鲲鹏软件性能调优

前言

本文从硬件特点分析如何进行性能调优，同时还介绍了项目中性能调优的思路和常用性能采集工具。

目标

学习本课程后，你可以了解到：

- 硬件特性会对软件性能带来哪些影响，如何结合硬件特性发挥最大性能
- 性能调优的思路和常用的性能采集工具

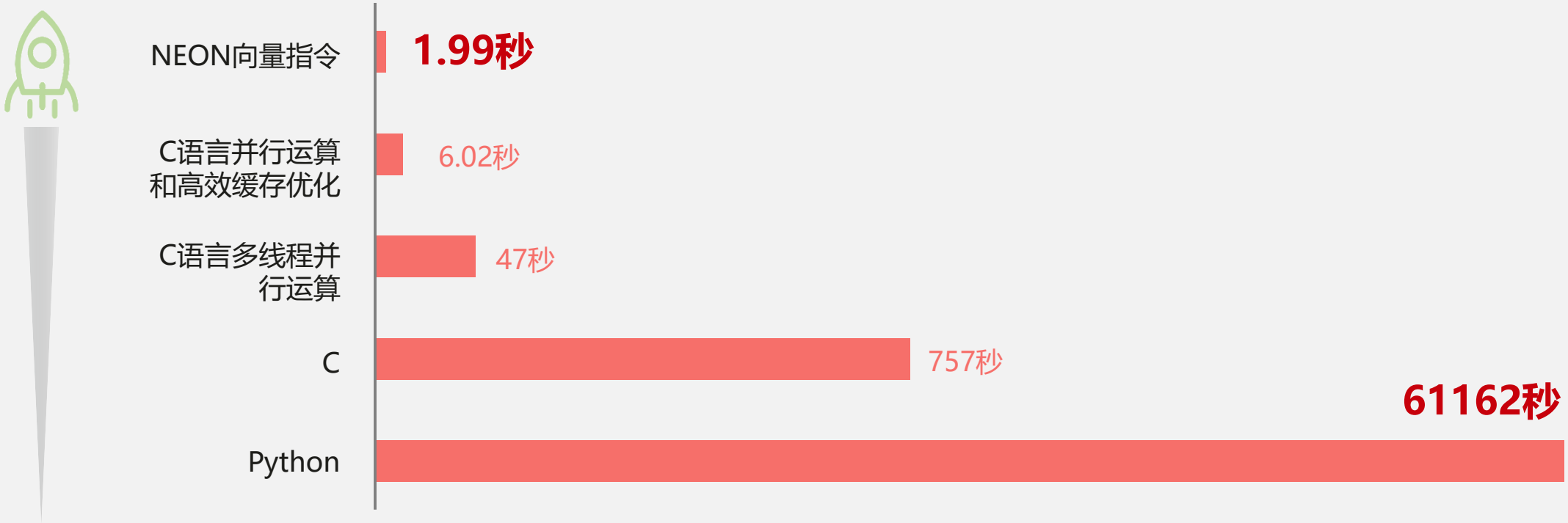
目录

基于硬件特性的性能调优方向

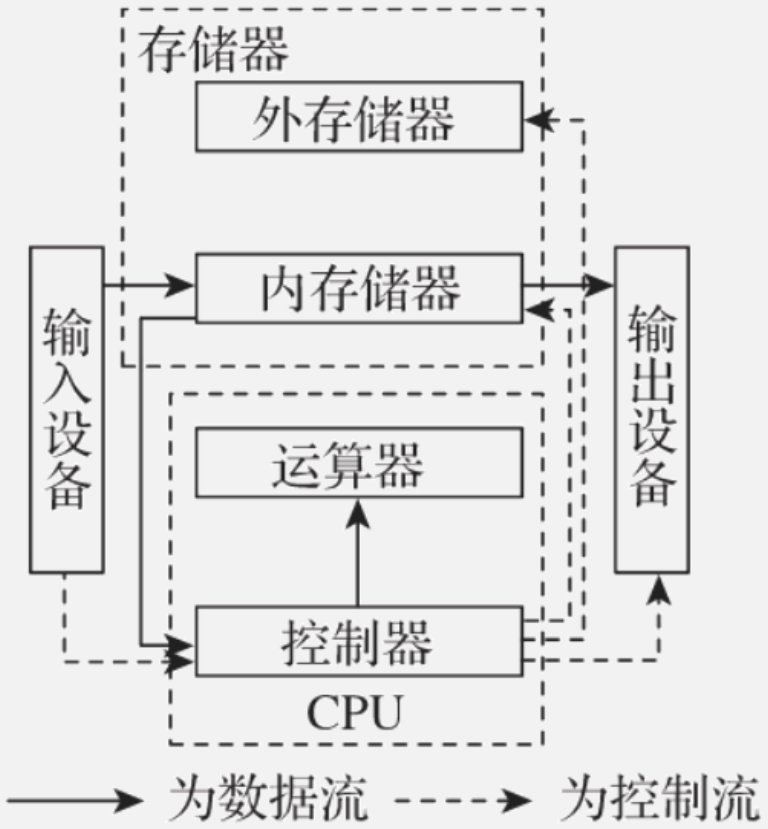
MariaDB性能调优案例

软硬协同带来万倍代码性能提升

4800*4800 矩阵乘法加速效果实测结果



从冯诺依曼架构看性能调优方向



性能优化四个方向

CPU/内存

磁盘

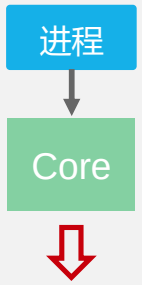
网卡

应用

基于鲲鹏处理器的软加速和硬加速

软加速

单核加速

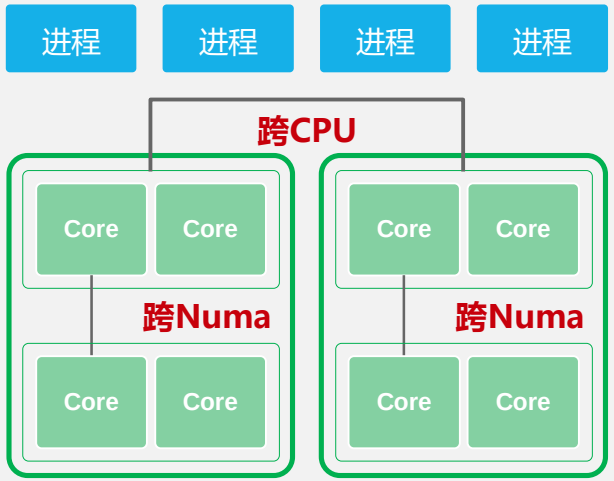


- 寄存器分配
- 指令布局
- 指令流水
- 迭代编译

1

编译优化

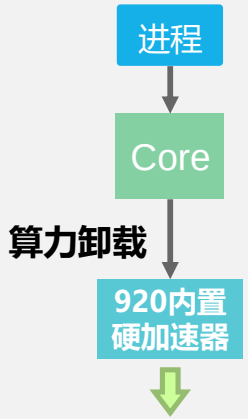
多核加速



2

NUMA-Aware亲和性优化

硬加速

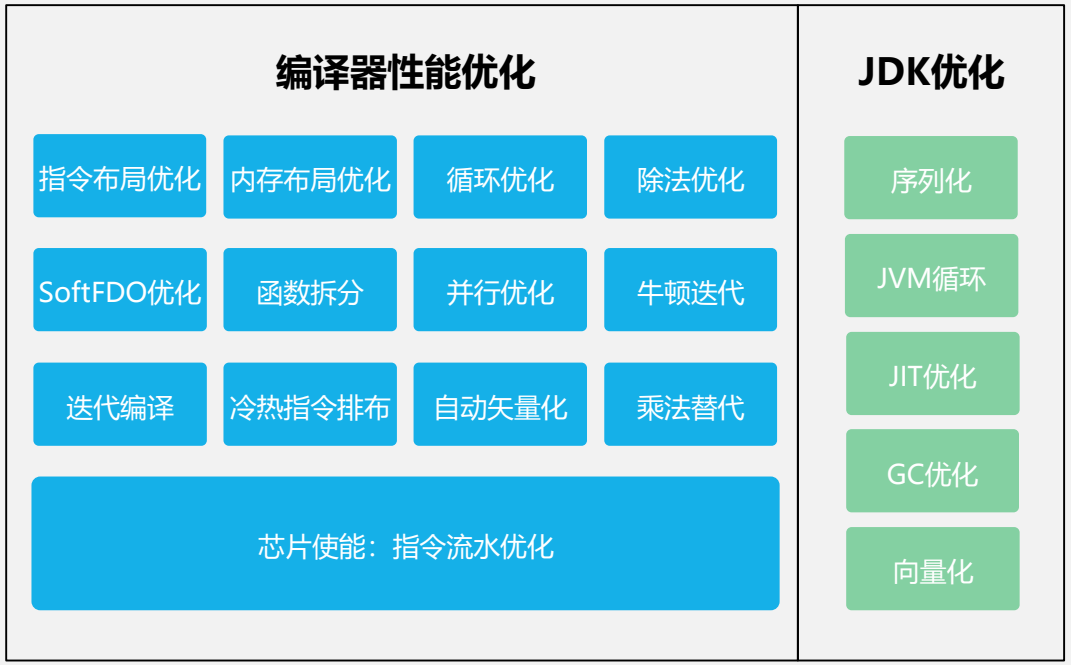


- 加解密
- 压缩/解压缩
- 大数据幂运算
- EC（纠删码）

3

硬件加速引擎

编译器：基于鲲鹏芯片微架构，构建鲲鹏极致性能和完善生态



编译器性能优化：

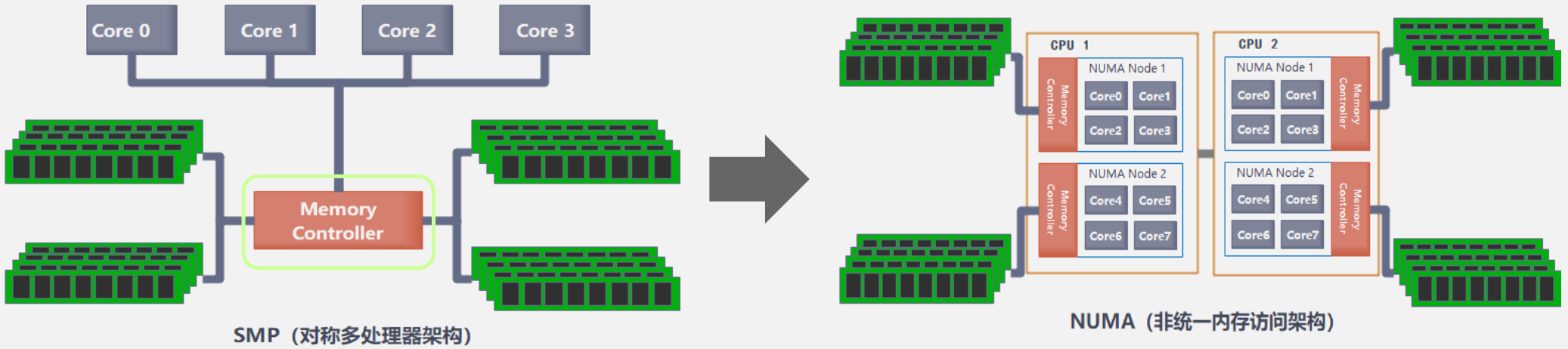
- **指令布局优化**：拆分函数代码，按照冷热指令重新排布，提升指令Cache命中率
- **内存布局优化**：按照内存数据访问频度，组合热数据区域，提升数据Cache命中率
- **循环优化**：分析循环迭代间数据访存依赖关系，对无依赖的循环并行到多核执行，无依赖的数据自动矢量化计算，加速程序运行

