软硬结合统一单机存储引擎

张振
百度基础架构部高级工程师
2020年/11月

目 录

CONTENT

01 统一单机存储引擎背景

01统一单机存储引擎的背景

■存储软件的性能

- 随着NVMe、Optane等存储硬件性能的发展，软件上的延迟在整个存储系统中所占的比例越来越大
- 应用程序需要控制I/O路径以获得性能的稳定性和确定性

■存储介质的多样性

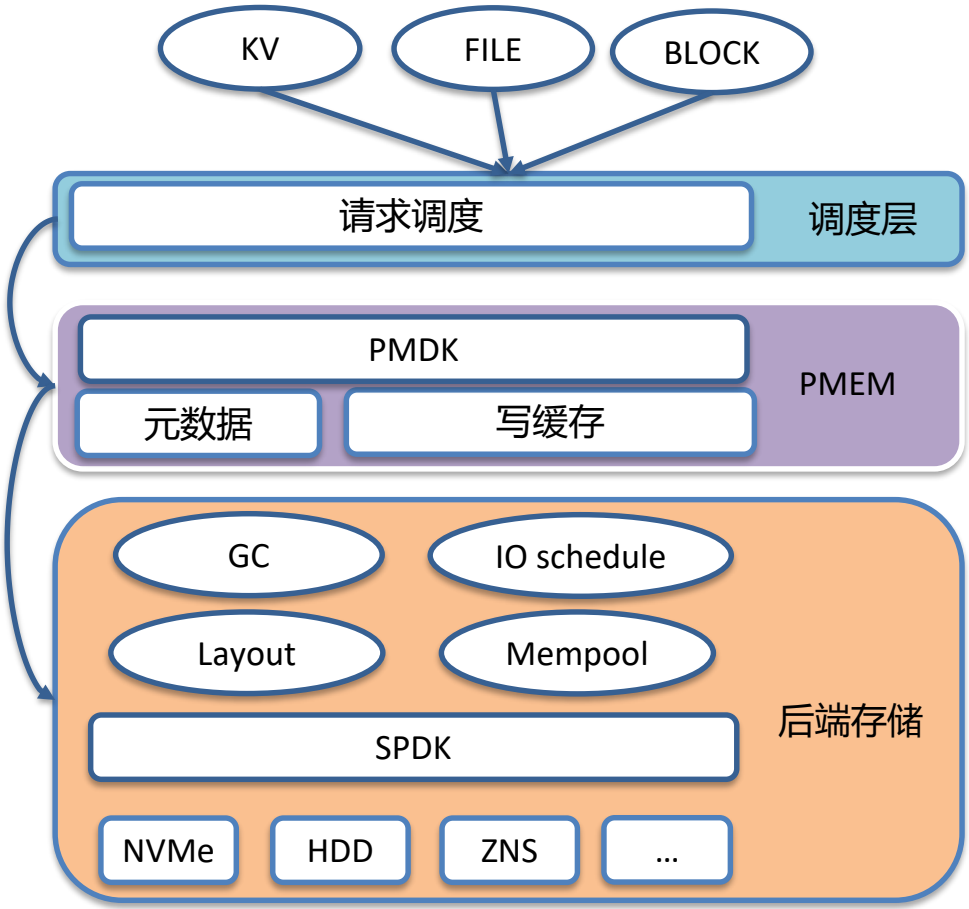
- 新型存储介质越来越多，Nvme、PMEM、OpenChannel、ZNS
- 硬盘容量不断变大，单位性能不断降低

