自适应内存异步回收

曾文斌
快手内核组

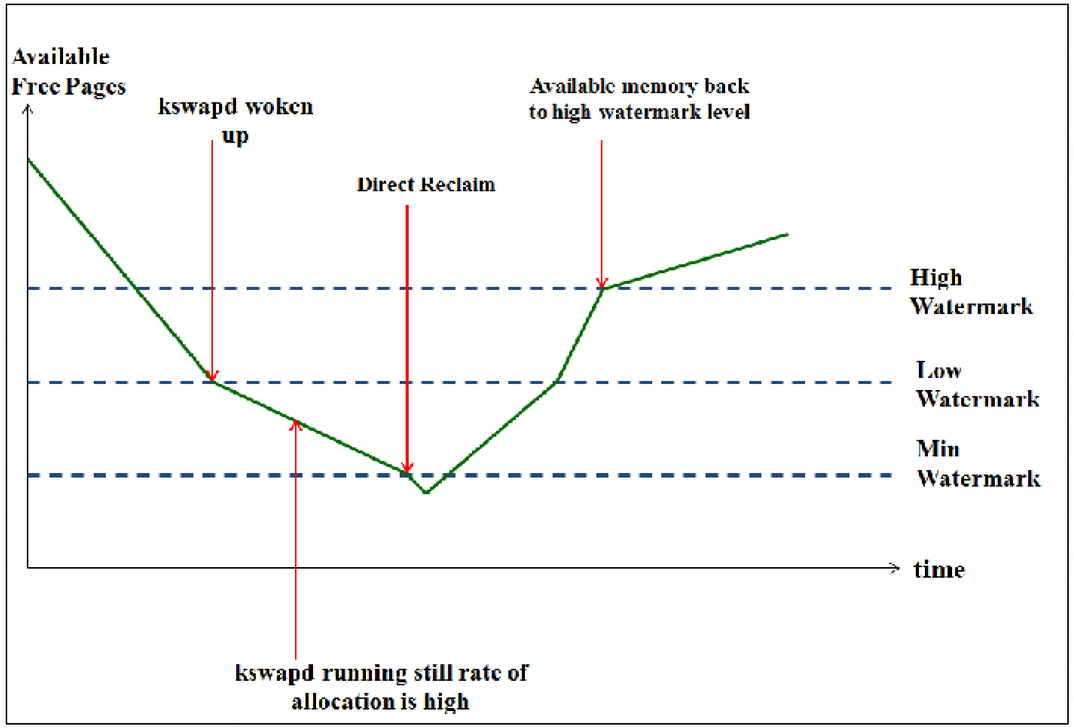
快手

内存回收在云计算场景下面临的挑战

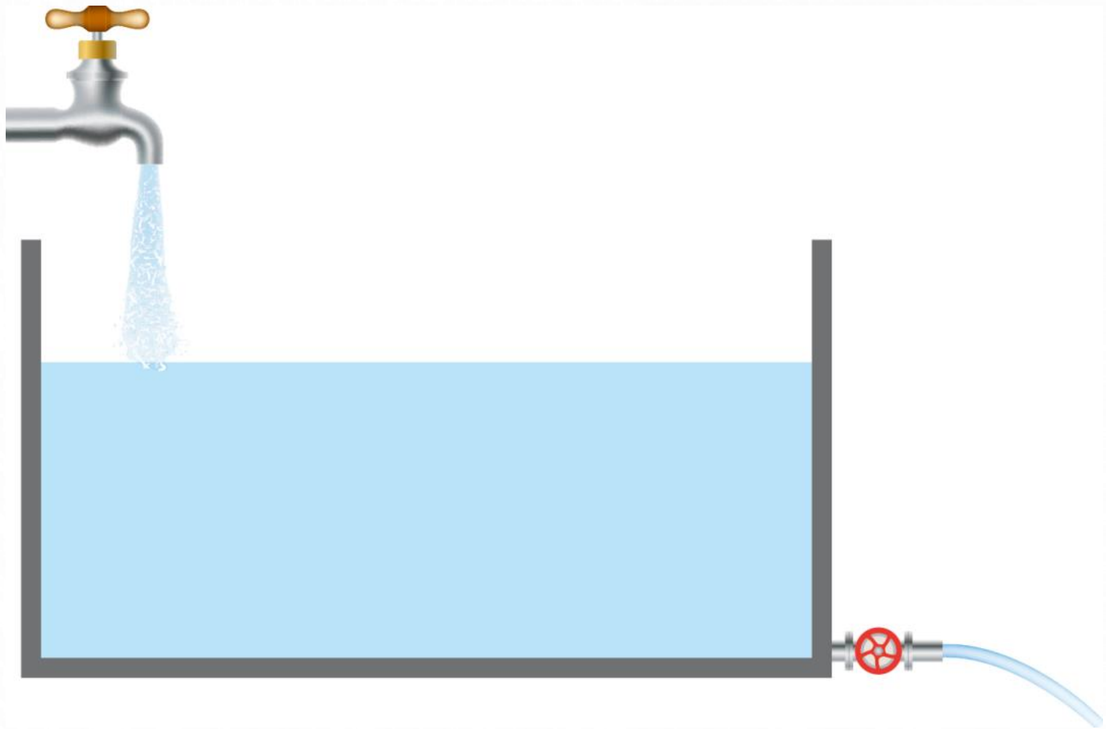
- 多种业务混合部署、动态调度
- 单一的内存回收机制难以自动适应不同的业务
- 直接回收(DIRECT RECLAIM)时有发生，导致内存分配延迟大

