基于openEuler内核的多线程亲和性调度优化实践

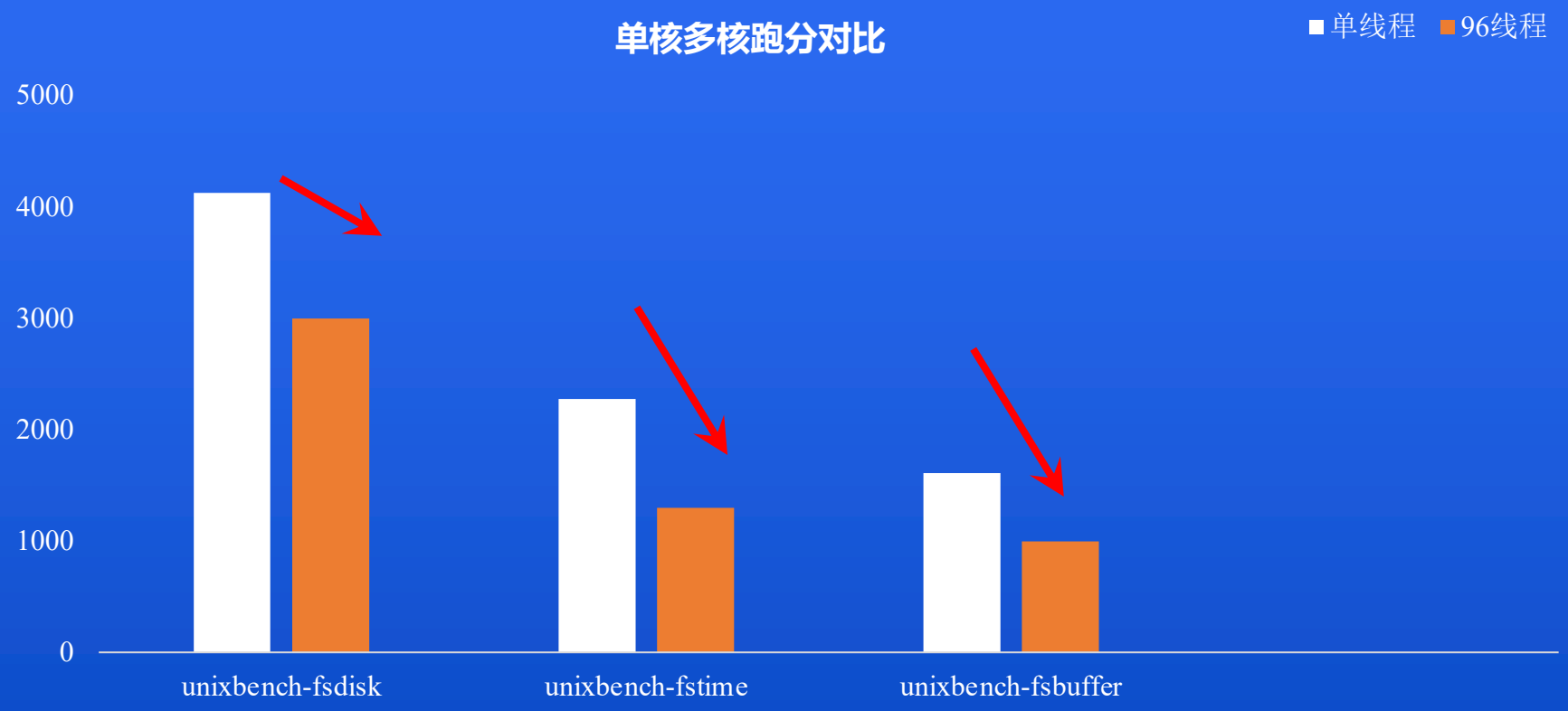
麒麟信安资深内核工程师 李强

开放原子开源基金会 OPENATOM FOUNDATION | OpenEuler

目录

- 多线程应用性能瓶颈分析
- 当前内核调度的不足与可行的优化方法
- 优化成果UnixBench跑分展示

多线程应用性能瓶颈——现象



使用UnixBench对鲲鹏920服务器进行单线程/96线程性能测试，以其中fsdisk、fstime、fsbuffer三项为例，在鲲鹏920上测试单线程和多线程（96线程）的跑分对比，从常理上，本以为多线程的跑分应该要多于多线程，但是实际的测试结果却显示，96线程反而要少于单线程。