JDK性能优化：

- JIT编译优化、GC内存回收管理优化提升内存管理性能
- JVM循环、向量化、序列化技术，提升程序执行性能

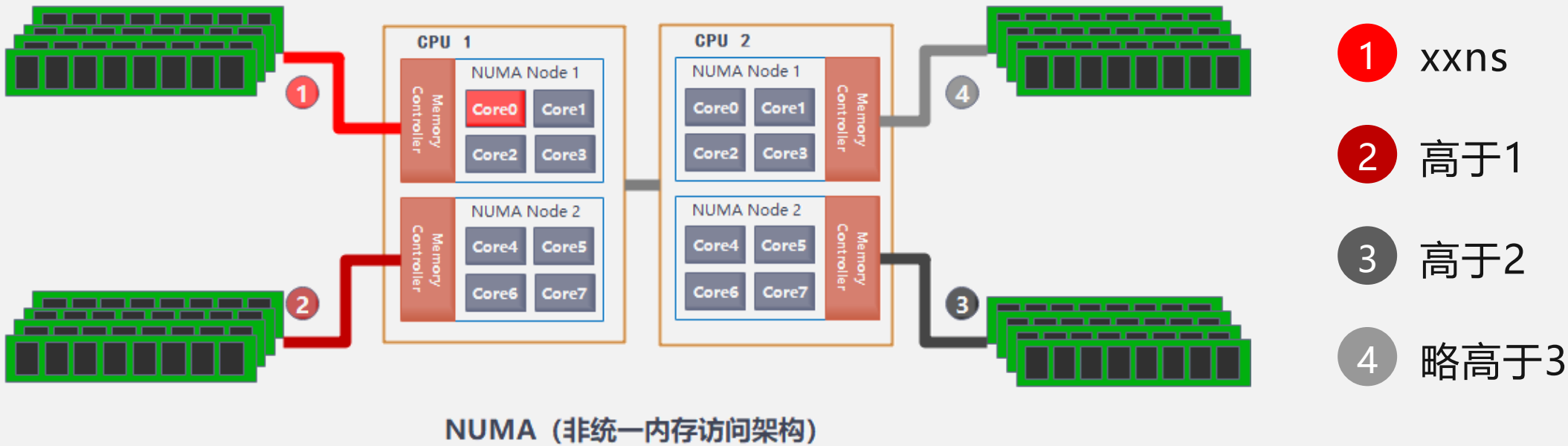
NUMA是CPU发展的一种必然趋势

在SMP系统中，核数的扩展受到内存总线的限制。非统一内存访问架构（Non-uniform memory access）很好的解决了这一问题。



发挥NUMA性能，需要克服内存访问速度不均匀的挑战

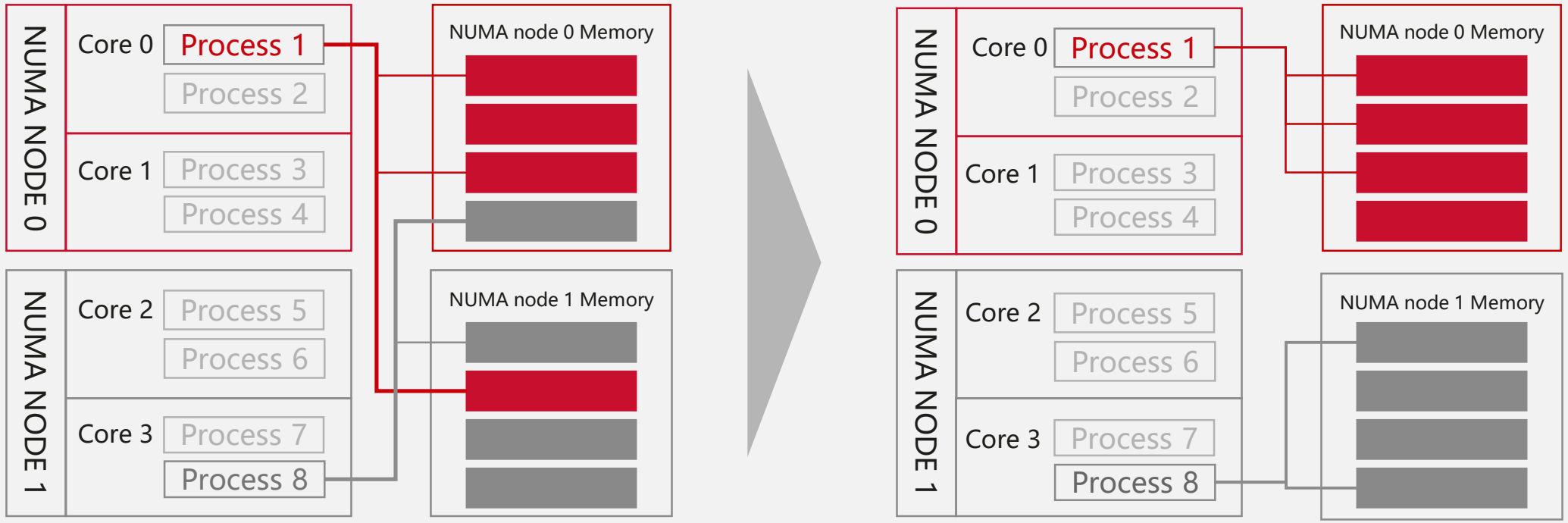
内存在物理上是分布式的，不同的核访问不同内存的时间不同。



NUMA（非统一内存访问架构）

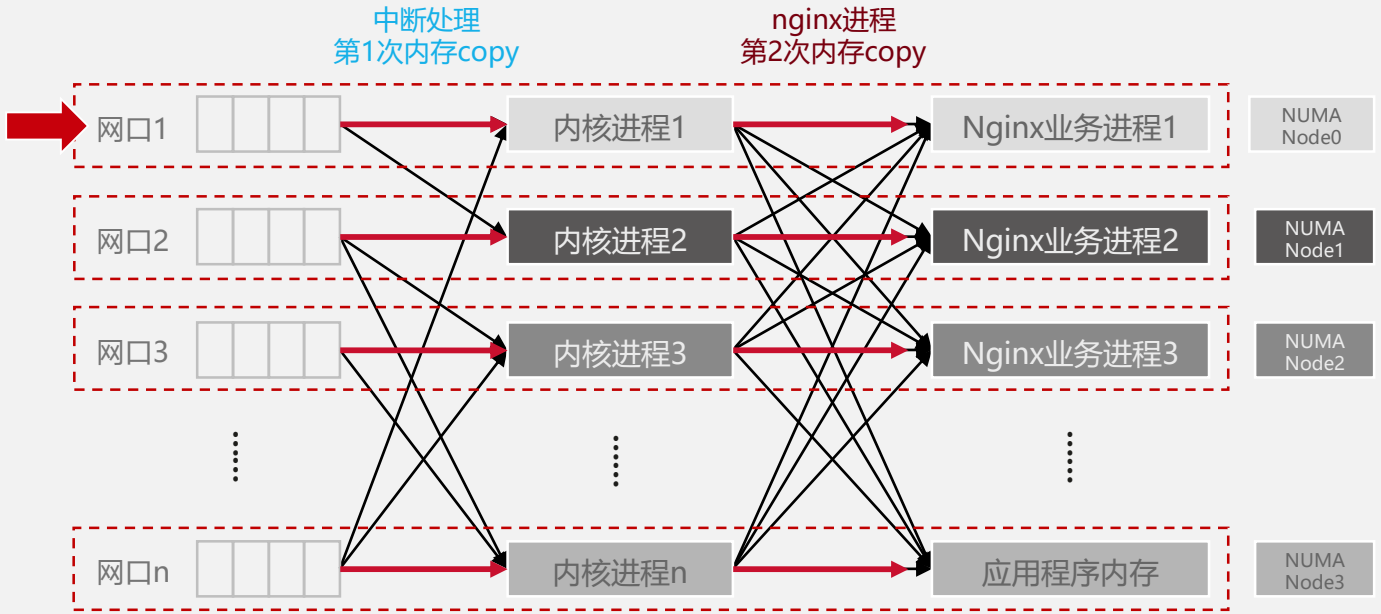
NUMA-Aware亲和性资源规划，让内存访问最短路径

利用CPU亲和性与内存分配策略，让进程与内存的距离更“短”



Nginx绑核优化举例

将Nginx进程分布到各个NUMA node之内，让系统整体的负载比较均衡，按照中断号将中断服务和Nginx绑定在一个NUMA内。性能将会有非常明显的提升。



三种NUMA绑核配置方法

1、使用系统工具numactl设置