内存回收的两条路径

- 异步回收 KSWAPD
  - 启动点：低水位
  - 停止点：高水位
- 直接回收 DIRECT RELAIM
  - 触发点：最低水位
  - 导致内存分配延迟



避免直接回收的思路



- 提高KSWAPD回收速度
- 提前启动KSWAPD
  - 原生的静态调控方式：
  - vm.min_free_kbytes
  - vm.watermark_scale_factor

静态调控难以适应多变的场景

- 回收不足和过度回收都有损性能
- 不存在静态的最优回收参数
  - 高峰和低谷所需的回收力度不同
  - 不同的业务所需的回收力度不同
    - 高IO型的业务通常消耗PAGE CACHE较快
    - 计算型的业务通常消耗PAGE CACHE不多

自适应内存异步回收机制

目标

- 自动调整内存回收力度到最佳状态
- 既要避免回收不足，也要避免回收过度
- 尽量减少对原有机制的侵入

实现方法

- 动态感知回收效率与内存消耗速度的差距
- 复用watermark_scale_factor机制
- 自动调整kswapd的启动时机
  - adaptive scale up
  - adaptive scale down

自适应内存异步回收算法
ADAPTIVE SCALE UP

- 时机：kswapd休眠之前
- 触发条件：发生过direct reclaim
- 限制条件：kswapd_failures不超过阈值
- 计算上调幅度
  - 因素1：新增的direct reclaim数量
  - 因素2：LRU file和reclaimable slab的数量

自适应内存异步回收算法
ADAPTIVE SCALE DOWN

- 时机：SCALE-UP保护期过后
  - SCALE-UP发生时建立一个workq任务，定时检查
- 触发条件：
  - 不再发生direct reclaim
  - kswapd的cpu使用率持续低于阈值
- 计算下调幅度
  - 因素1：kswapd的cpu使用率
  - 因素2：持续时间

自适应内存异步回收算法
避免副作用

- watermark_scale_factor变化会导致calculate_totalreserve_pages()重新计算totalreserve_pages，进而影响到下列逻辑：
  - /proc/meminfo的MemAvailable会减少
  - vm.overcommit_memory=0时，分配内存更容易返回内存不足
- 解决办法：在启用自适应内存异步回收时，calculate_totalreserve_pages()不更新totalreserve_pages。

观测内存回收效率

- /proc/vmstat
  - allocstall_*
  - pgsteal_direct
  - pgsteal_kswapd
  - pgscan_direct
  - pgscan_kswapd
  - kswapd_low_wmark_hit_quickly
  - kswapd_high_wmark_hit_quickly
  - pgalloc_*
  - pgfree

