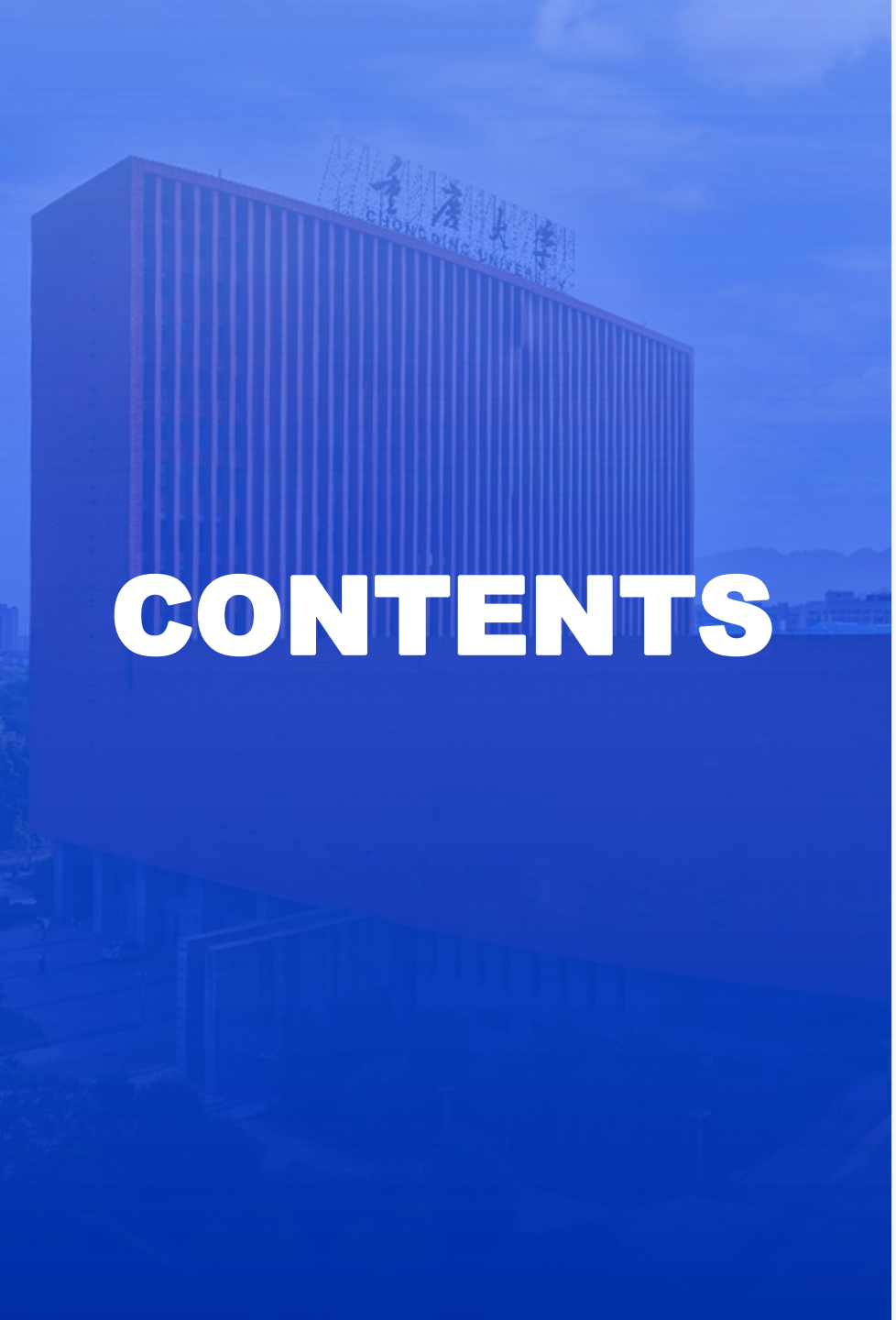
A-Tune 介绍

重庆大学 - 操作系统翻转课堂

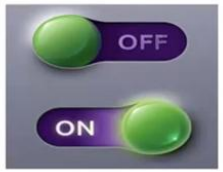


CONTENTS

背景

背景

- 操作系统中参数数量多，这些参数之间存在复杂的相关性
- 这些可调节的参数控制着运行的各个方面

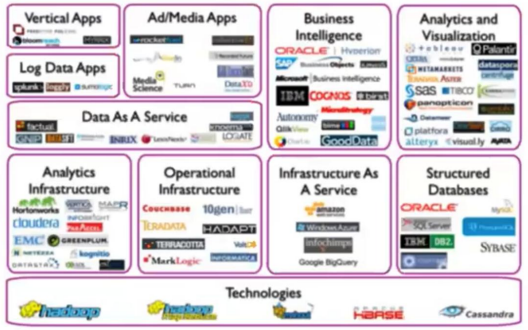
							
CPU • LLC Cache 预取 • 内存交织 • …	ACC • 加密/解密 • 证书 • …	NIC • 中断亲和性 • 功能卸载 • …	OS • 任务调度 • NUMA均衡 • 亲和性 • 锁 • …	编译器 • FDO • 并行 • …	框架 • 数据库 cache size • memcpy size	应用 • 应用负载 • 线程数 • 用户定义 • …	其它 • 7000+可调节的系统对象

➢ 参数数量多，且参数间存在依赖关系

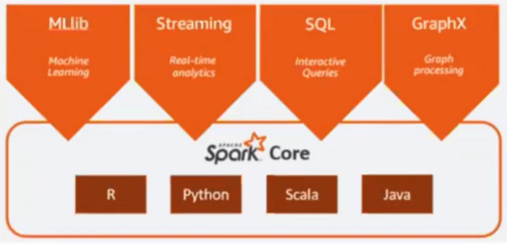
Symbol	Parameter	Groups
N_m	mapred.map.tasks	Map input split
N_r	mapred.reduce.tasks	Reduce output
s_{min}	mapred.min.split.size	Map input split
s_{max}	mapred.max.split.size	Map input split
B_m	io.sort.mb	Map output buffer
r_{rec}	io.sort.record.percent	Map output buffer
c	mapred.compress.map.output	Map output compr.
N_{copy}	mapred.reduce.parallel.copies	Reduce copy
N_{sf}	io.sort.factor	Reduce input
B_r	mapred.job.reduce.input.buffer.percent	Reduce input
S_{OB}	dfs.block.size	Reduce output
$N_{ma}(i)$	mapred.tasktracker.map.tasks.maximum	Set by HAC
$N_{ra}(i)$	mapred.tasktracker.reduce.tasks.maximum	Set by HAC

