Linux 在互联网行业的优化

马涛 & 李勇

boyu.mt@taobao.com

bosong.ly@alipay.com

内容

- 内核组介绍
- 淘宝内核的新特性
- 淘宝内核的优化

淘宝内核组介绍

成立于 2010 年 6 月

目前 8 名成员，主要工作领域在 cpu 调度，内存管理，文件系统， IO 系统， cgroup, perf 等领域。

http://kernel.taobao.org/index.php/Documents/kernel_team_members

淘宝内核组介绍

内核社区影响力

No.108	Hauppauge	185(0.06%)
No.109	EXAR	184(0.06%)
No.110	Voltaire	180(0.06%)
No.110	Calxeda	180(0.06%)
No.112	SANPeople	178(0.06%)
No.113	Toshiba	174(0.06%)
No.114	Tuxera	173(0.06%)
No.115	igalia	171(0.06%)
No.116	Bitmer	170(0.06%)
No.116	RisingTide Systems	170(0.06%)
No.118	Inktank	164(0.05%)
No.119	MathEmbedded Consulting	163(0.05%)
No.120	Tao Bao	161(0.05%)
No.121	Real-Time Remedies	156(0.05%)
No.122	CSR	151(0.05%)
No.122	Collabora Multimedia	151(0.05%)
No.124	Open Nandra	150(0.05%)
No.124	Fusion-io	150(0.05%)
No.126	Philosys Software	145(0.05%)
No.127	STRATO	144(0.05%)
No.128	secunet Security Networks AG	141(0.05%)
No.129	Barco	139(0.05%)
No.130	US National Security Agency	138(0.04%)
No.130	OMICRON electronics	138(0.04%)
No.186	Tungsten Graphics	8019(0.02%)
No.187	Apple	7940(0.02%)
No.188	LG Electonics	7871(0.02%)
No.189	emlix Gmbh	7805(0.02%)
No.190	Wacom	7756(0.02%)
No.191	Boundary Devices	7728(0.02%)
No.192	Ozmo	7665(0.02%)
No.193	Bluewater Systems	7561(0.02%)
No.194	Bluecherry	7436(0.02%)
No.195	Digigram SA	7430(0.02%)
No.196	Stream Processors	7400(0.02%)
No.197	ACM	7386(0.02%)
No.198	Semihalf Embedded Systems	7380(0.02%)
No.199	Tao Bao	7218(0.02%)
No.200	secunet Security Networks AG	7161(0.02%)
No.201	Collabora Multimedia	7090(0.02%)
No.202	Microgate	6992(0.02%)
No.203	Avionic Design Development GmbH	6859(0.02%)
No.204	GNU	6808(0.02%)
No.205	Vivecode	6714(0.02%)
No.206	EXOSEC	6596(0.02%)
No.207	Free Electrons	6549(0.02%)
No.208	Real-Time Remedies	6477(0.02%)
No.208	Digi International	6477(0.02%)
No.210	Telargo	6372(0.02%)

淘宝 2.6.32 内核介绍

RHEL6 操作系统 2.6.32 内核重要新特性

- Control Group 资源隔离
- 名字空间隔离
- RPS/RFS
- KVM 虚拟化技术
- 增强的硬件虚拟化技术
- 大内存页面
- 虚拟机间内存页面共享
- 更高精度的时钟源和计时器
- 更全面的 Ext4 文件系统
- XFS 商业支持

淘宝 2.6.32 内核介绍

淘宝 2.6.32 内核的增强特性

- Ext4 文件系统
  - big allocate
  - fallocate stale data
  - inline data
- Overlay 文件系统
- 内存管理
  - 性能退化的改进
  - 预读性能提升
  - 回写性能改进
- 网络
  - 驱动、协议栈缺陷修正
  - Netoops
  - 动态调整 TCP time_wait
- 可维护性
  - Netoops
  - Ext4 I/O Error Guard
- 性能分析
  - Page Cache 命中率
  - 文件系统 I/O 数据类型统计
  - 文件系统磁盘数据类型分析
  - 网络端到端响应时间统计
- I/O 栈优化
  - 带 Deadline 的 CFQ 调度器
  - 支持 page cache 的异步 I/O
  - flashcache 优化

