  - 模拟测试使用过一阵的文件系统

元数据性能问题诊断

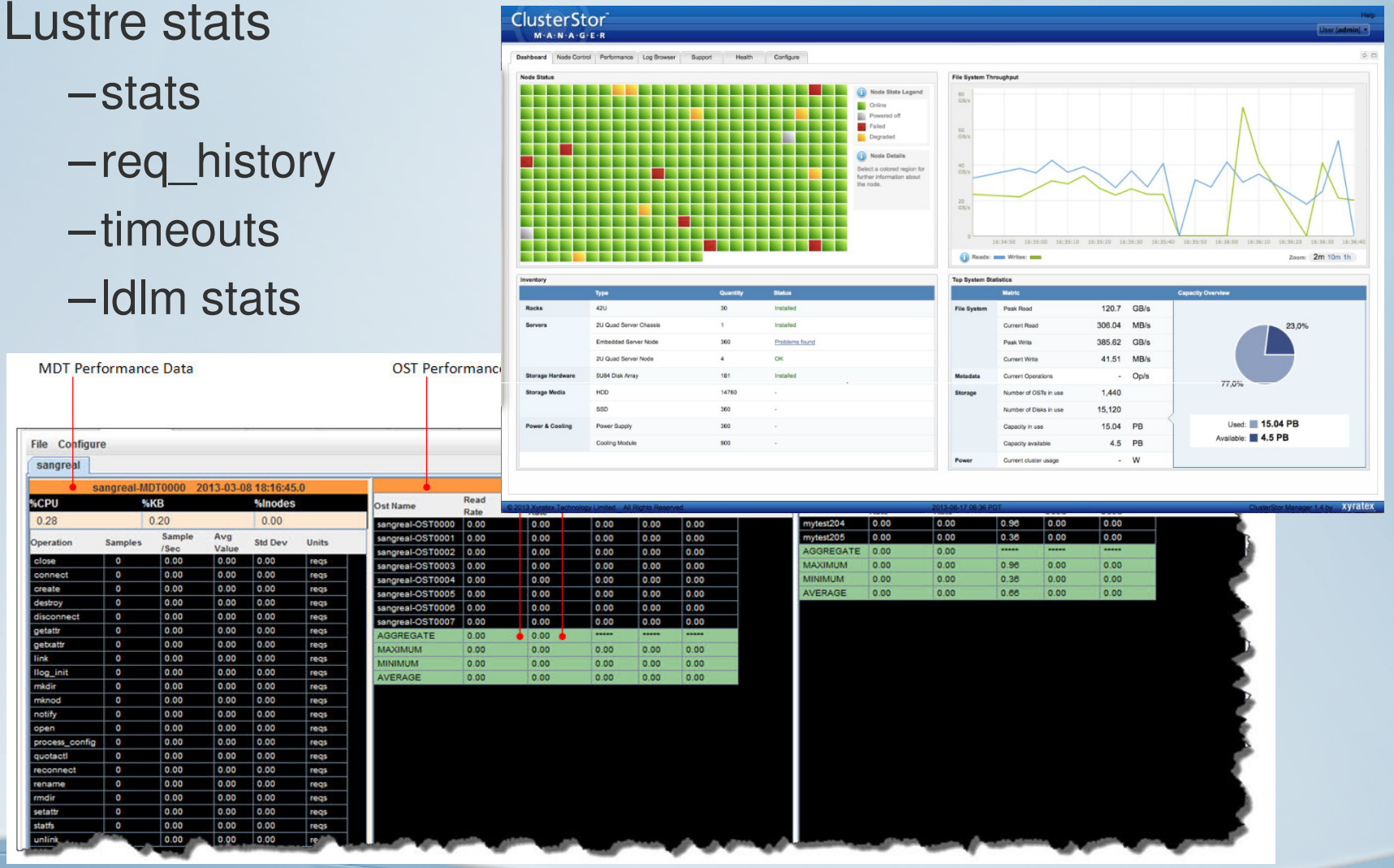
- 性能问题分类
  - 性能下降
  - 没达到预期指标
  - 用户反映
- 诊断方法
  - profiling
  - Lustre stats
  - crash dump