Hadoop中有超过190个参数

➢ 上层应用系统种类多



➢ 每个应用的负载也复杂多样



背景

OpenEuler

人工调优

- 调优时间长
- 依赖人工经验知识
- 高昂的人工成本
- 丰富的调优经验

智能调优

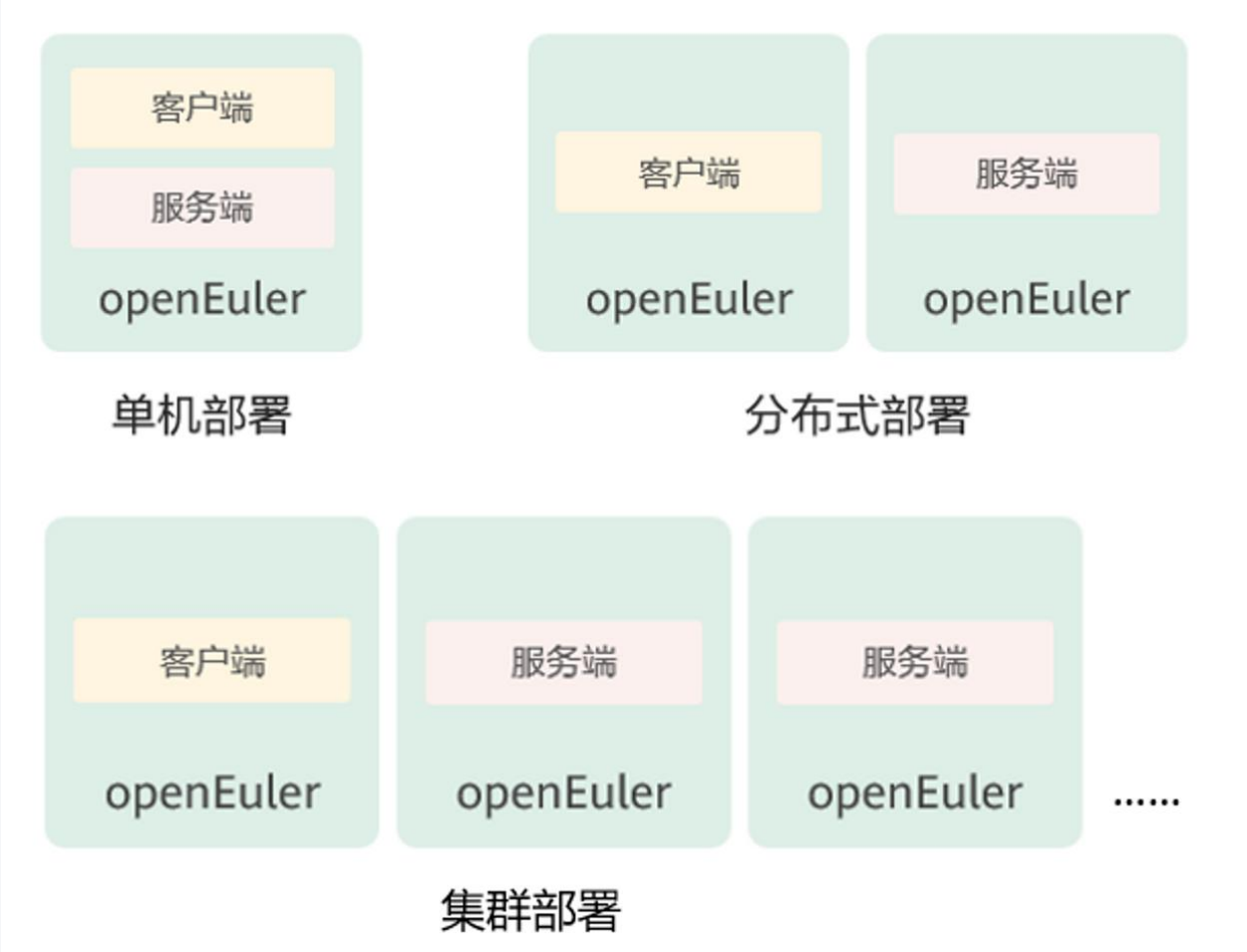
- 时间短
- 无需人工经验知识
- 仅需要机器成本
- 难以积累经验

目标：针对给定的应用负载，利用机器学习技术找到最优的参数使应用获得最佳性能

A-Tune架构

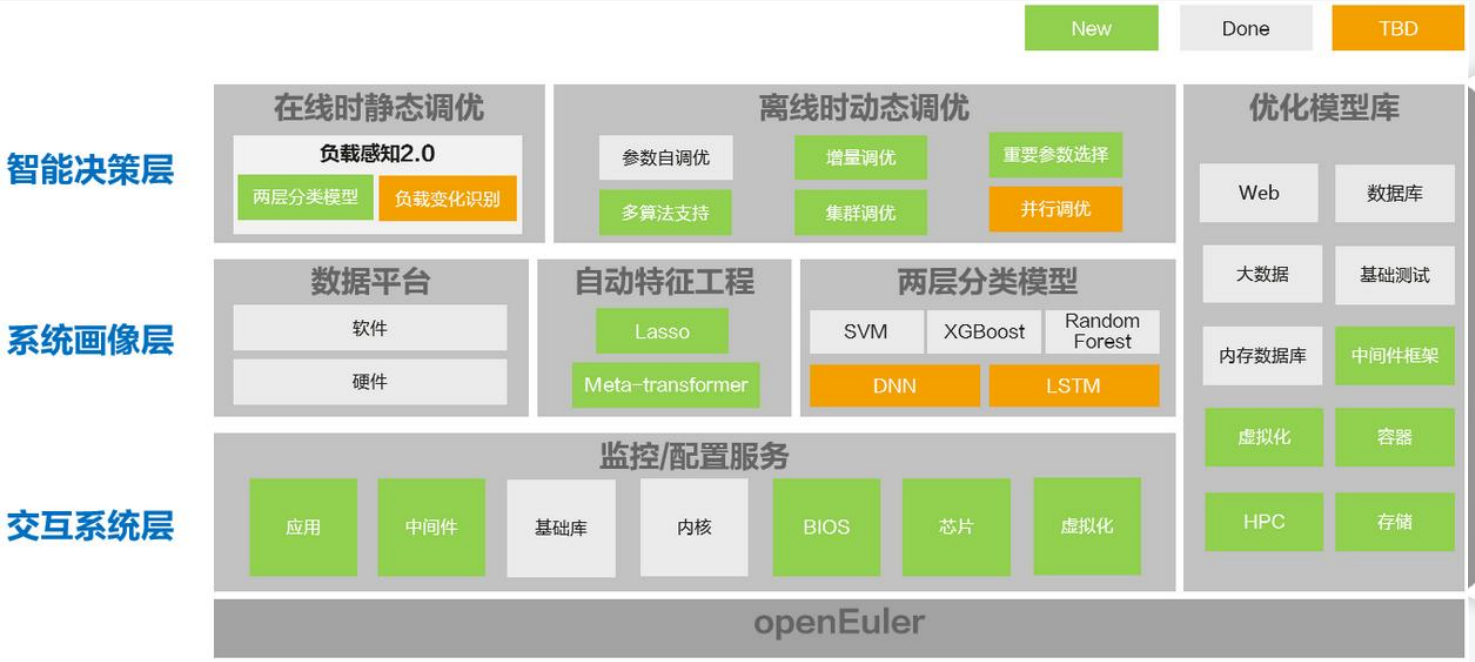
A-Tune整体架构

A-Tune整体上是一个C/S架构，以便于进行分布式和集群部署



A-Tune核心技术架构

