OpenGauss实验文档

一、实验介绍

多路服务器在各种服务中应用广泛，特别是金融行业通过多路服务器进行性能提升，支持更加大的并发交易场景。数据库随着从单机走向分布式数据库，需要充分利用底层硬件的多核能力、发挥CPU算力。本实验针对openGauss的NUMA场景进行性能调优，在调优的过程中，可以掌握一条SQL在数据库中的全生命周期执行过程、学习操作系统和数据库性能优化手段、NUMA-aware技术在数据库产品中使用，为进一步研究打下基础。

二、实验内容

环境搭建

安装环境

本实验每组为每组提供两台aarch64架构的多NUMA服务器，已分别预装**openEuler20.03LTS**和**centos7**系统，分组情况以及每组的账号密码可从如下链接获得：22秋高级云操作系统分组。

yum配置环境时可直接在root下进行，普通用户使用sudo -E yum install xxx。

安装OpenGauss

openGauss安装请参照华为官方文档：opengauss安装教程。选择**3.1.0**版本极简版或企业版安装。我们推荐功能更齐全，自定义程度更高的**企业版**。文档中的openGauss安装包下载链接失效，可用地址为https://opengauss.org/zh/download。请在openEuler系统上选择openGauss单节点安装。

Part1 物理性能测试

（1）fio测试

工具介绍

在Lab2 Cloud Storage中，大家已经使用过fio工具测试磁盘性能，在本次实验中我们依然使用它来做基准测试，以获得磁盘I/O的极限性能。

安装步骤

- **下载地址**

下载链接：https://github.com/axboe/fio

使用文档：https://fio.readthedocs.io/en/latest/index.html

- **编译安装**

```
./configure
make
make install
```

测试

- **测试场景和参数建议**

在这部分中，你需要对硬盘分别在read、write、randread和randwrite场景下的极限I/O性能进行测试，因此将iodepth设置成较大值，ioengine设置成异步的io_uring或者libaio以提升性能表现。SSD的读写速度较快，因此size和runtime建议不要太小，防止结果波动过大。