多线程应用性能瓶颈——软件原因

使用Perf对正在进行96线程fsdisk测试的程序进行监控，分析发现问题在于96个线程在竞争一个iNode的写锁，发现大量的CPU处于rwsem_optimistic_spin忙等待写锁的状态，导致了多线程fsdisk测试项的性能不及预期。通过分析确定fsdisk测试单线程持有iNode时间占比为43%，则使得多线程持锁时间100%即为理论多线程的最大性能跑分，通过以下公式计算理论多线程fsdisk的最大跑分：

$$u = t/T$$

$$P = 1/u \times p$$

u：单线程一个测试周期测试持锁时间占比

t：一个测试周期内持锁时间

T：一个测试周期时间

P：多核理论最大性能跑分

p：单核性能跑分

将单核跑分带入p=4125，u持锁时间占比43%。得到理论最大fsdisk性能P为9593。且fsdisk测试线程理论最大使用的CPU为3。多余的忙等待是完全没有必要的。

```
Samples: 505K of event 'cycles:ppp', Event count (approx.): 1317709446187
  Children      Self  Command          Shared Object            Symbol
+   99.91%     0.01%  fstime           fstime                   [.] c_test
+   99.85%     0.00%  fstime           [kernel.kallsyms]        [k] el0_svc
+   99.85%     0.00%  fstime           [kernel.kallsyms]        [k] el0_svc_handler
+   99.85%     0.06%  fstime           [kernel.kallsyms]        [k] el0_svc_common
+   99.51%     0.03%  fstime           libc-2.28.so             [.] write
+   99.43%     0.00%  fstime           [kernel.kallsyms]        [k] ksys_write
+   99.43%     0.00%  fstime           [kernel.kallsyms]        [k] __arm64_sys_write
+   99.43%     0.00%  fstime           [kernel.kallsyms]        [k] vfs_write
+   99.40%     0.01%  fstime           [kernel.kallsyms]        [k] __vfs_write
+   99.39%     0.00%  fstime           [kernel.kallsyms]        [k] ext4_file_write_iter
+   98.23%     0.13%  fstime           [kernel.kallsyms]        [k] down_write
+   98.10%     0.06%  fstime           [kernel.kallsyms]        [k] rwsem_down_write_failed
+   98.03%     0.18%  fstime           [kernel.kallsyms]        [k] rwsem_optimistic_spin
+   96.76%    92.01%  fstime           [kernel.kallsyms]        [k] osq_lock
+   95.73%     0.00%  fstime           fstime                   [.] _start
+    5.04%     5.04%  fstime           [kernel.kallsyms]        [k] __native_vcpu_is_preempted
+    0.70%     0.66%  fstime           [kernel.kallsyms]        [k] rwsem_spin_on_owner
+    0.63%     0.01%  fstime           [kernel.kallsyms]        [k] __generic_file_write_iter
+    0.54%     0.00%  fstime           [kernel.kallsyms]        [k] generic_perform_write
```