淘宝 2.6.32 内核介绍

性能提升

- Ext4 文件系统
  - I/O 吞吐量增加一倍， IOPS 降低一半
  - 磁盘可用空间增加 1%，元数据内存占用量明显降低
  - 文件系统 fsck 时间缩短为 1/4
  - 大文件预分配时间缩短 2 个数量级

- Control Group 资源隔离
  - 简单替换 xen 虚拟机，应用响应时间降低 20%
  - 显著降低多个虚拟机占用的内存数量
  - 成倍提升对 SSD 的 I/O 访问性能

- KVM 及增强的虚拟化技术
  - 基于 sheepdog 虚拟化系统将内核测试的时间从 20 天缩短到 2 天

淘宝 2.6.32 内核介绍

性能提升（Cont.）

- 内存管理和 I/O 调度器改进
  - 明显降低 I/O 120 秒超时发生几率
  - 更有效的预读
  - 对大数据作业支持更大的 I/O 批量

- 可调的 TCP time_wait 参数
  - 可以将 60 秒的 fin 超时时间动态调整为最短 1 秒
  - 明显降低系统整体负载和内存占用量

新功能与新特性

Ext4 文件系统

- Ext4 Bigalloc
  将磁盘分配单位扩展为大于 4KB 的多个连续的磁盘块（称为 cluster ）
  - 16KB, 32KB, 64KB 是比较常用的 cluster 大小
  - 每个块组的分配单元数量有限，分配单元变大 --> 单个块组变大
  - 增加可用磁盘空间，降低元数据内存占用
  - 降低元数据数量，进而降低 mkfs 、 fsck 时间 [非常重要 *]
  - 如果存在小文件或者小目录，配合 Ext4 inline-data 使用可能更佳

- No-jounral Ext4
  尽可能避免日志 I/O 的额外开销，有利于提高在 SSD 上或者元数据密集型 I/O 场景的性能
  •配合 noatime,nodiratime ， 完全没有额外的元数据日志 I/O

新功能与新特性

Overlay 文件系统

VFS（虚拟文件系统）层面的文件系统，可以在多个挂接点之间 Copy-on-Write 的共享磁盘文件在内存中的 Page Cache 。

与 "mount -o bind" 不同之处在于，所有挂接点都是可写的。当某一个挂接点要修改共享的文件时，会分配一个该挂接点使用的私有 inode 。

例如多个程序都有自己的私有根文件系统运行，但是这些根文件系统中绝大多数内容都是相同的，只有程序自己的日志和个别配置不同。那就可以将各个实例都叠加挂接（overlay）到一个基础文件系统上。

- 所有没有被修改的内容都会在内存中共享相同的 Page Cache
- 个别被修改的内容会在内存中分配不同的 Page Cache
- 被修改的内容都会非逸失的存储在各自挂接点所在的磁盘文件系统位置
- 在多实例同时运行的场景中，大大降低不必要的 Page Cache 占用量

新功能与新特性

内存管理

- 性能退化的改进
  - 缺省关闭 context_readahead
  - 缺省关闭 mlock_flush_pagevec
  - 缺省关闭透明大页（THP）和物理内存碎片整理（compaction）

- 预读性能的提升
  - 避免 pagevec 特性带来的预读碎片，增加连续预读的几率

- 回写性能改进
  - 增加以 PROT_EXEC 进行 mmap 的页面在内存的停留时间
  - 如果持续有页面置脏，则增加每次回写数量，并延长回写间隔

新功能与新特性

可维护性

- Netoops
  - 将内核故障信息通过网络发送到远程日志服务器
  - 只有内核 panic 、 OOM 和显式测试信息才会发送到远程日志服务器
  - 比 netconsole 具有更高的稳定性，更小的网络流量
  - 只要数据链路层以上协议栈代码可以正常工作，就可发送故障信息
  - 有利于快速确定内核故障，并对线上内核故障进行统计和分类