numactl -C 0-15 process name

-C：Core scope

2、在代码中调用亲和性设置参数

int sched_setaffinity(pid_t pid, size_t cpusetsize, cpu_set_t *mask)

3、多数开源软件中提供了配置接口

nginx.conf文件中worker_cpu_affinity参数

基于鲲鹏技术优势构建加速库，实现软硬加速互补

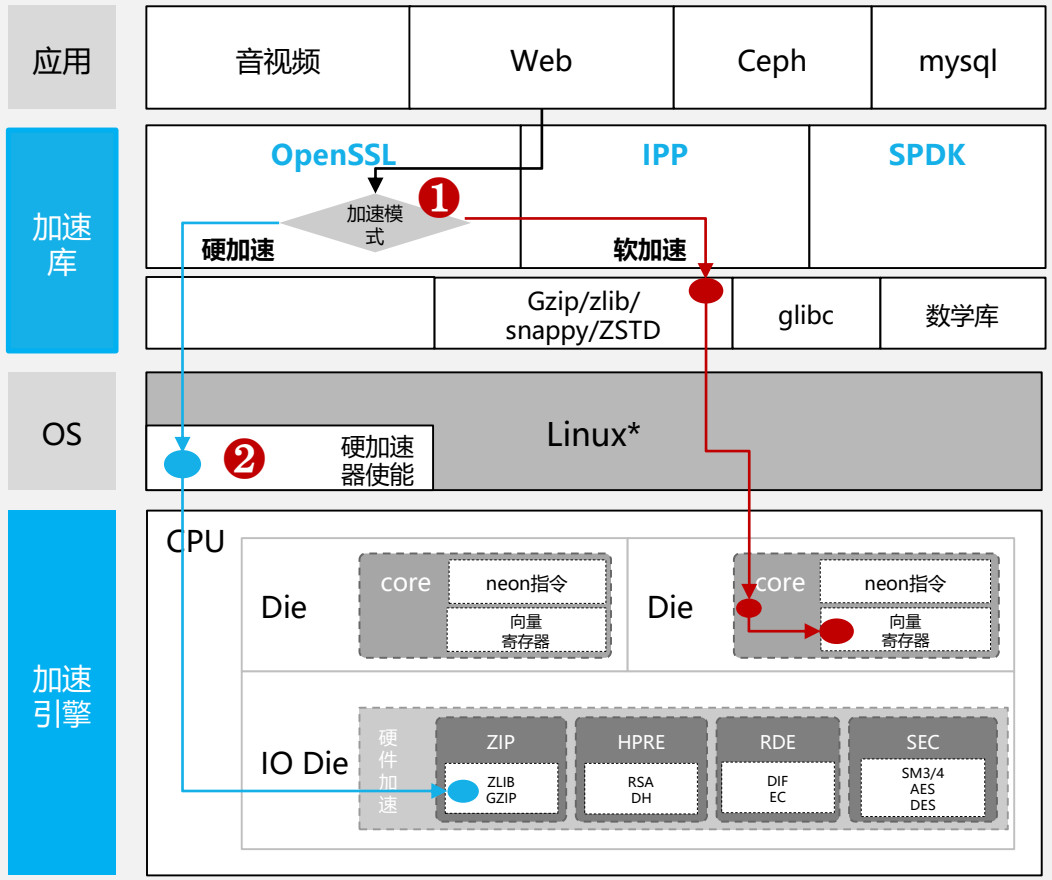
针对四类业务提供9大加速库，典型场景10%-100%性能提升

基础加速	压缩加速	加解密加速	多媒体加速
性能提升>17%	性能提升>20% (ceph)	性能提升>100% (nginx)	性能提升>10%
glibc	gzip/zlib	OpenSSL	X265
Hyerscan	ZSTD		ffmpeg
	snappy		HMPP

openEuler

Kunpeng

软硬加速库相结合，构筑鲲鹏高性能软件栈



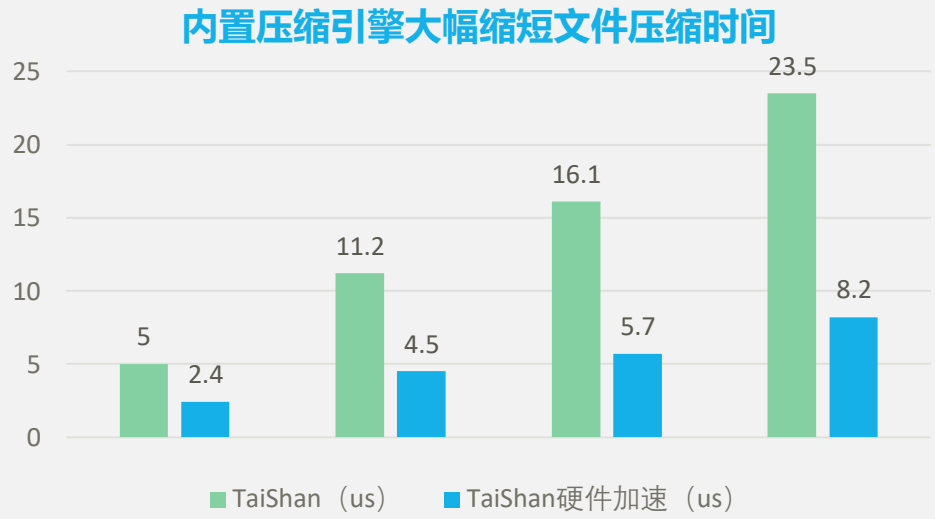
基于鲲鹏微架构，使用硬件加速器及Neon指令对业界主流软件库进行加速重构，**构筑鲲鹏高性能软件栈**