我们推荐你通过rwmixwrite，rwmixread等参数控制，进行一些读写混合的场景测试，以及适当更改blockSize（参数bs）大小查看读写速度的变化。

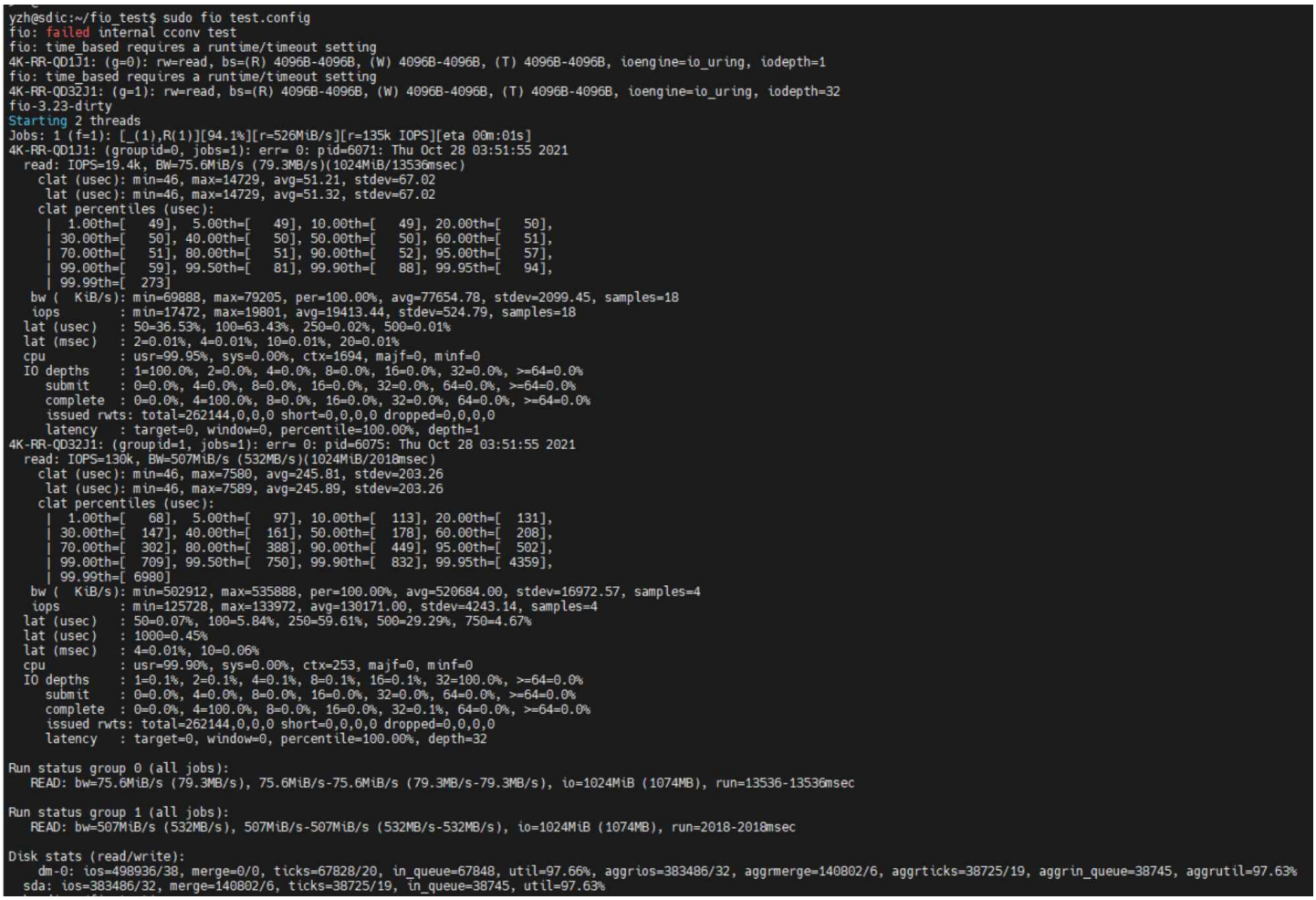
- **注意事项**

1. fio测试会对系统的硬盘进行读写，会存在破坏数据的风险，因此在生产环境上要谨慎使用，不要对生产环境的硬盘进行裸盘读写测试。如果需要进行测试，最好将硬盘挂载到文件系统上，创建一个文件，然后对指定的文件进行读写。
2. 建议多测几次，剔除不正常数据。
3. 测试时不要运行其他应用，防止CPU资源不足影响测试结果。

- **测试结果**

在得出的结果中需要关注**IOPS**

运行结果示例：



- **配置文件（供参考）**

运行指令：

```
fio fio_bench.config
```

fio_bench.config

```
[global]
name=NVMe-benchmark
filename=/dev/nvme1n1      #硬盘路径，请看注意事项
ioengine=io_uring
sqthread_poll=1
direct=1
numjobs=1
runtime=180
ramp_time=60
thread=1
time_based=1
randrepeat=0
```

```
norandommap=1
group_reporting=1
size=100G                #size建议不要过小，数据会不准确
stonewall

[4K-R]
bs=4k
rw=read
iodepth=32
stonewall

[4K-W]
bs=4k
rw=write
iodepth=32
stonewall

[4K-RR]
bs=4k
rw=randread
iodepth=32
stonewall

[4K-RW]
bs=4k
rw=randwrite
iodepth=32
stonewall
```

具体要求

你在实验报告中需要提交fio测试时的参数截图和运行结果的截图。

（2）其他测试

以上fio测试仅对平台SSD的IO性能进行了测试，我们推荐你对实验平台性能进行更全面的测试，如网络性能、跨NUMA节点内存访问性能等，请大家自行设计实验。

具体要求

你在实验报告中需要写明：

- 实验设计思路

- 测试工具
- 测试结果截图及分析

Part2 基础环境下的测试（baseline）

（1）跑TPC-C得出tpmC值

对于openGauss的TPCC性能调优，华为在官方文档中给出了基于arm平台调优的具体步骤和细节：TPCC性能调优指导 。你可以参考其中环境的配置，请按照自己的实验思路自行调整。

数据库和服务端绑核中，gs_guc设置的参数含义请查看GUC参数说明。

使用BenchmarkSQL工具进行TPCC测试。BenchmarkSQL是一款经典的开源数据库测试工具，内嵌了TPCC测试脚本，可以对EnterpriseDB、PostgreSQL、MySQL、Oracle以及SQL Server等数据库直接进行测试。