- Ext4 I/O Error Guard
  - 在 no-journal Ext4 文件系统中，加大对硬盘故障的容忍度
    - 设置采样周期（缺省 5 秒）
    - 设置捕捉到的 I/O 故障次数（触发阈值为 20）
  - 满足如下条件之一，方才将磁盘上的文件系统设置为只读模式，
    - 当在采样周期内捕捉到达到阈值的 I/O 故障次数
    - 在连续若干个采样周期内都有 I/O 错误，且错误总数达到阈值

新功能与新特性

性能分析

- Page Cache 命中率
  - 定量的评估预读的效果和 Page Cache 在特定工作负载下的效率

Devices:	read-ahead	read-miss	read-hit	write-miss	write-hit
sdc:	6794	594	12728	5493	1792
sdc1:	6794	594	12728	5493	1792
sdd:	8910	726	17083	3337	8
sdd1:	8910	726	17083	3337	8

  - 只有在打开分析功能时，才有 2% 的额外 CPU 消耗
  - 平常不打开分析功能时，几乎没有额外开销

新功能与新特性

性能分析（Cont.）

- 文件系统 I/O 数据类型统计
  - 接口文件 /sys/fs/ext4/<device>/io_stats

```
VERSION: 1.0
TYPE                          READ           WRITE
super block                      0               0
group descriptor                 0               1
inode bitmap                     0              10
block bitmap                     2               6
inode table                     37             177
extent block                     0               0
indirect block                   0               0
dir entry                      336            1006
extended attribute               0               0
regular data                 32769           99304
```

  - 可在运行时动态显示磁盘 I/O 分别对应的是哪类数据
  - 进而分析应用的 I/O 瓶颈或者热点

新功能与新特性

性能分析（Cont.）

- 文件系统磁盘数据类型分析
  - 集成在淘宝维护的 e2fsprogs 工具包中
  - 分析文件系统中不同类型数据所占的磁盘空间（以块为单位）

```
dumpe4fs 1.42 (29-Nov-2011)
[Blocks Usage (Unit: blocks)]
Super block:                6
Group descriptor:           6
Reserved GDT:               763
Inode table:                8192
Inode bitmap:               16
Block bitmap:               16
Link block:                 0
Journal:                    8192
Directory:                  1009
Extent:                     0
Uninit Extent:              0
Regular File Data:          66536
ACL block:                  0
```

- 运行 dumpe4fs -s <dev>
- 显示不同类型数据和元数据总数
- 定量分析文件系统磁盘利用效率

新功能与新特性

I/O 栈优化

- 带 Deadline 的 CFQ 调度器
  - 在兼顾高吞吐量的同时，避免某些低优先级 I/O 被饥饿
  - 在读写混合的工作负载中，明显改善响应时间

- 支持 Page Cache 的异步 I/O
  - 弥补当前 Linux 异步 I/O 必须使用 DirectIO 机制的限制
  - 如果不想应用程序在用户态维护 Cache，则可以用该机制
  - 在依然实现异步 I/O 的同时，可以使用内核 Page Cache 来缓存热数据

- flashcache 的优化
  - 支持按照比例控制

新功能与新特性

cgroup 优化

- 实现 instance aware 的性能监控
  - cpu 使用时间监控，让 top 等正常工作
  - context switch 的 instance 监控
- 实现 memory 的弹性分配
  - http://adc.taobao.com/ppts/up-1342080820-0.pdf
- 优化 io throttle 以及 io controller
  - 提供 io throttle 的队列长度监控
  - 支持按照比例控制

性能优化实例 -- 文件系统

Ext4 Bigalloc

- 格式化

  mkfs.ext4 -m 0 -O bigalloc, ^resize_inode,^uninit_bg,extent,meta_bg,flex_bg <dev>

  不加 -C 参数指定簇大小时，缺省为 64KB

- /etc/fstab

  不需要额外特殊挂载选项

线下压力测试

对比数据点	Normal Ext4	Ext4 Bigalloc	结论
write 创建新文件（cache write）	7772ms	1658ms	**时间缩短 78.7%**
write 创建新文件（direct io）	15617ms	15678ms	没有变化
fallocate 创建新文件	4198ms	1704ms	**时间缩短 59.4%**