1 业务&基础软件库加速使能

- 主流基础库支持加速改造
- 关键场景应用库加速改造

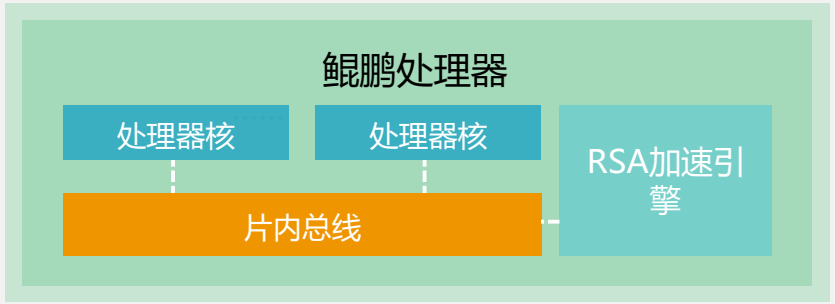
2 内核态硬加速器件使能

- 硬加速部件适配内核驱动

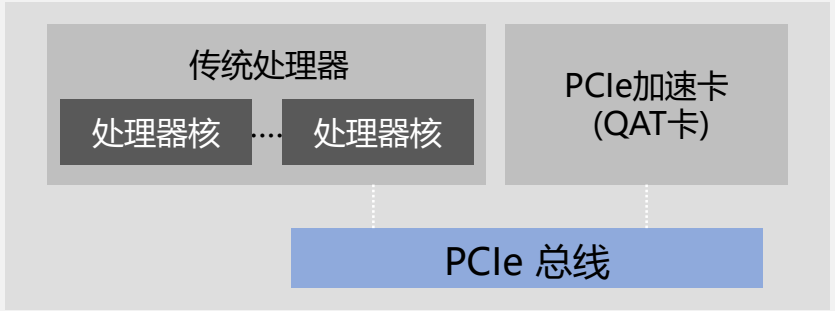


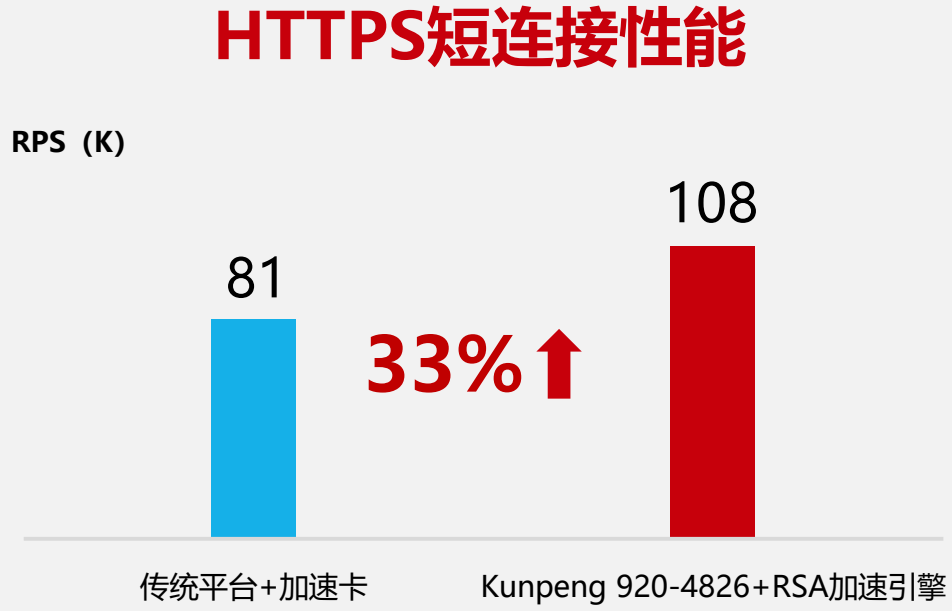
调用鲲鹏RSA加密加速引擎，提升Web应用Https性能

鲲鹏
加速方案

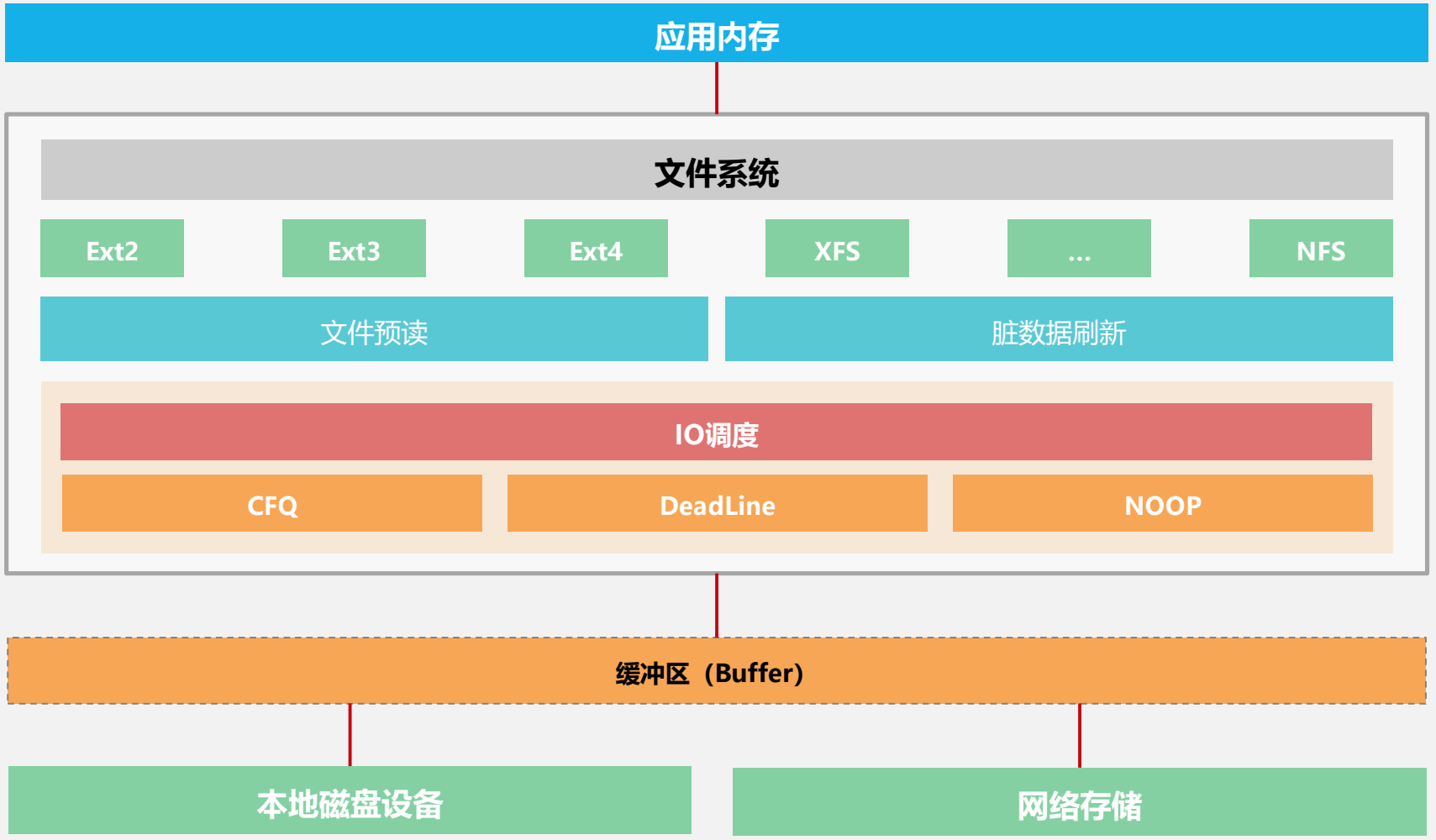


传统
加速方案





文件系统决定了磁盘加载到内存过程的快慢



磁盘预取可以充分利用磁盘带宽

Command：Get A、B、C、D Value

Cache			