在客户端安装BenchmarkSQL测试工具并部署环境，步骤如下：

a. 下载BenchmarkSQL标准测试工具benchmarksql-5.0。

b. 将目录lib/postgresql下面的*.jar 替换为openGauss适配的jar包。

```
$ pwd
/your path/benchmarksql-5.0/lib/postgres
$ ls
postgresql.jar #openGauss jdbc驱动。
postgresql-9.3-1102.jdbc41.jar.bak    # 自带jar备份。
```

openGauss适配的JDBC版本包获取路径为openGauss-x.x.x-JDBC.tar.gz，请下载openeuler_aarch64版本。

c. 进入benchmarksql-5.0根目录，输入ant命令进行编译。

```
$ cd /your path/benchmarksql-5.0/
$ ant
```

d. 创建benchmarkSQL配置文件，使用benchmarkSQL前需要配置数据库相关的信息，包括数据库账号、密码、端口、数据库名称。

```
$ cd /your path/benchmarksql-5.0/run
```

```
$ cp props.pg props.opengauss.1000w
$ vim props.opengauss.1000w
```

从props.pg拷贝一份配置文件并按如下修改。

```
db=postgres
driver=org.postgresql.Driver
// 修改连接字符串，包含IP、端口号、数据库，其中8.92.4.238为数据库服务端的千兆网卡IP。
conn=jdbc:postgresql://8.92.4.238:21579/tpcc1000?prepareThreshold=1&batchMode=on&fetchsize=10
// 设置数据库登录用户和密码。
user=bot
password=Gaussdba@Mpp

warehouses=1000 // 数量可自行调整
loadWorkers=200

// 设置最大并发数量，跟服务端最大work数对应。
terminals=812
//要为每个终端运行指定事务--runMins必须等于零
runTxnsPerTerminal=0
//要运行指定的分钟 - runTxnsPerTerminal必须等于零
runMins=5
//每分钟总事务数
limitTxnsPerMin=0


//在4.x兼容模式下运行时，设置为True。
//设置为false以均匀使用整个配置的数据库。
terminalWarehouseFixed=false

//以下五个值相加之和为100。
//45、43、4、4和4的默认百分比与TPC-C规范匹配。
newOrderWeight=45
paymentWeight=43
orderStatusWeight=4
deliveryWeight=4
stockLevelWeight=4

//创建文件夹以收集详细结果数据。
//通过注释取消此内容。
resultDirectory=my_result_%tY-%tm-%td_%tH%tM%tS
osCollectorScript=./misc/os_collector_linux.py
osCollectorInterval=1
//收集OS负载信息。
```

```
40 osCollectorSSHAddr=osuer@10.44.133.78
41 osCollectorDevices=net_enp3s0 blk_nvme0n1 blk_nvme1n1 blk_nvme2n1 blk_nvme3n
   1
```

数据库名称和端口、ip、用户名等需要按照数据库服务端配置来设置，其余参数配置，比如\warehouse、terminals等等，需要你根据环境的实际情况和实验需要进行修改。

e. 导入数据前的准备。

需要替换benchmarkSQL中的文件，路径为benchmarksql-5.0/run/sql.common/tableCreates.sql。详细操作可以参考openGauss数据库性能调优。

6. 导入数据。

首先，连接数据库，创建数据库和用户数据。

```
1 create user bot identified by 'Gaussdba@Mpp' profile default;
2 alter user bot sysadmin;
3 create database tpcc1000 encoding 'UTF8' template=template0 owner bot;
```

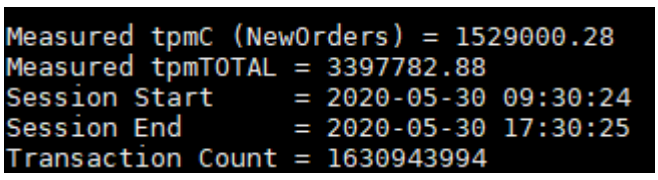
然后，执行脚本导入数据。

```
1 ./runDatabaseBuild.sh props.opengauss.1000w
```

g. 运行TPCC测试程序。

```
1 numactl -C 0-19,32-51,64-83,96-115 ./runBenchmark.sh props.opengauss.1000w
```