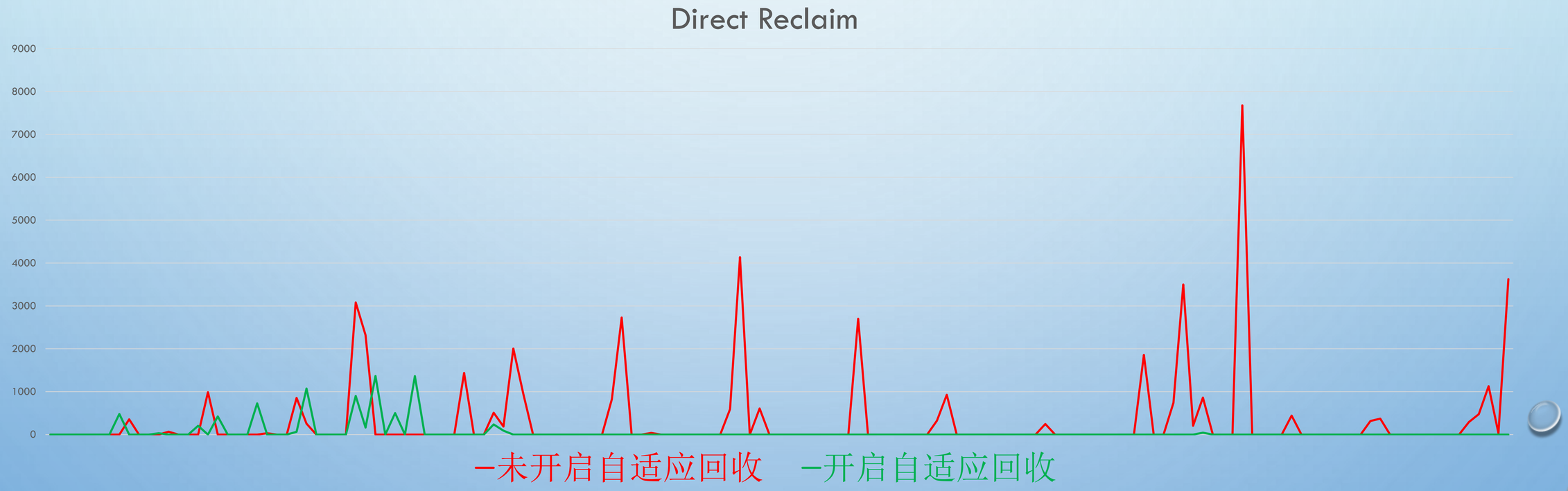
- 自制观测工具 cachemon

```
01:14:02 anon_act anon_inact   file_act file_inact unevict  slab_rec slab_unrec
           10075       4155      10419      43262       0      8387       7792

       stall dsteal  dscan dthrt  ksteal    kscan lqck hqck  pgalloc   pgfree
           0      0      0    0        0        0    0    0        5         4

        node   zone      high       low       min      free    (+/-) kswapd
           0    DMA        32        27        22      3977        0
           0  DMA32      5955      4963      3971    724940        0
           0 Normal     10586      8822      7058   1232830        0     0%
           1 Normal     17211     14343     11475   2003902        0     0%
```

压测程序

```
static int read_file(char *fname, unsigned long length)
{
	int fd;
	char *addr, c;
	unsigned long i;
	// mmap file
	fd = open(fname, O_RDWR, (mode_t)0600);
	if (fd == -1) {
		perror("open");
		return 1;
	}
	addr = mmap(0, length, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
	if (addr == MAP_FAILED) {
		perror("mmap");
		return 1;
	}
	printf("total %d pages\n", length / 4096);
	// loop and dirty each of the pages:
	while (1) {
		for (i = 0; i < length / 4096; i++) {
			c = *(addr + i * 4096);
			c = *(addr + i * 4096 + 1024);
			c = *(addr + i * 4096 + 2048);
			if (stress > 0 && i % stress == 0)
				usleep(FREQ);
		}
	}

	// cleanup
	munmap(addr, length);
	close(fd);
	return 0;
}
```

自适应内存异步回收的效果



改进空间

- 属于系统全局性的回收，不具备保护重点业务的能力
- 未触及异步回收的效率改善

THANKS

快手