H	I	J	K
A	B	C	D

四次I/O请求

A			
B			
C			
D			

未开启磁盘预取

Command：Get A、B、C、D Value

Cache			
H	I	J	K
A	B	E	F
Q	C	D	V

两次I/O请求

A	B	E	F
Q	C	D	V

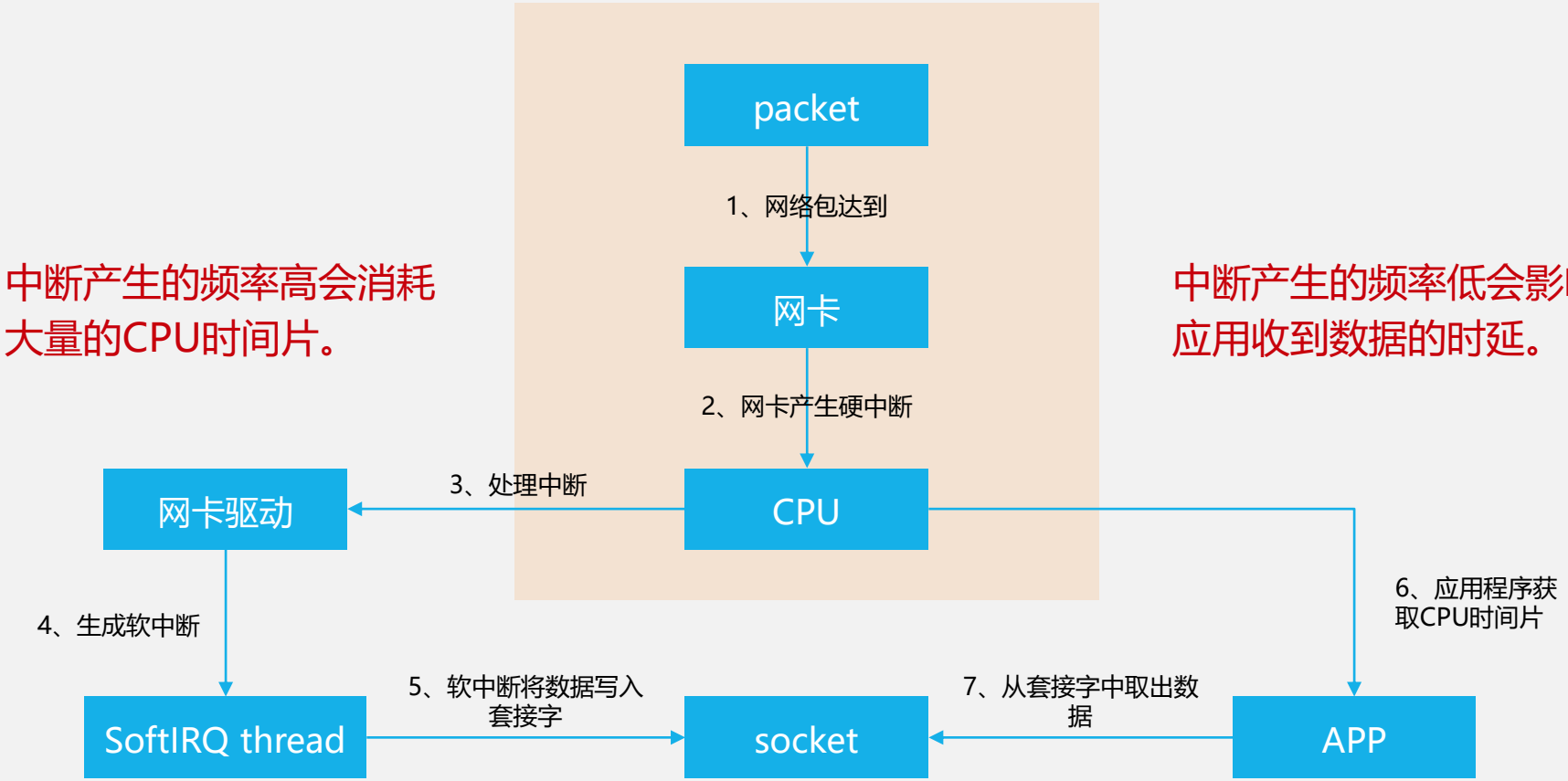
开启磁盘预取

适用于大数据读场景：Hibench测试spark，yarn-client模型，read_ahead_kb由128修改至4096，性能约提升10%。

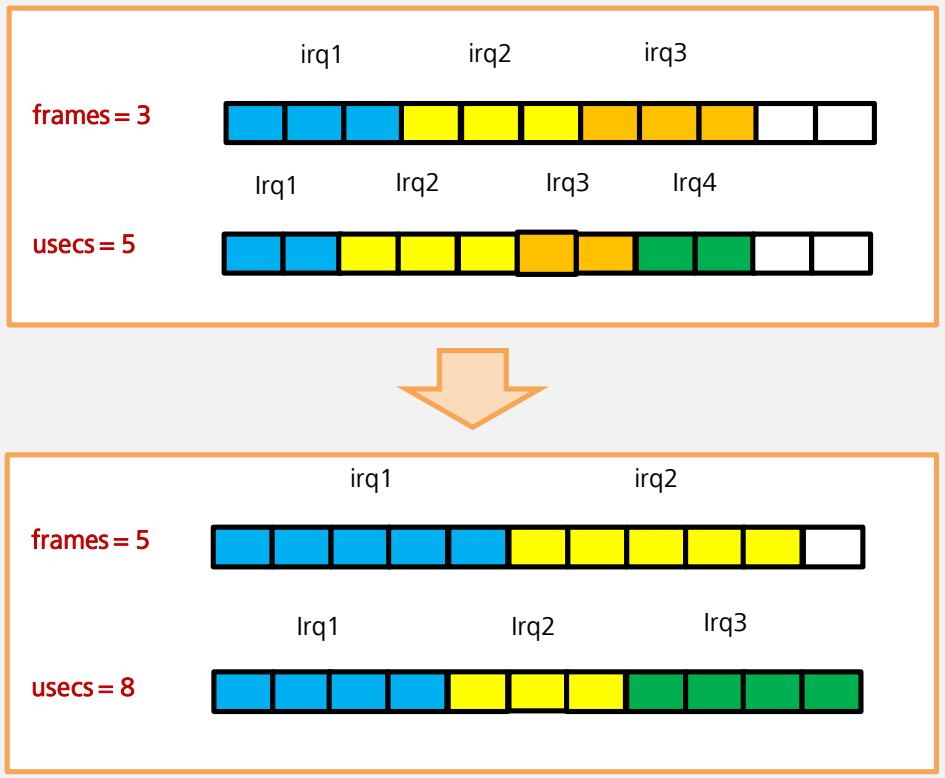
网卡中断产生频率会影响应用的吞吐和延迟

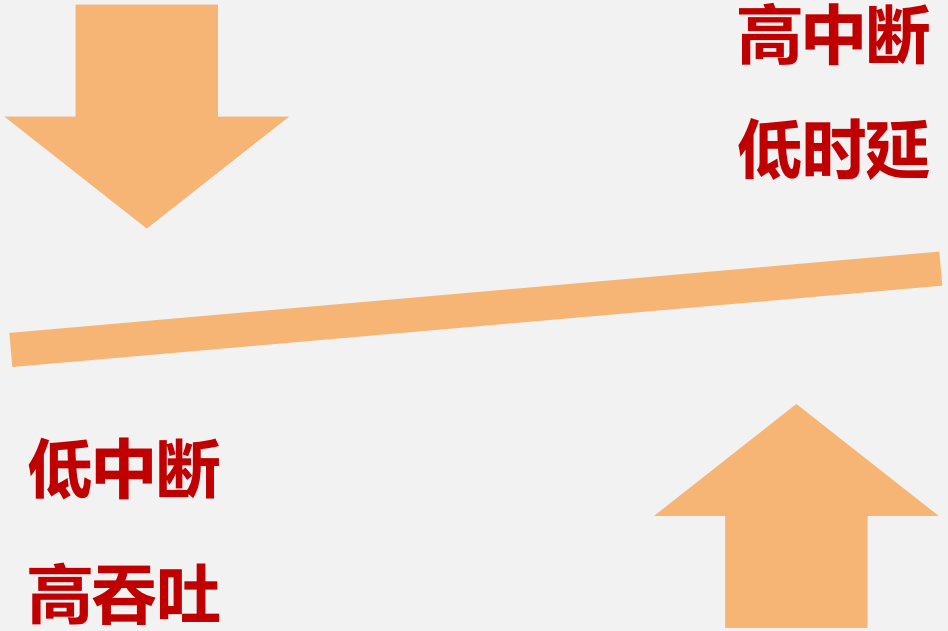