运行TPCC测试程序后，可以得到类似于下图的测试结果。



具体要求

在搭建了openGauss数据库服务的物理机上，使用BenchmarkSQL进行TPCC性能测试。请提供顺利运行测试后，程序运行的截图和结果。该部分作为环境搭建部分的评分指标之一。

（2）单核多核测试结果分析（扩展性）

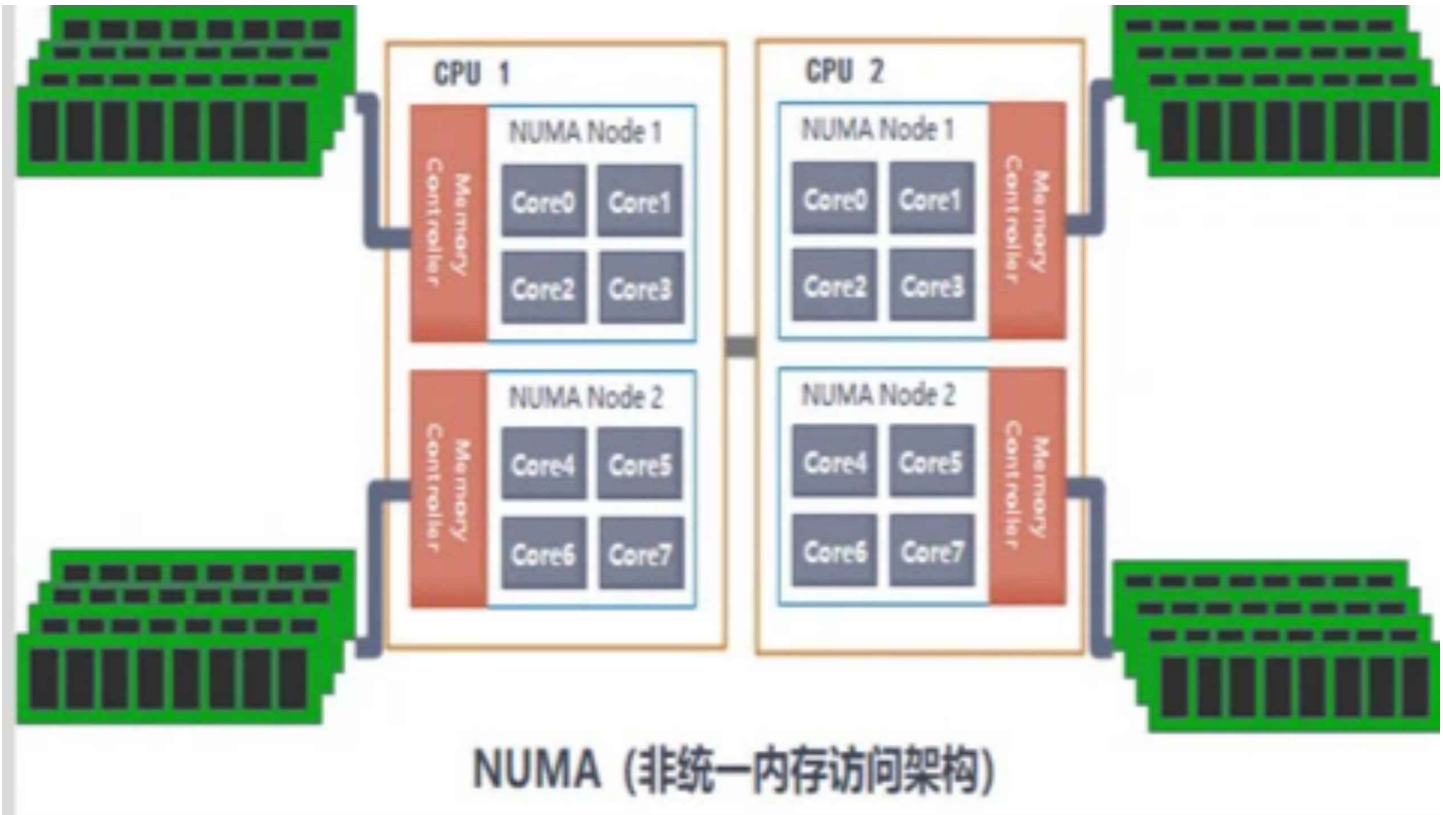
- 请你根据使用的CPU核数从少到多进行TPC-C测试（测试组数越多更容易得到完整趋势），画出性能变化趋势图（折线图或拟合曲线图）。请注意你测试时分配给openGauss使用的核应当位于不同的numa节点上。
- 结合性能变化趋势图，分析在numa架构的情况下性能随核心数增加的可扩展性。根据Part1测试的极限性能结果，分析硬盘I/O速度等硬件性能是否为你当前实验测试的瓶颈所在。