■存储成本

- 数据量越来越大，存储总体成本在持续增长

02 统一单机存储引擎架构

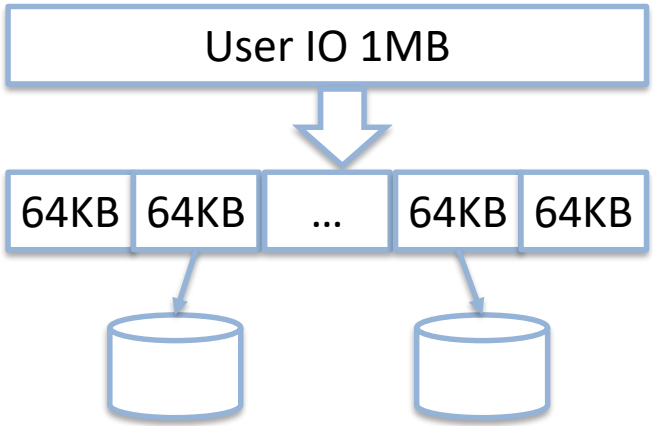
02统一单机存储引擎架构

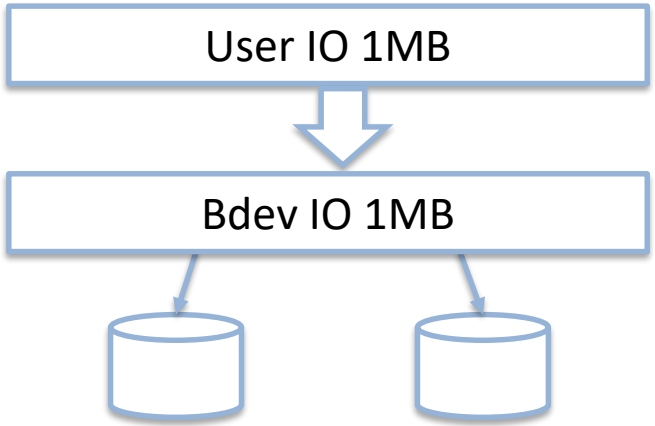


03 SPDK的应用实践

03 SPDK的应用实践

◘ SPDK Bdev 对读写IO默认拆分





Device:	rrqm/s	wrqm/s	r/s	w/s	rMB/s	wMB/s	avgrq-sz	avgqu-sz	await	svctm	%util
sda	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
sdb	0.00	817.50	0.00	299.00	0.00	69.78	477.97	32.64	111.98	3.34	100.00

Device:	rrqm/s	wrqm/s	r/s	w/s	rMB/s	wMB/s	avgrq-sz	avgqu-sz	await	svctm	%util
sda	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
sdb	0.00	0.00	0.00	88.50	0.00	88.50	2048.00	123.48	1219.27	11.30	100.00

问题：SPDK bdev层默认对大io size拆分后，导致单盘带宽利用率偏低，HDD场景尤为明显

解决方法：修改SPDK bdev 参数默认不拆分，修改前,单盘带宽69MB/s，修改后单盘带宽提高到88MB/s。

03 SPDK的应用实践

业务部署需要non-root权限

```
[test@                                    bdevperf]$ ./bdevperf -c bdev.conf -m 0x8000 -o 4096 -w write -t 30 -
Nvme0n1 -q 32
Starting SPDK v20.01.2 git sha1 e73019a8c / DPDK 19.11.2 initialization...
[ DPDK EAL parameters: bdevperf --no-shconf -c 0x8000 --log-level=lib.eal:6 --log-level=lib.cryptodev:5 --log-level=
er1:6 --base-virtaddr=0x200000000000 --iova-mode=va --match-allocations --file-prefix=spdk_pid43773 ]
EAL: No available hugepages reported in hugepages-1048576kB
EAL: VFIO support initialized
app.c: 656:spdk_app_start: *NOTICE*: Total cores available: 1
reactor.c: 334:_spdk_reactor_run: *NOTICE*: Reactor started on core 15
EAL:   using IOMMU type 1 (Type 1)
Running I/O for 30 seconds...
 Logical core: 15
 Nvme0n1             :  580239.83 IOPS    2266.56 MiB/s
 ====================================================
 Total               :  580239.83 IOPS    2266.56 MiB/s
```

问题：依赖SPDK的业务代码在部署时，因为安全要求，通常需要non-root权限运行

解决方法：

1. 底层驱动使用vfio替换uio
2. 初始化EAL --iova-mode=va
3. 使用setup脚本初始化环境时，设置TARGET_USER
4. 修改目标用户memlock权限

03 SPDK的应用实践

◘ SPDK AIO Bdev io hang问题

Device:	rrqm/s	wrqm/s	r/s	w/s	rMB/s	wMB/s	avgrq-sz	avgqu-sz	await	svctm	%util
sdal	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
sdam	0.00	1470.00	0.00	98.00	0.00	98.00	2048.00	56.00	570.15	10.20	100.00

Device:	rrqm/s	wrqm/s	r/s	w/s	rMB/s	wMB/s	avgrq-sz	avgqu-sz	await	svctm	%util
sdal	0.00	1016.00	0.00	210.00	0.00	76.62	747.28	34.04	160.03	4.76	100.00
sdam	0.00	1462.50	0.00	97.50	0.00	97.50	2048.00	40.00	421.15	10.26	100.00

问题：SPDK同一reactor线程上的io channel共用同一个io_context，如果某个aio bdev提交过多的io，其他aio bdev会出现io hang的现象。
解决方法：SPDK同一reactor线程上的io channel共用同一个io_context，如果应用 某个aio bdev提交过多的io，其他aio bdev就会出现io hang的现象。
https://review.spdk.io/gerrit/c/spdk/spdk/+/3808/8