中断产生的频率高会消耗大量的CPU时间片。

中断产生的频率低会影响应用收到数据的时延。



调整网卡中断聚合，在低时延和高吞吐取平衡点





在数据库TPCC测试模型中降低网卡中断可以提升吞吐

修改前

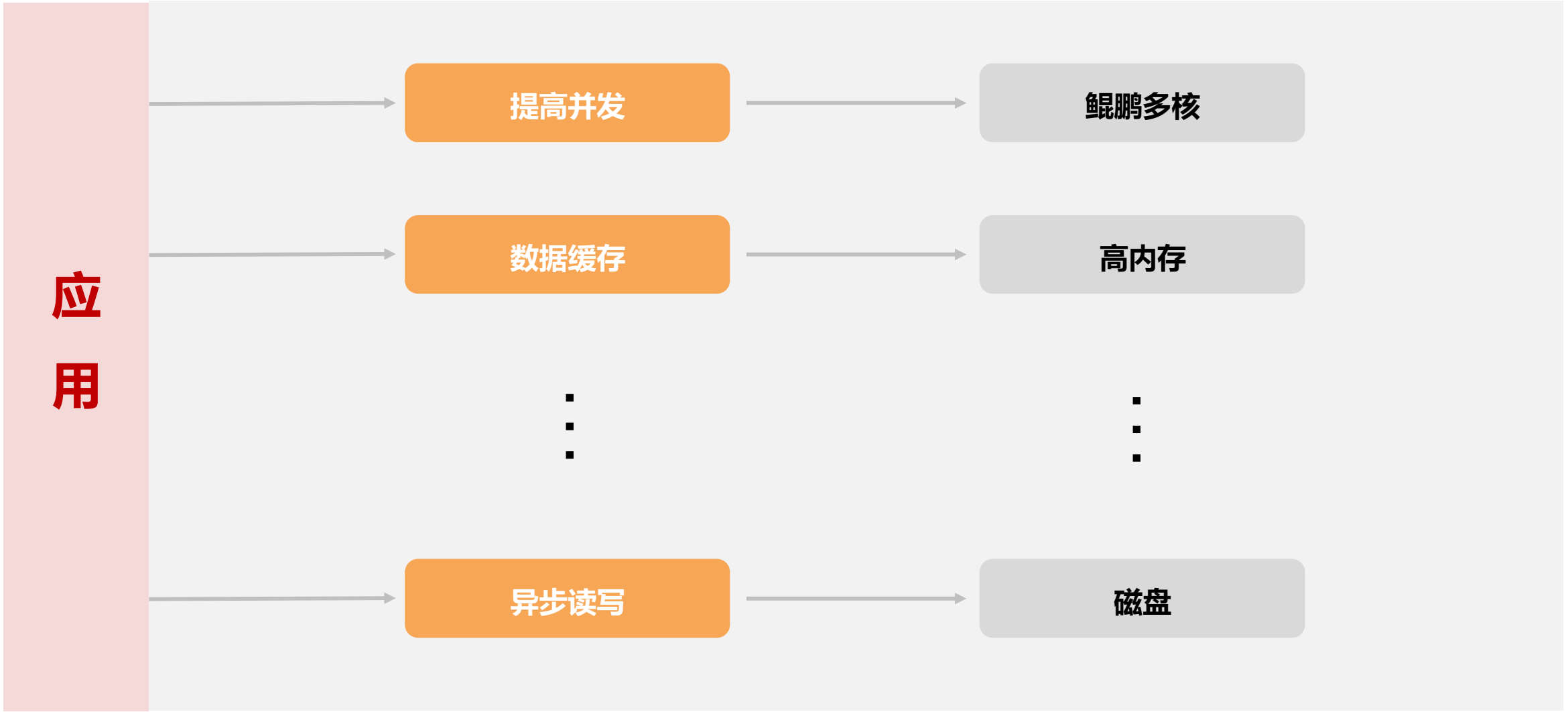
变量	数值
tx-frames	32
tx-usecs	16
rx-frames	32
rx-usecs	16
hi and si in cpu	15%
Total tpm	190w
Avg_rt	11.01ms

修改后

变量	数值
tx-frames	300
tx-usecs	200
rx-frames	300
rx-usecs	200
hi and si in cpu	10%
Total tpm	210w
Avg_rt	13.02ms