Part3 numa-aware调优

背景介绍

传统数据库基于多进程模型，在NUMA架构下一般使用进程绑核的方式,或者通过对操作系统NUMA的配置进行控制修改内存分配方式来进行性能提升。Linux内核本身已经适配了NUMA特性，对应用提供了一些API用于更细粒度话的NUMA控制，即NUMA-aware技术，应用可以按照特性对内存和CPU使用特点进行NUMA优化。OLTP类型数据库因为其低时延高并发的业务特定，更需要在多核下进行NUMA-aware的业务造或数据结构优化以达到最优性能。

OpenGauss数据库基于PostgreSQL，是基于多进程模型设计的。对于每个新加入的客户端，OpenGauss都会fork一个新worker进程对客户端请求进行处理。各worker进程的内存并不共享，但为提供主进程和worker进程间通信、锁共享、维持日志写入等功能，主进程和worker进程间会通过共享内存来维护一些数据结构。这些共享的数据结构中有些在数据库运行过程中会被频繁访问，在NUMA场景下，若内存的分配位置不进行NUMA-aware的优化，则会导致**跨核的内存访问**，引发**可观的访问延迟**，严重影响整体系统性能。



numa-aware调优思路

请你针对OpenGauss中**PGPROC**这一共享数据结构进行NUMA-aware优化。

PGPROC是OpenGauss中一个重要的结构体，在运行时维护了worker进程的线程ID、线程当前状态(等待的信号量)、会话ID、数据库用户OID等信息。该对象的所有实例均由主线程维护，在worker进程内全局唯一，**存储在共享内存中**。对于每个客户端连接，OpenGauss都会fork一个新的worker进程，并由主进程分配一个PGPROC结构体给新的worker进程。在worker进程执行涉及事务的相关操作时（如新建事务、事务提交、获取事务快照等），worker进程都需要访问/更新PGPROC结构体内数据，在有海量事务执行的OLTP场景下会被频繁访问。

```
struct PGPROC {
    /* proc->links MUST BE FIRST IN STRUCT (see ProcSleep,ProcWakeup,etc)
    SHM_QUEUE links; /* list link if process is in a list */

    PGSemaphoreData sem; /* ONE semaphore to sleep on */
    int waitStatus;      /* STATUS_WAITING, STATUS_OK or STATUS_ERROR */
                         /* STATUS_BLOCKING for update block session */

    Latch procLatch; /* generic latch for process */

    LocalTransactionId lxid; /* local id of top-level transaction current
                              * being executed by this proc, if running;
                              * else InvalidLocalTransactionId */
    ThreadId pid;            /* Backend's process ID; 0 if prepared xact
    /*
     * session id in mySessionMemoryEntry
     * stream works share SessionMemoryEntry with their parent sessions,
     * so sessMemorySessionid is their parent's as well.
     */
    ThreadId sessMemorySessionid;
    uint64 sessionid; /* if not zero, session id in thread pool*/
    GlobalSessionId globalSessionId;
```

若worker进程在访问、更新PGPROC结构体内数据时，触发了NUMA远程访问，将严重影响性能。请你通过分析源码的方式，学习PGPROC在OpenGauss数据库内部的使用方式，并尝试通过**内存分区、线程绑定等NUMA-aware的方法**，在主进程中优化PGPROC结构体的内存分配与存储结构，以此提高OpenGauss的OLTP性能。

具体要求

你的工作应该包含三部分：

- 分析PGPROC数据结构及其相关操作函数在TPC-C负载下的执行特点。
  - 请你学习主进程中PGPROC类型对象的分配方式，并分析其在内存分布上有什么特点，在NUMA场景下可能会导致怎样的性能问题。