当前内核的处理方法存在的不足

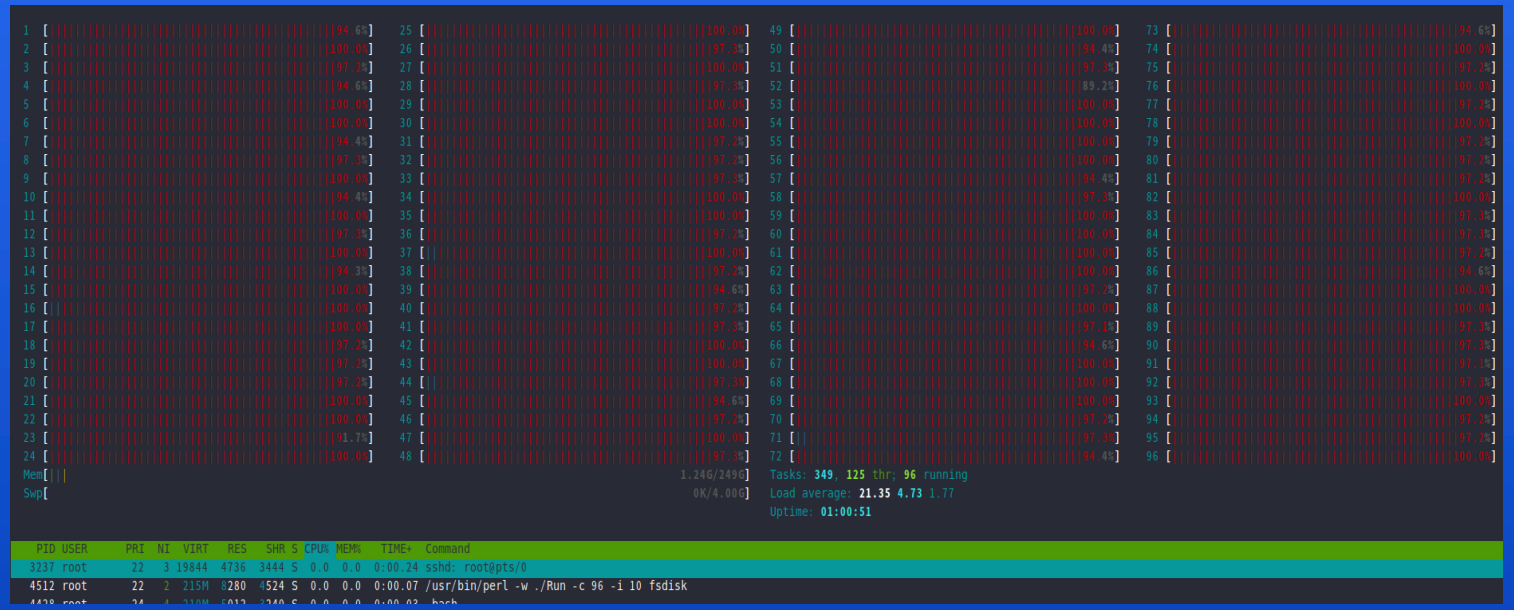
理论值： 在鲲鹏920上fsdisk 96线程的测试项理论有效使用的最大CPU为3个，理论最大的fsdisk性能跑分应该在9593。