元数据性能问题诊断 （续）

- Profiling工具
  - 负载
    - iostat
    - perf / oprofile
  - 延迟
    - latencytop
    - systemtap

元数据性能问题诊断 （续）

- Lustre stats
  - –stats
  - –req_history
  - –timeouts
  - –ldlm stats



元数据性能问题诊断 （续）

- 实例分析
  - 同样的server 版本， 2.3的客户端比1.8.8的客户端元数据性能下降
  - 性能测试工具: 各任务进程在同一个目录下创建文件，然后删除。没有子目录。
  - 文件创建速度，2.3客户端和1.8.8客户端之间没有差别
  - 文件删除速度，2.3客户端仅是1.8.8客户端的三分之一

元数据性能问题诊断 （续）

- 诊断过程
  - 成功复制性能问题
  - 使用systemtap来测量unlink调用路径的延迟
  - 使用的脚本

```
global times[100]

probe module("lustre").function("ll_unlink").return {
        times["ll_unlink"] <<< (gettimeofday_us()-@entry(gettimeofday_us()))
}

probe module("mdc").function("mdc_unlink").return {
        times["mdc_unlink"] <<< (gettimeofday_us()-@entry(gettimeofday_us()))
}

probe module("mdc").function("mdc_reint").return {
        times["mdc_reint"] <<< (gettimeofday_us()-@entry(gettimeofday_us()))
}

probe end {
        foreach (func in times) {
                printf("\n\n==== %s stat ====\n", func)
                printf("total samples: %d, min %d, max %d, avg %d\n",
                        @count(times[func]),
                        @min(times[func]),
                        @max(times[func]),
                        @avg(times[func]))
                print(@hist_linear(times[func], 0, 45000, 400))
        }

        printf("\nbye now\n")
}
```

元数据性能问题诊断 （续）

- 诊断过程
  - –对1.8和2.3客户端的延迟测量结果进行比较
  - –脚本输出例子 (ll_unlink的延迟)

1.8 客户端

```
total samples: 8000, min 211, max 144987, avg 506
 value |------------------------------------------------- count
     0 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  7989
  2000 |                                                     6
  4000 |                                                     0
  6000 |                                                     0
       ~
 40000 |                                                     0
 42000 |                                                     0
 44000 |                                                     2
 46000 |                                                     0
 48000 |                                                     0
       ~
 54000 |                                                     0
```

2.3 客户端

```
total samples: 8000, min 249, max 145504, avg 1539
 value |------------------------------------------------- count
     0 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  7391
  2000 |                                                    30
  4000 |                                                    47
  6000 |                                                    41
  8000 |                                                    47
 10000 |                                                    75
 12000 |                                                    37
 14000 |                                                    83
 16000 |                                                   104
 18000 |                                                    69
 20000 |                                                    25
 22000 |                                                    19
```

  - –延迟的绝大部分来自客户端等待MDS的请求应答

元数据性能问题诊断 （续）

- 诊断过程
  - MDS端调用路径

```
mdt_reint_unlink
        mdt_object_find_lock // lock parent dir
        mdd_unlink           // actual unlink
```

  - profiling结果

函数	1.8客户端	2.3客户端
mdt_reint_unlink	304	1328
mdd_unlink	75	76
mdt_object_find_lock	51	1110

  - 为什么MDS在处理2.3客户端的unlink请求时需要更长的时间获取父目录的锁？

元数据性能问题诊断 （续）

- 诊断过程
  - 查看性能测试工具源代码
    - 创建文件的工作流
    - 删除文件的工作流
      - 调用readdir获取下一个目录项
      - 调用stat以保证只删除一般文件
      - 删除目录项和对应的文件

- Lustre 1.8和2.3的区别
  - readdir chunk size从1个页面增加到256个页面
  - 锁冲突的机会增加

元数据性能问题诊断 （续）

- 结论
  - 某些性能测试工具的工作流程无法达到Lustre所能提供的最优的元数据性能，有进一步调优的空间。用户应用程序也会有同样的问题存在。
  - 用systemtap作为profiling工具可以有效地进行性能问题的诊断

谢谢

cheng_shao@xyratex.com