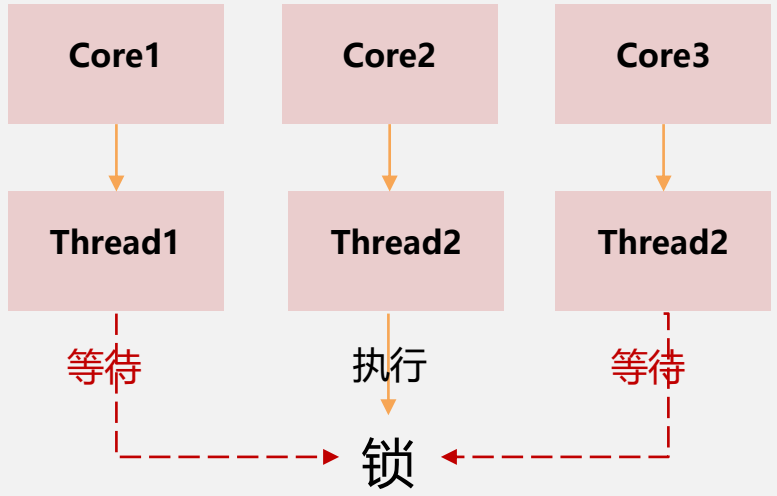
降低网卡中断频率，可以带来约10%的吞吐提升，但也会造增加20%左右的延迟

软件调优的本质是充分发挥硬件性能



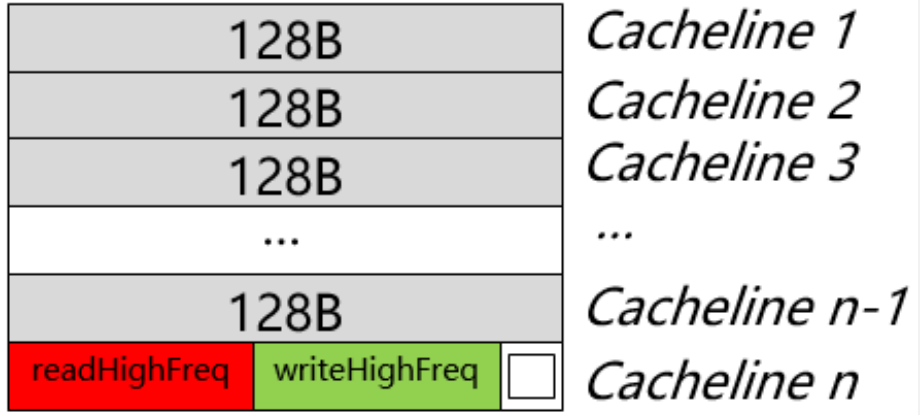
减少资源抢占，提升并行度，发挥多核性能优势

锁



- 无锁编程
- 大锁变小锁
- 高性能原子操作指令

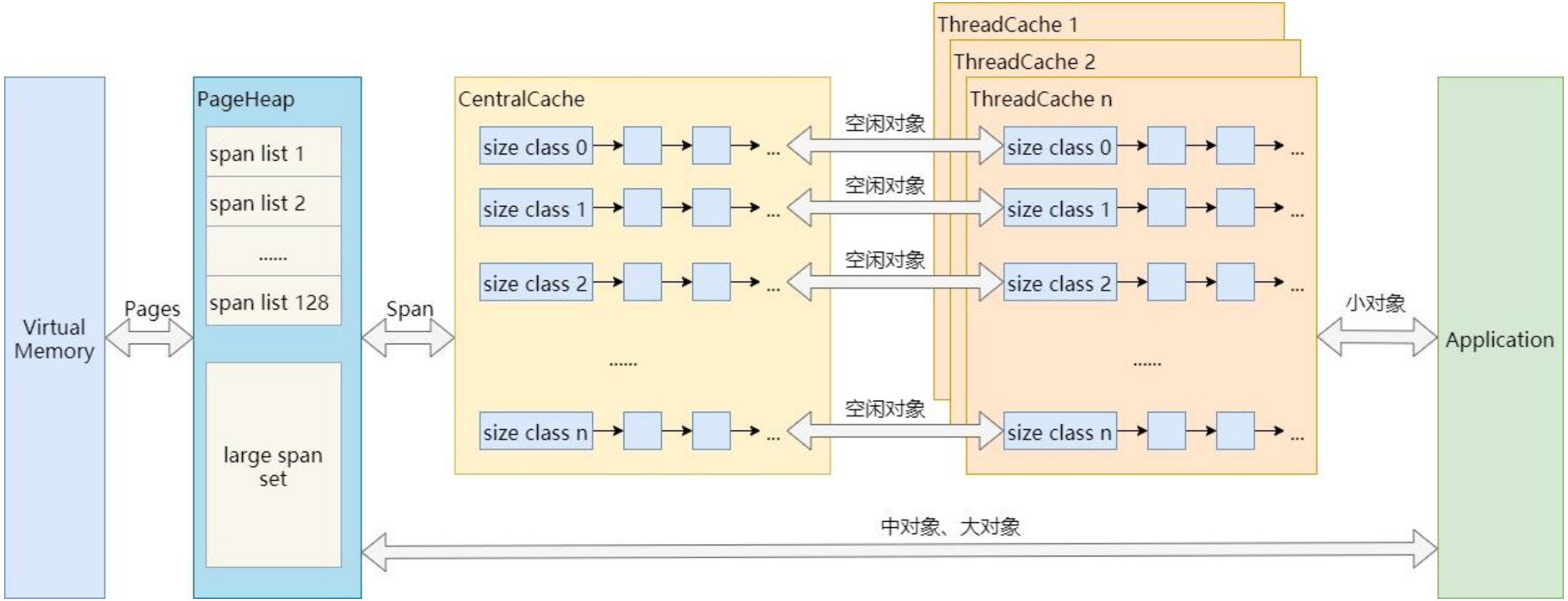
Cache



- 鲲鹏920的CacheLine大小为128字节
- 读和写频繁的变量分别放入不同Cacheline，避免伪共享

Tcmalloc通过减少内存分配中的锁以提升高并发下的性能

Tcmalloc使用线程缓存，尺寸小于256K的小内存申请均由ThreadCache进行分配；通过ThreadCache分配过程中不需要任何锁，可以极大的提高分配速度。



Mysql5.7.12内存对齐硬编码，导致伪共享

c

- issue path：https://github.com/mysql/mysql-server/pull/66/files
- cpu的L3 cacheline 获取方式可以通过读取系统配置文件获取，如在centos 7.6中的/sys/devices/system/cpu/cpu1/cache/index3/coherency_line_size文件。

4 storage/innobase/btr/btr0sea.cc

```
@@ -62,7 +62,7 @@ ulint         btr_search_n_hash_fail  = 0;
62  62    /** padding to prevent other memory update
63  63    hotspots from residing on the same memory
64  64    cache line as btr_search_latches */
65      - byte        btr_sea_pad1[64];
    65  + byte        btr_sea_pad1[CACHE_LINE_SIZE];
66  66
67  67    /** The latches protecting the adaptive search system: this latches protects the
68  68    (1) positions of records on those pages where a hash index has been built.
@@ -74,7 +74,7 @@ rw_lock_t**   btr_search_latches;
```

性能调优十板斧

CPU/内存	磁盘	网卡	应用
• 调整内存页大小 • CPU预取 • 修改线程调度策略	• 脏数据刷新 • 异步文件操作（libaio） • 文件系统参数	• 网卡多队列 • 开启网卡TSO • 开启网卡CSUM	• 优化编译选项 • 文件缓存机制 • 缓存执行结果 • NEON指令加速

鲲鹏调优十板斧链接：https://bbs.huaweicloud.com/blogs/126788

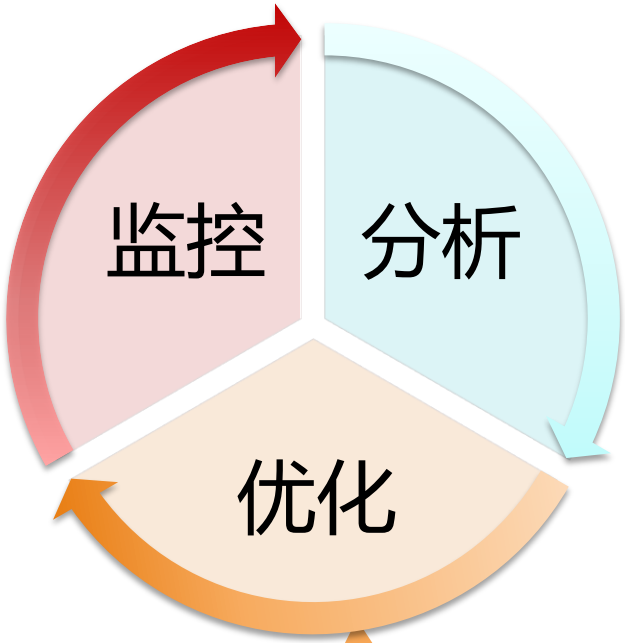
目录

基于硬件特性的性能调优方向

MariaDB性能调优案例

性能优化三步法

CPU：top、dstat
内存：numastat、free
磁盘：iostat、blktrace
网卡：sar、ethtool

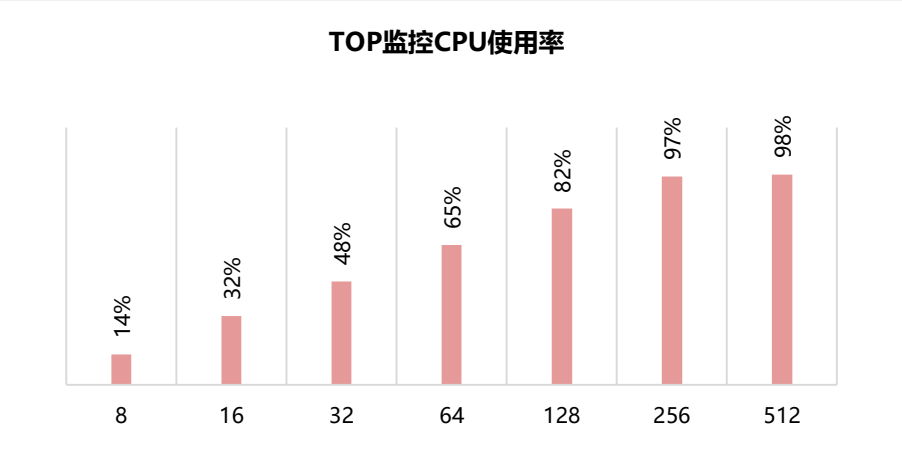


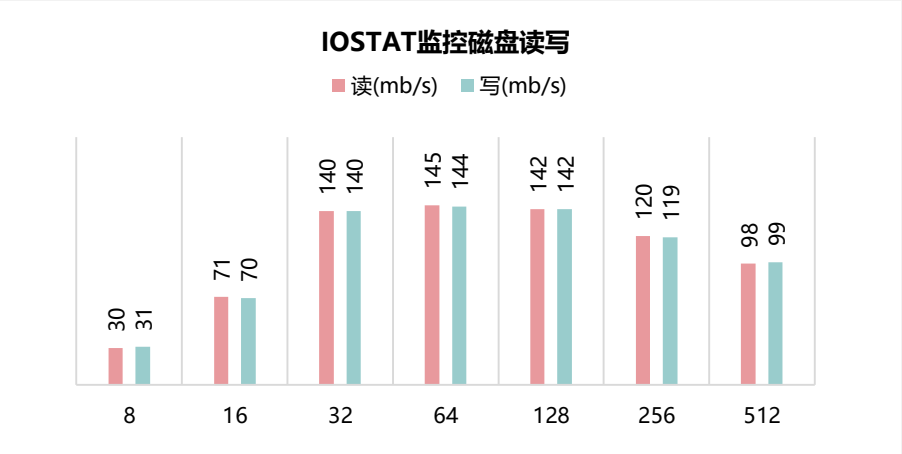
CPU：us、hi、si
内存：numa_hit、mem
磁盘：iowait、util%
网卡：txkB/s、tx_usecs