实际上： 使用htop对正在进行96线程fsdisk测试的程序进行监控，发现每个CPU的占用率都接近100%，使得绝大部分CPU在忙等待。

影响范围：

- 占用了CPU资源；
- 增加了系统能耗；
- 使得锁与资源同步发生在跨NUMA数据访问。

从而使得fsdisk多线程跑分比单线程还低。



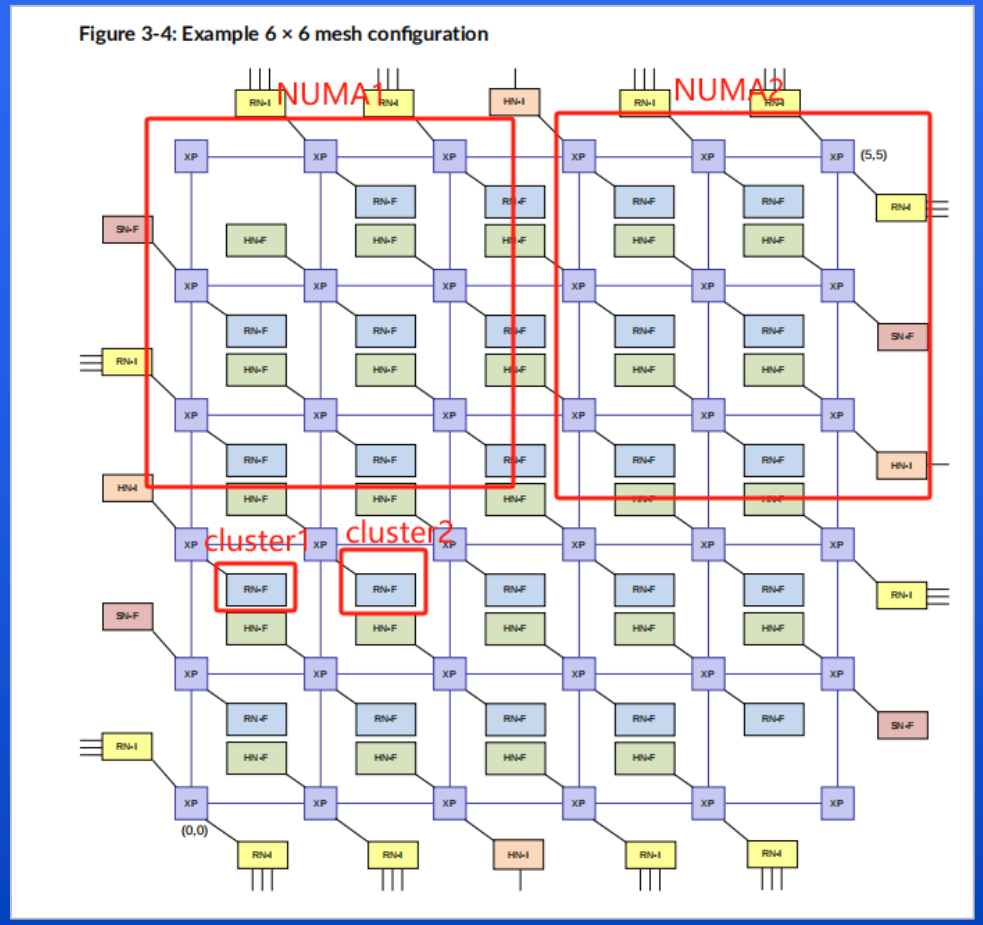
服务器芯片内部结构介绍与可行的优化方法

问： 鲲鹏920上fsdisk 96线程测试项理论最大性能跑分是9000多，那为什么实际测试得到的结果只有3000分呢？

答： 因为每个fsdisk测试线程都在一个CPU上执行，共享资源在不同CPU之间流转，会不断的出现跨NUMA数据同步，而导致多线程性能进一步降低，从而会出现多线程性能低于单线程的情况。

- Cluster内通信：数据通路最近，内部总线，L2 cache
- NUMA内通信：数据通路稍近，共享LLC cache
- DIE内通信：数据通路稍远，跨物理内存通信
- SOCKET内通信：数据通路远，跨物理内存通信

例： 以arm64服务器芯片内部mesh结构为例，临近CPU之间不仅总线通信距离近，而且存在共享Cache的情况，从而性能有效提升。而CPU之间的数据同步花费是随着CPU物理架构上的亲和性而增加的（即cluster<numa<DIE<SOCKET），所以根据多线程的负载对CPU的需求，将关联线程集中到最合适的调度域中可以有效的提升多线程的性能。