A-Tune 目标：全面感知计算、存储、网络等应用负载资源，支持行业主流应用，实现全栈协同优化，实现大规模业务迁移，达到极简体验，高效调优的目的



智能决策层：在线静态调优与离线动态调优

系统画像层：数据平台、AI模型、训练推理模块

交互系统层：系统全栈监控和配置服务

优化模型库：各大类场景的经验模型库

在线静态调优

A-Tune 在线静态调优

场景：普通用户，应用需一直保持运行
思路：感知当前应用负载，通过分类模型，匹配到已知负载，直接输出经验参数

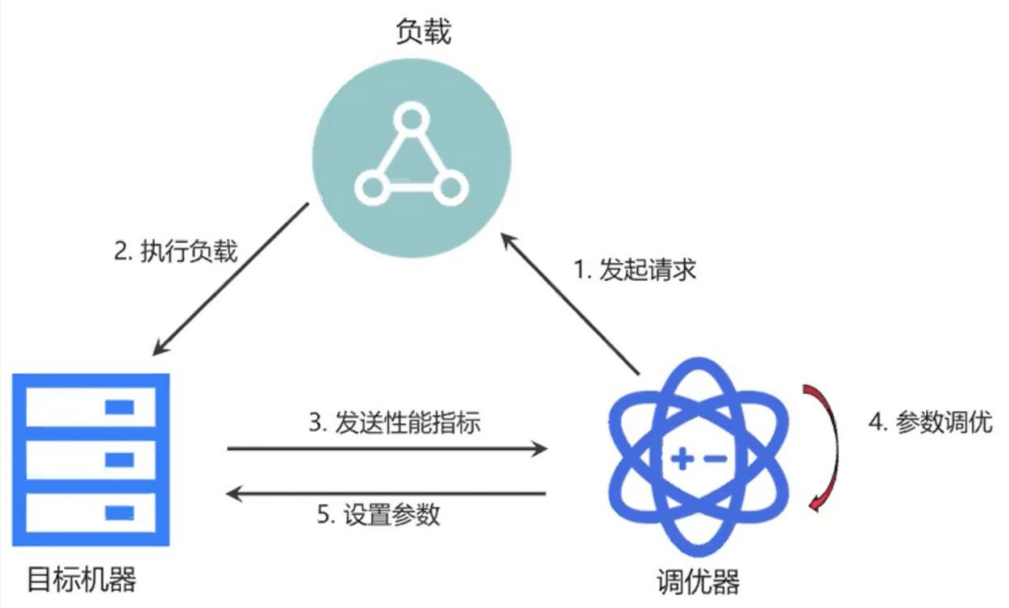
关键技术

- ✓ **重要特征分析**：自动选择重要特征，剔除冗余特征，实现精准用户画像
- ✓ **两层分类模型**：准确识别当前负载
- ✓ **负载变化感知**：主动识别应用负载的变化，实现自适应调优