03 SPDK的应用实践

SPDK APP单进程restart

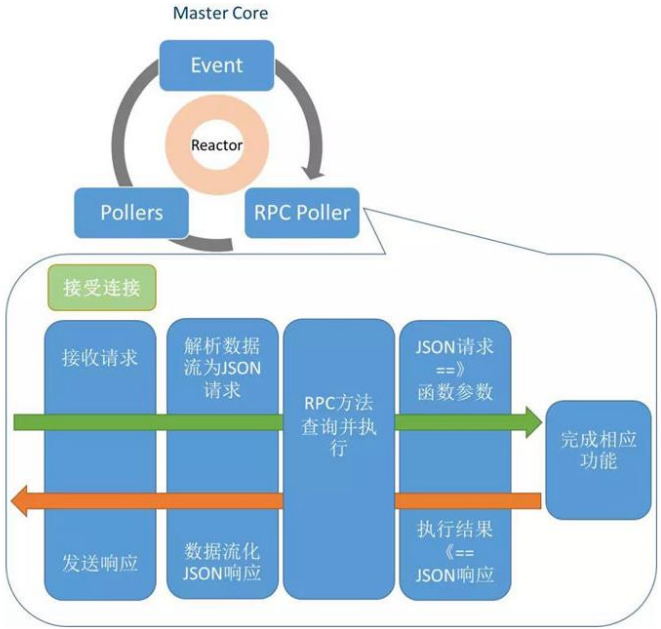
```
hello_bdev.c: 100:read_complete: *NOTICE*: Stopping app
Starting SPDK v20.01.2 git sha1 e73019a8c / DPDK 19.11.2 reinitialization...
app.c: 656:spdk_app_start: *NOTICE*: Total cores available: 1
reactor.c: 334:_spdk_reactor_run: *NOTICE*: Reactor started on core 0
notify.c:  73:spdk_notify_type_register: *ERROR*: Notification type 'bdev_register' already registered.
notify.c:  73:spdk_notify_type_register: *ERROR*: Notification type 'bdev_unregister' already registered.
hello_bdev.c: 199:hello_start: *NOTICE*: Successfully started the application
hello_bdev.c: 217:hello_start: *NOTICE*: Opening the bdev Malloc0
hello_bdev.c: 225:hello_start: *NOTICE*: Opening io channel
hello_bdev.c: 167:hello_write: *NOTICE*: Writing to the bdev
hello_bdev.c: 144:write_complete: *NOTICE*: bdev io write completed successfully
hello_bdev.c: 111:hello_read: *NOTICE*: Reading io
hello_bdev.c:  91:read_complete: *NOTICE*: Read string from bdev : Hello World!

hello_bdev.c: 100:read_complete: *NOTICE*: Stopping app
```

问题：测试需求单进程不退出，实现spdk app stop，start过程。
解决方法：修改spdk app start逻辑，判断是否为同一进程内重启，选择性初始化spdk env。
https://review.spdk.io/gerrit/c/spdk/spdk/+/2260

03 SPDK的应用实践

◘ SPDK RPC动态配置实现在线上下盘



问题：为降低单盘故障导致的数据迁移成本，单机引擎需实现在线上下盘功能，只替换故障盘

解决方法：单机引擎底层磁盘管理功能使用SPDK提供的RPC cmd动态创建和销毁Bdev，不依赖静态config文件，实现在线上下盘功能。

03 SPDK的应用实践

NVMe盘smart监控

```
[root@ nvme-cli-spdk-1.6]# ./nvme intel smart-log-add 0000:d8:00.0
Additional Smart Log for NVME device:0000:d8:00.0 namespace-id:ffffffff
key                               normalized raw
program_fail_count              : 100%       0
erase_fail_count                : 100%       0
wear_leveling                   : 100%       min: 21, max: 42, avg: 29
end_to_end_error_detection_count: 100%       0
crc_error_count                 : 100%       0
timed_workload_media_wear       : 100%       63.999%
timed_workload_host_reads       : 100%       65535%
timed_workload_timer            : 100%       65535 min
thermal_throttle_status         : 100%       0%, cnt: 0
retry_buffer_overflow_count     : 100%       0
pll_lock_loss_count             : 100%       0
nand_bytes_written              : 100%       sectors: 2805772
host_bytes_written              : 100%       sectors: 2374149
```

问题：Nvme盘切换成SPDK用户态驱动后，原有的硬件监控工具无法对硬件状态进行监控

解决方法：通过spdk专有nvme cli实现用户态驱动NVMe盘的smart监控

04 下一步工作

04 下一步工作

- 和RDMA做定制优化
- 探索更多存储产品类型，比如ZNS
- 提供压缩等更多特性
- …

THANKS