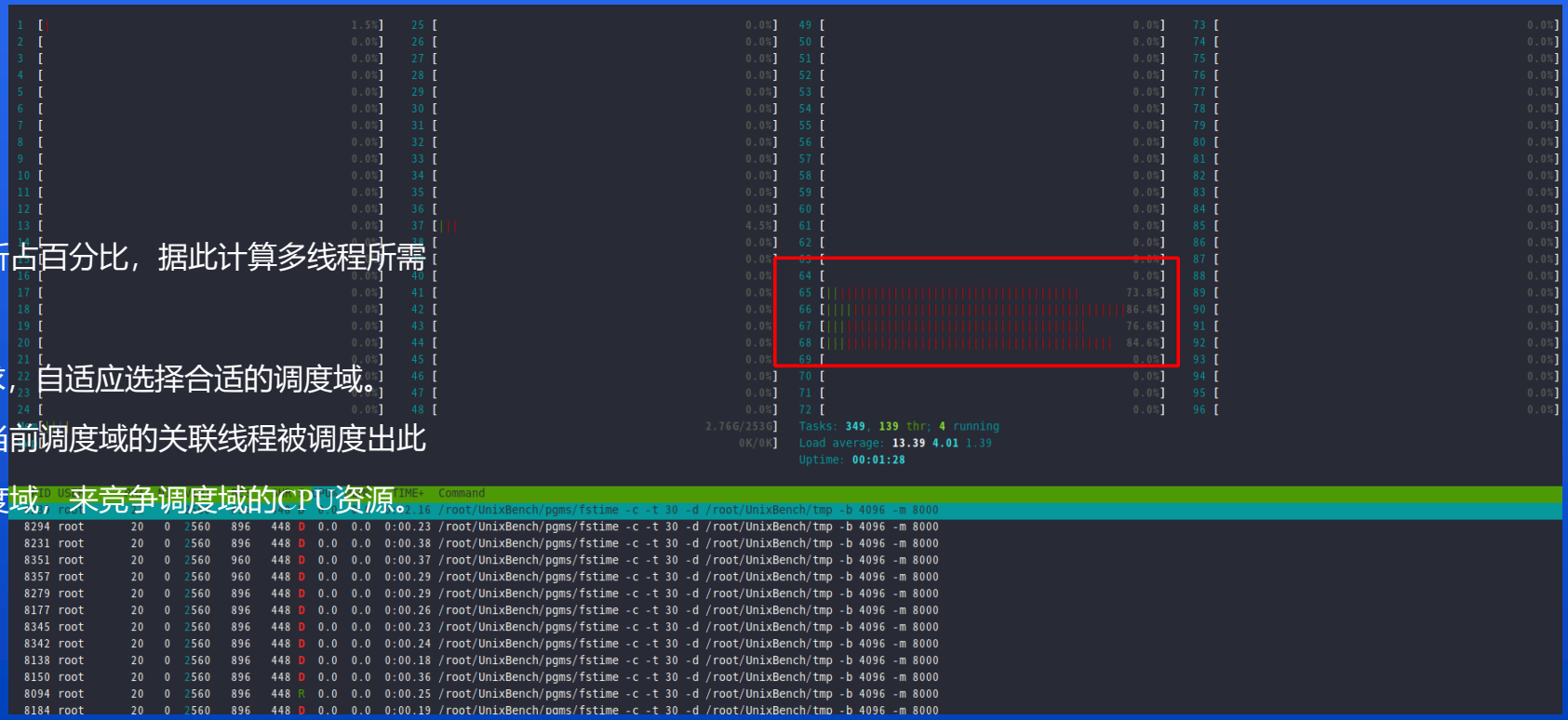
优化方法介绍

优化原理：基于原有的CFS调度器框架，优化了内核进程调度算法，在不增加进程调度延时的情况下，把相互关联（占用同一资源）的应用线程尽可能的集中到（唤醒关联线程时调度到）最合适的调度域中，使得应用的关联线程之间的数据同步花费降到了最低。

成果检验：在包含了此优化patch的麒麟信安内核上，再进行fsdisk测试时，使用htop所有cpu的负载情况，发现fsdisk测试线程都被集中到一个cluster中。从而降低了测试线程之间的数据同步消耗。