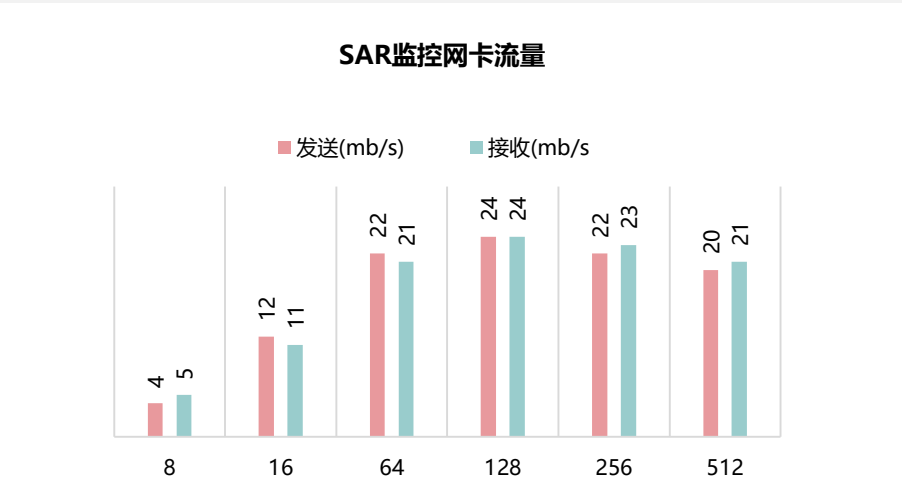
CPU：提高并发、线程绑核
内存：减少跨numa访问、大页内存
磁盘：I/O调度策略、异步I/O
网卡：中断聚合、网卡中断绑核

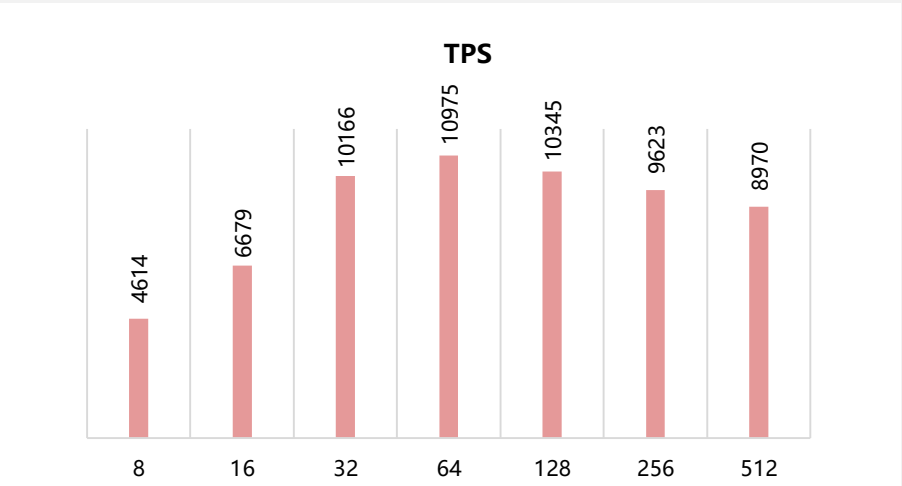
MariaDB性能调优——监控

测试模型sysbench压测Maria DB10.3.8数据库，OLTP模型1:1读写









CPU：使用率已经接近100%

磁盘：未达到性能瓶颈

网卡：未达到性能瓶颈

业务：随着并发线程数增加，性能不升反降。

MariaDB性能调优——分析

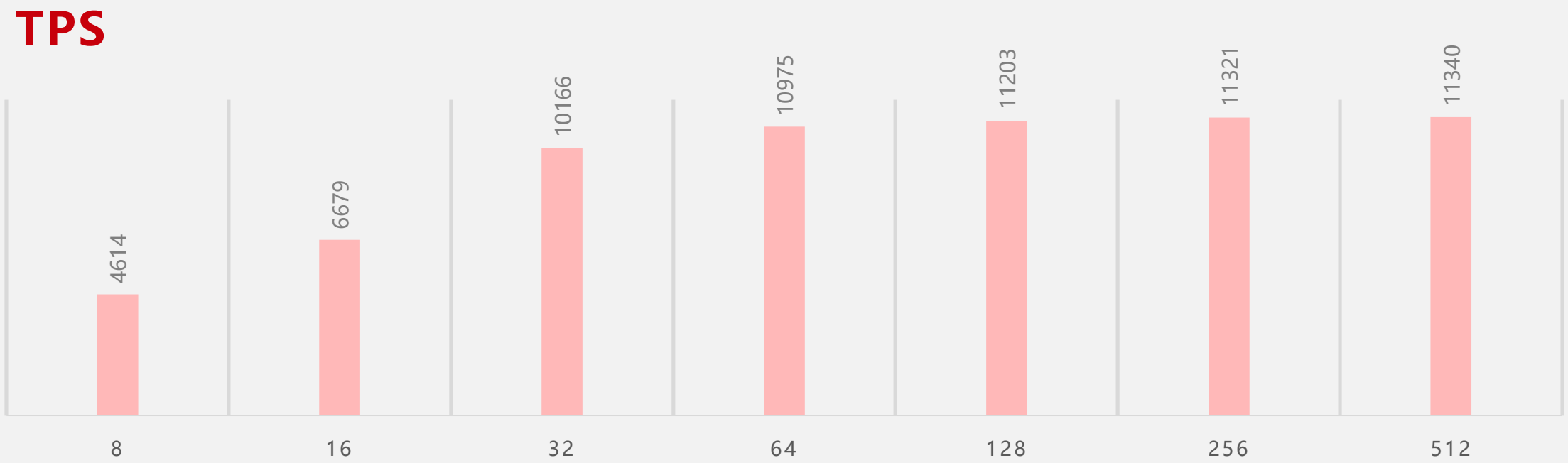
- 当CPU是业务的性能瓶颈时，可以通过分析进程热点函数来寻找优化空间。
- 对于本测试用例中的热点函数分析如下：随着并发线程数的增多，CPU的时间片集中在了锁的争抢中，这部分无用功造成了CPU资源的浪费。

```
- start_thread
  - 97.40% handle_one_connection
    - 97.30% do_handle_one_connection
      - 97.21% do_command
        - 85.92% dispatch_command
          - 70.93% mysql_parse
            - 63.41% mysql_execute_command
              - 39.60% execute_sqlcom_select
                - 37.49% handle_select
                  - 37.43% mysql_select
                    - 34.88% JOIN::optimize
                      - 33.87% JOIN::optimize_inner
                        - 30.69% join_read_const_table
                          - 30.25% join_read_const
                            - 30.22% handler::ha_index_read_idx_map
                              - 30.15% handler::index_read_idx_map
                                - 29.71% ha_innobase::index_read
                                  - 29.48% row_search_mvcc
                                      17.63% ut_delay
                                    + 10.11% ReadView::open
                                    + 1.01% btr_cur_search_to_nth_level_func
                          0.96% JOIN::optimize_stage2
```

热点函数采集：perf record –a –g –p 进程ID

采集内容查看：perf report

MariaDB性能调优——优化



- innodb_thread_concurrency：控制并发线程数，默认值0表示，不限制并发数。
- innodb_sync_spin_loops：减少原子操作轮休次数
- innodb_spin_wait_delay：增加原子操作轮休间隔时间

本章总结

- 1、CPU/内存、磁盘、网卡、应用，是我们性能调优的四个主要方向。
- 2、采集性能指标、分析性能瓶颈、优化相关参数代码，是调优的基本思路。
- 3、充分利用硬件资源才能发挥软件的最优性能。
- 4、时延、吞吐、并发需要寻找一个均衡点。


谢 谢
www.huawei.com