    - 请你学习PGPROC类型对象相关操作函数的功能，并在TPC-C负载下，通过**火焰图**分析PGPROC相关的操作函数执行的CPU时间占比，指出相关操作中性能瓶颈所在。
- 设计并实现NUMA-aware的PGPROC结构体分配方式。请在报告中描述清楚你的优化思路与方式、使用了哪些NUMA相关API、修改了哪些函数。openGuass中已经自带了numa优化的实现，可在源代码中通过宏定义ifdef __USE_NUMA找到所有numa相关函数定义及调用的地方。你可以在该版本上继续优化，也可以另起炉灶设计新的算法。重新编译openGauss请参照官方文档编译指导书，注意在**configure**时是否要添加**-D_USE_NUMA**参数。
- 对经过优化后的系统进行TPCC测试，与不使用numa优化（包括自带的）的系统进行比较。

参考

- openGauss中NUMA相关功能与参数设置
    - src\bin\scripts\bind_net_irq.sh ： 网卡按numa node绑核脚本
    - numa_distribute_mode: NUMA GUC参数控制,开启则表示将部分共享数据和线程分布到不同的NUMA节点下，减少远端访存次数，提高性能
- PGPROC相关实现原理
    - 源代码位置：
        - https://github.com/postgres/postgres/blob/master/src/include/storage/proc.h
        - https://github.com/postgres/postgres/blob/master/src/include/storage/procarray.h
        - https://github.com/opengauss-mirror/openGauss-server/blob/master/src/gausskernel/storage/lmgr/proc.cpp
- NUMA API
    - openGauss已经具备numa能力，引入了numa lib作为三方件，代码中直接应用numa.h头文件即可，numa api可以参考linux： https://linux.die.net/man/3/numa
    - 常用的API如下三个:
        - numa_run_on_node: 将当前任务及子任务运行在指定的Node
        - numa_set_localalloc: 将调用者线程的内存分配策略设置为本地分配
        - numa_alloc_onnode: 在指定的NUMA Node上申请内存
- NUMA-aware相关研究（仅供参考）
    - Enabling NUMA-aware Main Memory Spatial Join Processing: An Experimental Study
    - Compact NUMA-aware Locks
    - DeLoc: A Locality and Memory-Congestion-Aware Task Mapping Method for Modern NUMA Systems

- NUMA-aware CPU core allocation in cooperating dynamic applications
- NUMA Awareness: Improving Thread and Memory Management
- Scalable NUMA-aware Blocking Synchronization Primitives

三、评分标准

- 材料提交要求
  - 每个小组只需组长提交实验材料
  - 提交内容需要包括一份实验报告"组号_report.{docx/doc/pdf}"、一个包含截图的文件夹"组号_data"、和一个包含修改过的代码文件的文件夹（加上README文件简要描述代码修改的位置）。请将它们放在一起以压缩包形式提交。
  - 需要包含所有fio测试、tpc-c测试及其他测试的配置和测试结果的截图，若使用配置文件或脚本可以文件形式提交
- 评分细则（满分100分）
  - Part1 （基础）：20分
    - fio测试结果截图：10分
    - 其他测试：10分
  - Part2（基础）：35分
    - TPC-C测试结果截图：10分
    - 单核和多核下TPC-C性能趋势图：15分
    - 性能分析：10分
  - Part3（拓展）：总分45分
    - 分析PGPROC数据结构及其相关操作函数在TPC-C负载下的执行特点
      - 分析PGPROC结构在内存分布上的特点以及NUMA场景下的性能问题：5分
      - 在TPC-C负载下，画出数据库运行时的火焰图：5分
      - 尝试分析性能瓶颈所在：5分
    - 根据实验平台调整参数或修改代码来提升数据库性能，并结合资料阐述你的优化思路：15分
    - 添加numa-aware机制后的优化效果（和不使用numa优化的系统相比）：15分
- 禁止抄袭他人的实验数据、实验内容。助教们会对实验测试结果的真实性进行普查，请确保能复现实验结果
- 报告内除实验主体内容外，还需要小组成员和贡献度情况（阐述明确工作内容）