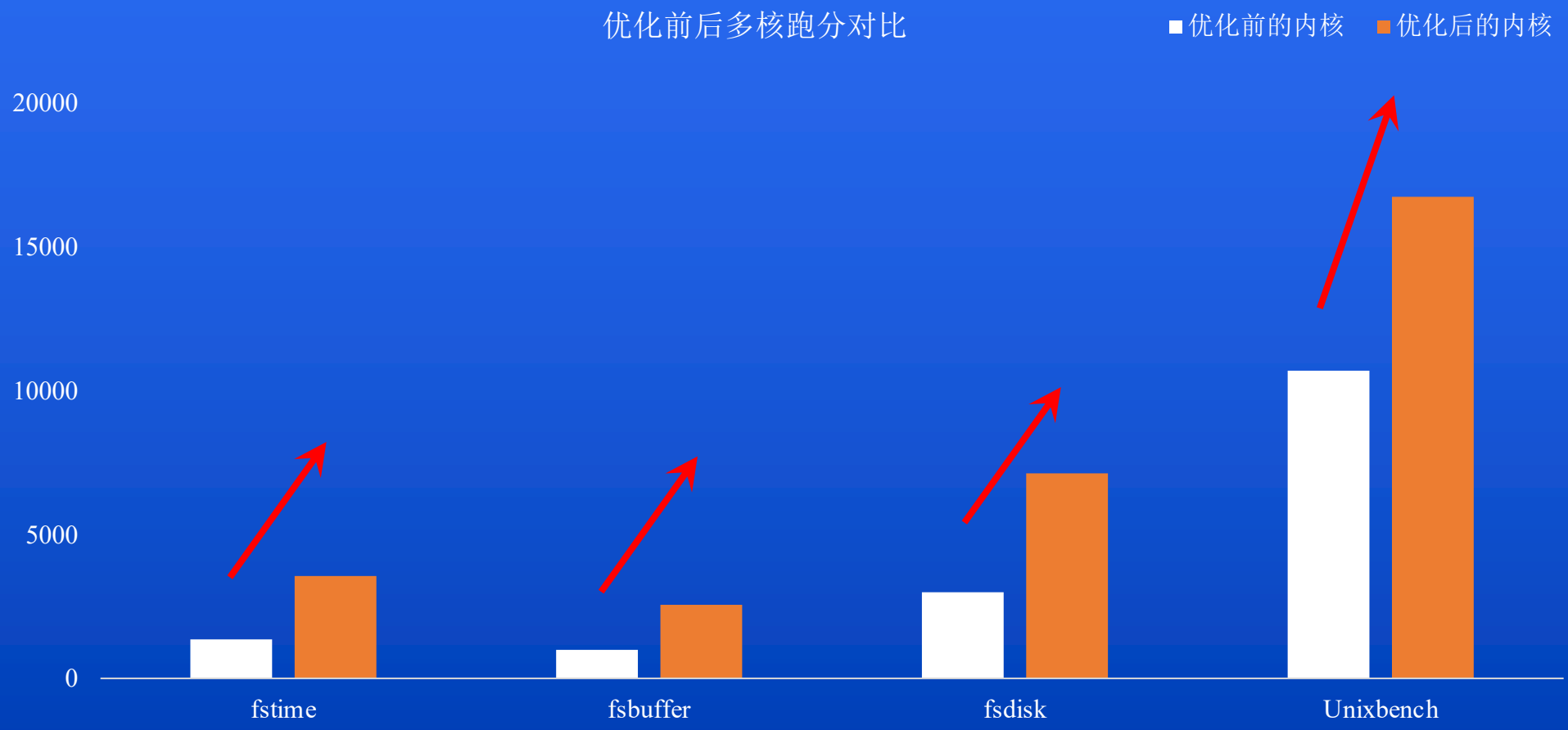
关键功能：

- 分辨关联线程，根据单个线程持锁时间所占百分比，据此计算多线程所需的CPU个数。
- 遍历所有的CPU调度域，根据CPU的需求，自适应选择合适的调度域。
- 隔离出调度域，不参与负载均衡，避免当前调度域的关联线程被调度出此调度域。且阻止其他任务被调度到此调度域，来竞争调度域的CPU资源。



优化成果

由优化成果看，麒麟信安操作系统优化后的内核UnixBench 96线程跑分比优化前大幅提升。



THANKS

THANKS

THANKS