离线动态调优

A-Tune 离线动态调优

场景：高阶用户，对性能要求高
思路：调优器给目标机器设置参数并得到反馈的性能指标，不断迭代，最终得到最优参数



关键技术

- ✓ **重要参数选择**：自动选择重要的调优参数，减少搜索空间，提升训练效率
- ✓ **调优算法构建**：用户可从适用场景、参数类型、性能要求等方面选择最优算法
- ✓ **知识库构建**：将当前负载特征和最优参数增加到知识库，提升后续调优效率

A-Tune使用流程

A-Tune 使用流程

1. 编写调优脚本

① **Benchmark 脚本：**

- 运行测试，并输出指标

② **Tuning Client yaml 脚本：**

- 提供从benchmark输出提取指标的方法
- 定义迭代轮数及算法

③ **Tuning Server yaml 脚本：**

- 定义待调节项，包括设置、获取方法

A-Tune 使用流程

编写调优脚本-Benchmark 脚本

- ✓ 启动生产者及消费者进程
- ✓ 10min之后 kill 生产者及消费者进程
- ✓ 根据生产者和消费者日志计算性能指标并输出

```
# run test
cd /opt/huawei/1.3.13.100/KOPsdv2dmq-demo-1.3.13.100/bin; sh start.sh
ssh 10.33.50.23 'source /etc/profile; cd /opt/huawei/1.3.13.100/liaofkafkaconsumer/bin; sh start.sh'

# wait for a period of time
totaltime=10
for i in $(seq 1 $totaltime); do
echo "test running progress: ($i / $totaltime) minutes..."
sleep 60
done

# kill test programs
echo 'kill producer...'
pid=`ps aux | grep dmq-demo | grep -v grep | awk '{print $2}' | head -1`
[ "$pid" == "" ] && echo 'producer is not running!'
[ "$pid" != "" ] && echo "producer pid is $pid" && kill -9 $pid
echo 'kill producer done'
ssh 10.33.50.23 'sh /home/A-Tune/kill.sh' 2>/dev/null

# extract test result
echo "producer_tps: `cat /opt/huawei/1.3.13.100/KOPsdv2dmq-demo-1.3.13.100/logs/nohup.out | grep 'TAG: producer' |
echo "delay: `cat /opt/huawei/1.3.13.100/KOPsdv2dmq-demo-1.3.13.100/logs/nohup.out | grep 'TAG: producer' | tail
ssh 10.33.50.23 'sh /home/A-Tune/get-result.sh' 2>/dev/null
```

A-Tune 使用流程

编写调优脚本-Tuning Client yaml 脚本

- ✓ 指定benchmark脚本
- ✓ 从benchmark输出中提取相关性能指标
- ✓ 设置各个指标的调优权重
- ✓ 迭代轮数及调优算法

```
benchmark : "sh /root/A-Tune-master/examples/tuning/kafka/kafka_bench.sh"
evaluations :
  -
    name: "producer_tps"
    info:
        get: "echo '$out' | grep 'producer_tps:' | awk '{print $2}'"
        type: "negative"
        weight: 25
  -
    name: "delay"
    info:
        get: "echo '$out' | grep 'delay:' | awk '{print $2}'"
        type: "positive"
        weight: 25
  -
    name: "comsumer_tps"
    info:
        get: "echo '$out' | grep 'consumer_tps:' | awk '{print $2}'"
        type: "negative"
        weight: 25
  -
    name: "consumer_min_tps"
    info:
        get: "echo '$out' | grep 'consumer_min_tps:' | awk '{print $2}'"
        type: "negative"
        weight: 25
```