Hadoop 测试

对比数据点	Normal Ext4	Ext4 Bigalloc	结论
单机格式化后的可用磁盘空间	22180G	22312G	**单机增加可用空间 132G**
单机 fsck 的时间	1916s	453s	**时间缩短 70.3%**

性能优化实例 -- 文件系统

OverlayFS

多个挂接点可以叠加的挂接在一个基准内容之上，叠加点内可以 Copy-on-Write 的写入内容。
为 VFS 层实现，不需更改磁盘格式或进行文件系统格式化。

挂接方法，

```
mount -t overlayfs none -o lowerdir=/base,upperdir=/snap1 /snap1
```

上述挂接方法，将 /snap1 目录叠加在 /base 目录上，并通过挂接点 /snap1 访问叠加后的空间。

如果 uppperdir 是另外一个目录，那进入 /snap1 看到的将是 lowerdir 和 upperdir 两个目录叠加后的结果。

性能优化实例 -- 文件系统

OverlayFS (Cont.)

如果不用 OverlayFS，要可写的访问 32 个文件系统，需要创建 32 个副本。假定每个副本中有 1GB 的文件内容需要被访问，则在 24G 物理内存的服务器上，

- 不使用 OverlayFS，直接访问 32 个文件系统副本： Page Cache 占用为 21181 MB
- 使用 OverlayFS，访问 32 个文件系统的挂接点： Page Cache 占用为 1037 MB

结论，

- OverlayFS 可以使得 Page Cache 占用量即便副本数增加，也不会明显增加。
- 副本数量越多，使用 OverlayFS 节省的 Page Cache 越多。

性能优化实例 -- 文件系统

Buffered AIO

合并在 2.6.32-220 系列内核中，使用方法如下，

- 传递给 io_submit() 的 struct iocb.aio_lio_opcode 中指定常数 9
  - 我们目前还没有维护 libaio，因此没有为常数 9 指定符号名称
  - 以后如果维护 libaio 后，会为 9 起一个名字，保持 ABI 二进制兼容
- 打开文件时，不再使用 O_DIRECT 标志

