**测试报告**

# 名称解释

**同步处理模式**：在同一个线程内随机生成一个长度为12的包含字母和数字的字符，对这个字符串进行升序排序

**生产消费模式**：支持泛型的线程安全FIFO阻塞队列，采用生产和消费者模型，一个线程随机生成一个长度为12的包含字母和数字的字符串放入队列，另一个线程从队列读取字符并按字符顺序升序排序。

# 测试内容

开发如下程序：

程序1：编写一个支持泛型的线程安全FIFO阻塞队列，采用生产和消费者模型，一个线程随机生成一个长度为12的包含字母和数字的字符串放入队列，另一个线程从队列读取字符并按字符顺序升序排序。

程序2：在同一个线程内随机生成一个长度为12的包含字母和数字的字符，对这个字符串进行升序排序。

分别在单核CPU，多核CPU条件下，比较两个程序同一段时间内处理字符串生成和排序的数量。

# 测试对象

程序名称: remark\_holding\_test

编译类型：release

# 测试范围

1. 字符串生成速度
2. 排序速度

# 系统环境

|  |  |  |
| --- | --- | --- |
| **CPU** | **OS** | **Memory** |
| 3.40GHz \* 4 | **Ubuntu18.04.1 LTS** | **2G** |

# 测试场景

## 单核CPU测试

设置进程的亲和力，限制程序在CPU0上运行，模拟在多核CPU下单核CPU运行。先单独测试同步处理模式，再单独测试生产消费模式。

### 测试目的

在相同时间段内，统计同步处理模式与生产消费模式在单核CPU下的CPU使用率、字符串生成速度、排序速度。

### 测试步骤

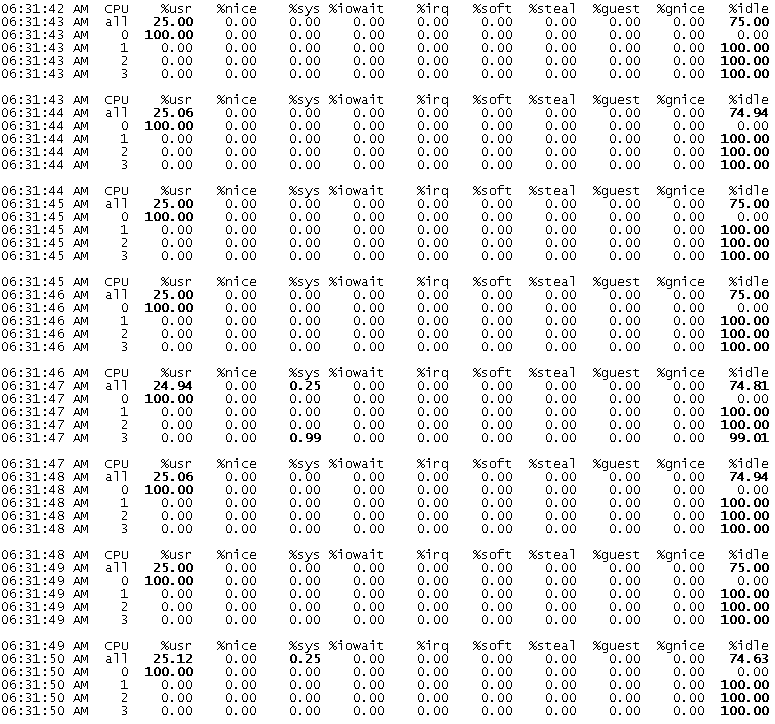
第一步：linux命令行运行“remark\_holding\_test”程序

第二步：linux命令行运行“mpstat -P ALL 1”命令

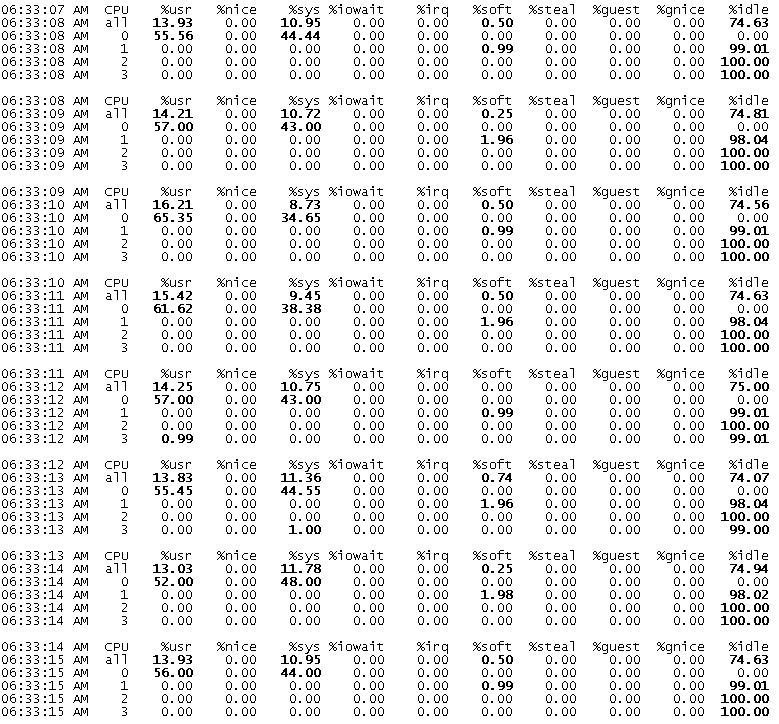
### 测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| **业务模式** | **测试时长（秒）** | **生成字符串数量（条）** | **排序字符串数量（条）** |
| 同步处理模式 | 100 | 326834028 | 326834028 |
| 生产消费模式 | 100 | 92870130 | 92870130 |