A-Tune 使用流程

编写调优脚本-Tuning Server yaml 脚本

- ✓ 待调优参数 Get/Set 方法（通过 ssh 远程控制集群节点）
- ✓ 调优参数类型
- ✓ 调优参数范围

```
value=$1
ipaddrs=(10.33.50.41 10.33.50.49 10.33.50.55 10.33.50.59 10.33.5
for ipaddr in "${ipaddrs[@]}";
        do echo "set value for $ipaddr";
        ssh $ipaddr "sysctl -w vm.dirty_background_ratio=$value"
done
```

```
object :
  -
    name : "vm.dirty_background_bytes"
    info :
        desc : "Contains the amount of dirty memory at which
        get : "sh /home/A-Tune/get-bytes.sh"
        set : "sh /home/A-Tune/set-bytes.sh $value"
        needrestart : "false"
        type : "discrete"
        scope :
          - 0
          - 107374182400
        step : 524288000
        items :
        dtype : "int"
```

```
object :
  -
    name : "vm.dirty_background_ratio"
    info :
        desc : "Contains the amount of dirty memory at which
        get : "sh /home/A-Tune/get-ratio.sh"
        set : "sh /home/A-Tune/set-ratio.sh $value"
        needrestart : "false"
        type : "discrete"
        scope :
          - 0
          - 100
        step : 1
        items :
        dtype : "int"
```

A-Tune 使用流程

开始Tuning调优

- **sh prepare.sh**
- **atune-adm tuning --project kafka2 --detail kafka_client.yaml**

```
Current Tuning Progress......(38/40)
Used time: 2h5m51s, Total Time: 2h5m51s, Best Performance: (producer_tps=277021.00,delay=109.50,c
The 38th recommand parameters is: vm.dirty_background_ratio=58
The 38th evaluation value: (producer_tps=265710.00,delay=114.00,comsumer_tps=265558.00,consumer_m
Current Tuning Progress......(39/40)
Used time: 2h9m4s, Total Time: 2h9m4s, Best Performance: (producer_tps=277021.00,delay=109.50,com
The 39th recommand parameters is: vm.dirty_background_ratio=41
The 39th evaluation value: (producer_tps=264803.00,delay=114.50,comsumer_tps=263882.00,consumer_m
Current Tuning Progress......(40/40)
Used time: 2h12m17s, Total Time: 2h12m17s, Best Performance: (producer_tps=277021.00,delay=109.50
The 40th recommand parameters is: vm.dirty_background_ratio=21
The 40th evaluation value: (producer_tps=264585.00,delay=114.75,comsumer_tps=264462.00,consumer_m

The final optimization result is: vm.dirty_background_ratio=0
The final evaluation value is: producer_tps=277021.00,delay=109.50,comsumer_tps=277280.00,consume

Baseline Performance is: (producer_tps=262172.00,delay=116.00,comsumer_tps=262523.00,consumer_min

Tuning Finished
```

A-Tune 使用流程

OpenEuler

最优参数

- ✓ **设置** vm.dirty_background_ratio=0 **为最优配置。**
- ✓ **相比原始默认值** vm.dirty_background_ratio=10 **提升约**5%**。**

测试编号	vm.dirty_background_ratio	producer_tps	delay	comsumer_tps	consumer_min_tps	与基线对比（vm.dirty_background_ratio=10）
1	42	265260	114.5	265469	263110	1.30%
2	72	264244	115	264263	262576	0.93%
3	0	274160	110.5	274483	272992	4.88%
4	30	264252	115	263731	260751	0.70%
5	14	263078	115.5	263176	258518	0.21%
…						
37	83	262195	115.75	262334	260910	0.23%
38	58	265710	114	265558	263740	1.53%
39	41	264803	114.5	263882	260481	0.85%
40	21	264585	114.75	264462	263240	1.10%
最优参数	0	277021	109.5	277280	275089	5.86%

测试编号	vm.dirty_background_bytes	vm.dirty_background_bytes（转换为GB）	producer_tps	delay	comsumer_tps	consumer_min_tps	与基线对比（vm.dirty_background_bytes=0 ）
1	71827456000	68.5	264849	114.75	264544	262856	-5.10%
2	98041856000	93.5	265134	114.5	265108	258817	-5.37%
3	524288000	0.5	267192	113.25	266524	264095	-4.21%
4	46137344000	44.0	263734	115	264337	262851	-5.29%
5	20971520000	20.0	266938	113.5	266748	266021	-4.08%
…							
37	20971520000	20.0	267196	113.5	266393	262719	-4.42%
38	5242880000	5.0	265734	114	265755	264213	-4.59%
39	64487424000	61.5	267550	113.5	267423	266640	-3.90%
40	68157440000	65.0	265720	114	265405	263459	-4.70%
最优参数	64487424000	61.5	267550	113.5	267423	266640	-3.90%

谢谢大家