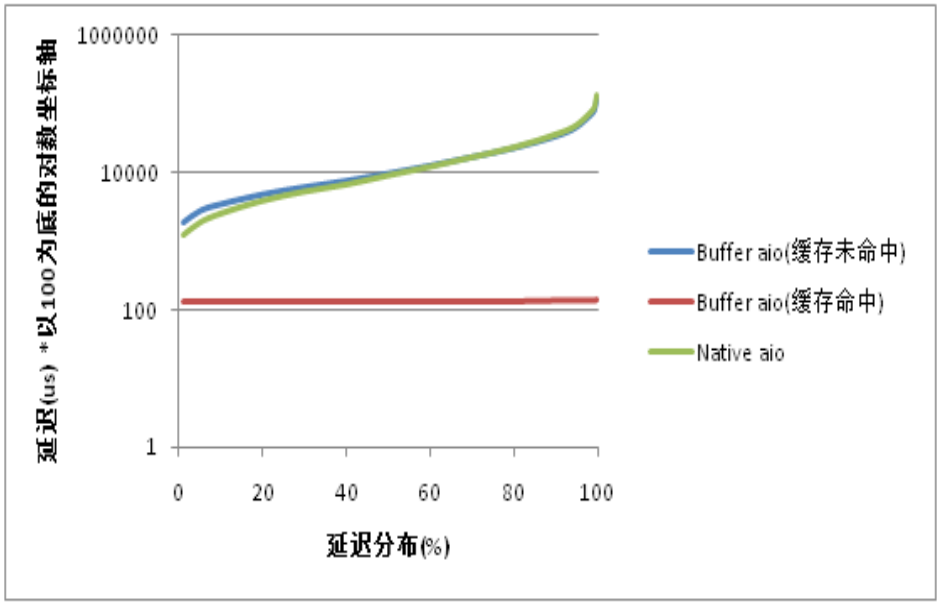
“优点”，

- 此外不需要额外修改现存程序的运行逻辑
- 应用程序为异步 I/O 申请内存时不需要额外的对齐要求

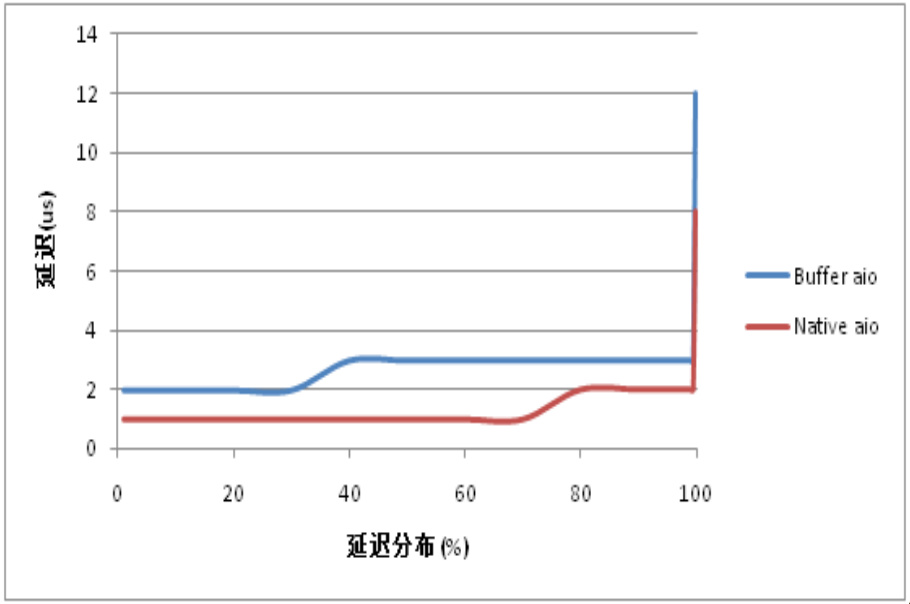
性能优化实例 -- 文件系统

Buffered AIO (Cont.)

磁盘 I/O 延迟对比



RAMdisk I/O 延迟对比



结论：额外的 Page Cache 复制开销很小，为 1 微秒左右
在当前磁盘和 SSD 设备上基本可以忽略。

性能优化实例 -- 文件系统

Buffered AIO (Cont.)

与多线程同步 I/O 对比，

	线程数	IOPS	平均延迟
Buffered AIO	1	1036	15425 us
多线程同步 IO	8	829	9635 us
多线程同步 IO	16	1049	15230 us

达到相同的 I/O 负载， buffered AIO 需要的线程数量大大降低，进程调度开销和 loadavg 统计明显改善。

性能优化实例 -- 内存管理

页面回收

- RHEL6 自带的 2.6.32 内核对页面的回写是非常激进的。
  - （优点）及时的将脏页写往硬盘，避免数据丢失
  - （缺点）在淘宝的部分应用中，需要 mmap 一个大文件，然后随机的更新文件内容，由于 writeback 的激进，造成很大的 IO 写压力，以致应用整体被拖慢

- 针对这一性能退化，我们在 kernel writeback 模块的 maintainer FengGuang Wu 的帮助下，去掉了 writeback 的回绕机制，增加了对 inode “弄脏”时间的更新，合理降低了 writeback 的写频度

- writeback 旧的回绕机制： writeback 在把一个文件的脏页按顺序从头到尾都写回硬盘以后，如果发现文件又有新的脏页出现，则回到文件头，重新开始写。（对于随机写，“新的脏页”是任何时候都有的，所以造成 writeback 高强度的循环写页）

性能优化实例 -- 内存管理

页面回收（ Cont. ）

- 测试用例： mmap 一个 256MB 的文件，然后在 256MB 的范围内随机的写，一次写 8 个字节，一共写 25 亿次，统计总运行时间
  - 在 RHEL 自带的 2.6.32 的内核上运行时间： 805 秒
  - 在 Taobao-kernel 上运行时间： 390 秒

我们的工作

kernel.taobao.org

GIT 源码树 URL：
http://kernel.taobao.org/git/?p=taobao-kernel.git;a=summa

Q&A

谢谢！