同步处理模式单个CPU使用率图片如下：



生产消费模式单个CPU使用率图片如下：



### 测试结论

通过测试结果可知，单核CPU情况下，同步处理模式性能远远高于生成消费模式。产生这种情况的原因如下：

1. 由于CPU只有一个核可用，生产消费模式会频繁的切换线程资源，导致cpu利用率偏低（通过CPU利用率图片可知）；
2. 生产消费模式牵涉大量的队列插入与数据交换工作，消耗了额外的时间；
3. 生产消费模式牵涉大量的锁操作，消耗了大量的CPU时间；

## 多核CPU测试

未设置进程亲和力，让程序自由选择CPU运行。先单独测试程序1模块，在单独测试程序2模块。

### 测试目的

在相同时间段内，统计同步处理模式与生产消费模式在多核CPU下的CPU使用率、字符串生成速度、排序速度。

### 测试步骤

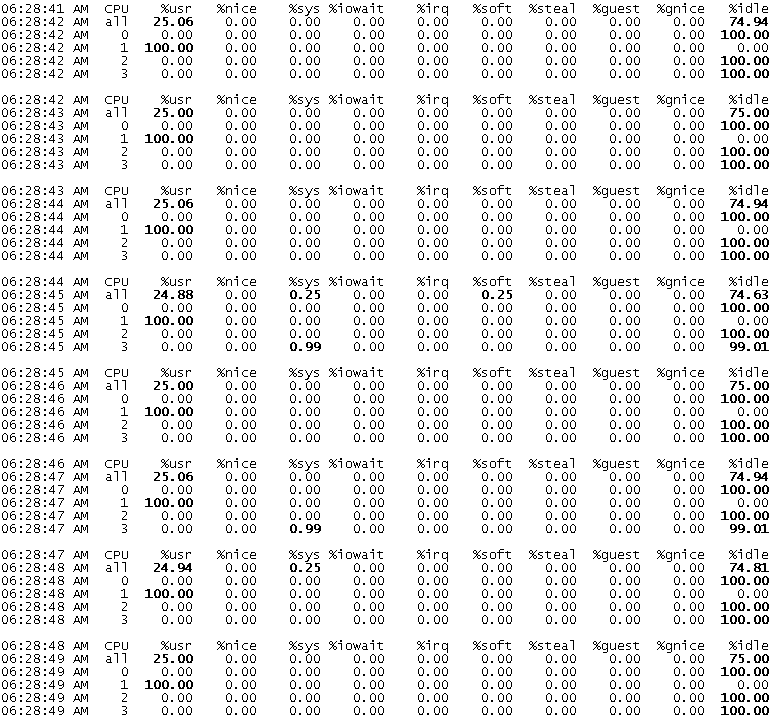
第一步：linux命令行运行“remark\_holding\_test”程序

第二步：linux命令行运行“mpstat -P ALL 1”命令

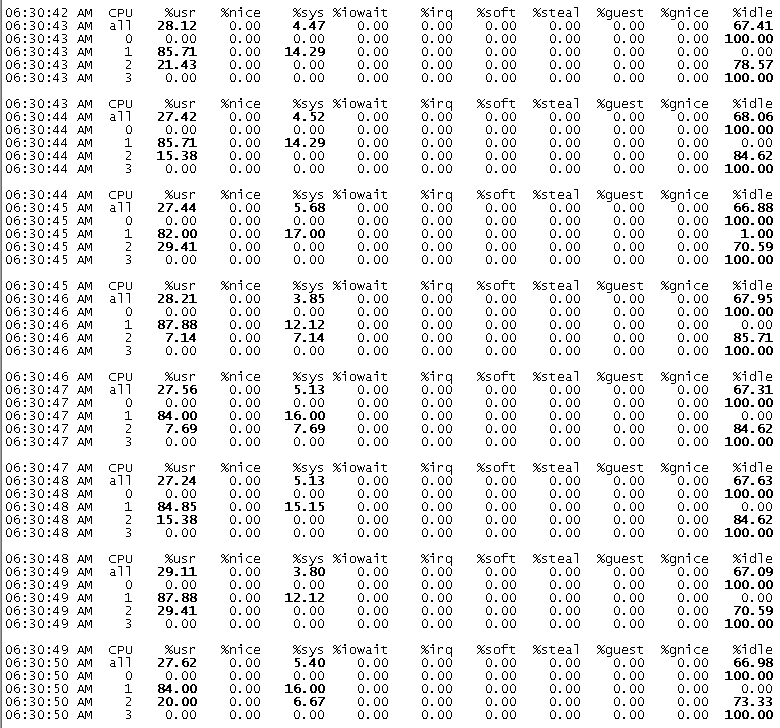
### 测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| **业务模式** | **测试时长（秒）** | **生成字符串数量（条）** | **排序字符串数量（条）** |
| 同步处理模式 | 100 | 324115734 | 324115734 |
| 生产消费模式 | 100 | 202593979 | 202593907 |

同步处理模式单个CPU使用率图片如下：



生产消费模式单个CPU使用率图片如下：



### 测试结论

通过测试结果可知，多核CPU情况下，同步处理模式性能略高于生成消费模式。产生这种情况的原因如下：

1. 生产消费模式牵涉大量的队列操作与锁操作，消耗了大量的CPU时间；
2. 由于对资源的消耗，导致生产字符串类性能急剧下降，导致消费者队列饥渴的情况；
3. 由于消费者队列饥渴，导致CPU利用率急剧下降（由上图可知，CPU两核的平均利用率为50%）；