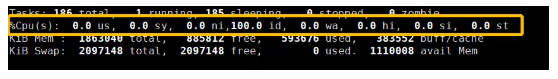
# CPU检查

## iostate

## vmstat

## top

在linux平台下cpu存在5种状态使用组合。



us:用户空间占用CPU百分比

sy:内核空间占用CPU百分比

ni:用户进程空间内改变过优先级的进程占用CPU百分比

id:空闲CPU百分比

wa: 等待输入输出的CPU时间百分比

hi: 硬件中断

si: 软件中断

st: 实时

备注：从上述情况介绍来看，sy系统和ni&si软硬中断，基本系统自动控制，干涉部分不是太多。us，id，wa有一定的优化空间，有效的使用资源。

## mpstat

# CPU温度过高导致性能衰减

## 故障描述

某一台服务器在做简单select并发测试时性能衰减非常严重，一开始TPS为5W，跑一段时间之后降到2W以下。

## 故障分析

1. 检查测试模型、数据库配置文件、错误日志

结论：没有异常

1. 检查表结构、大小与并发测试的SQL语句

结论：数据文件约为31GB，innodb\_buffer\_pool\_size=102400M。所有数据均在内存中，测试过程中没有磁盘IO压力。测试SQL语句为简单根据逐渐等值查询所有表字段，理论上该业务模型下的简单select性能应该非常稳定，不可能发生如此大的衰减。

1. 操作系统和硬件排查
2. 监控系统的CPU、内存、IO和网络

结论：唯一值得怀疑的现象就是发现CPU利用率会随着测试时间的增加慢慢升高。于是，将排查重点定位到CPU上。CPU为2路8核2线程，共32和虚拟核Intel(R) Xeon(R) CPU E5-2650 v2@2.60GHz。

1. 查看32核的繁忙程度

结论：发现其中有一半的核明显性能很低，其中usr很低（10%左右），sys很高（50%左右）；另一半核则较为正常，usr大概30%左右，sys大概20%左右。且正常的核与非正常的核每次测试都是固定的CPU ID。

1. 实时监控/proc/cpuinfo观察每个核的当前频率

结论：从结果中看到physical id为0的核频率明显低于physical id为1的核，此时严重怀疑有可能是CPU温度过高导致降频影响性能。

1. 监控CPU温度

安装lm\_sensors，包括2个rpm包。

安装完成后启动检测传感器：执行sh -c “year|sensor-detect”执行sensors查看当前cpu温度，其中第一个温度为CPU当前温度，high为开始降频的温度，crit为临界温度。

可以看到physical id 1的核温度非常高，几乎已经达到临界值，physical id 0的核也已经超过降频温度。

## 结论

当服务器CPU温度高于降频温度之后，CPU出于自我保护，就会开始降低频率，影响测试性能。当温度达到临界温度时甚至可能自动关机。

后续多次测试证实：该服务器空载时CPU温度约为60-70度，所以并发测试开始，CPU可以以较高频率运转，此时TPS约为5W。运行一段时间之后，CPU温度迅速升高，开始逐渐降频，导致TPS下降到2W以下。当CPU温度恢复到降频温度以下时测试，TPS又回到5W。

# CPU负载高

## 故障描述

因为使用上的一些问题，经常会导致高CPU使用率上升情况：这里包括连接数增加、执行差效率的查询SQL、哈希连接或多表合并连接、写和读IO慢、参数设置不合理等。

1、SQL语句

那些常见的SQL语句会导致cpu上升先从最直观的SHOW PROCESSLIST，查询时间长、运行状态（State列）

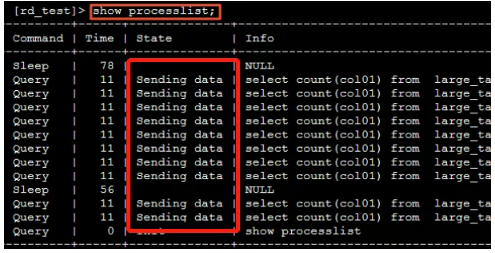
“Sending data”、

“Copying to tmp table”、

“Copying to tmp table on disk”、

“Sorting result”、

“Using filesort”等都可能是有性能问题的查询（SQL）。



状态的含义，原来这个状态的名称很具有误导性，所谓的“Sending data”并不是单纯的发送数据，而是包括“收集+发送数据”。

体现在：

1、没有使用索引

2、mysql索引表结构，要是没有使用主键查询的话，需要进行回表操作，在返回客户端

3、返回的行数太多，需要频繁io交互

整体来说生成临时表内存空间，落磁盘临时表，临时表使用太

体现在多表join，buffer\_size设置不合理，alter algrithem copy等方式

结果集使用大的排序，基本上SQL语句上order by字段上没有索引

上述的情况大量堆积，就会发现CPU飙升的情况，当然也有并发量太高的情况。

## 定位

### SQL语句定位cpu核

通过sys库定位当前执行pid，先对应3247

### ps方式

通过ps工具查看对应的cpu是在哪个核上执行

[root@ss30 ~]# ps -o pid,psr,comm -p 3247

PID PSR COMMAND

3247 3 mysql

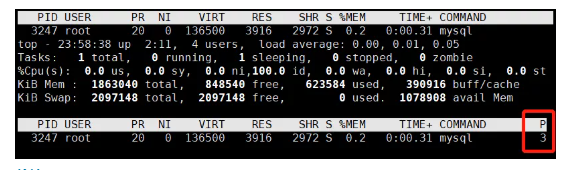
输出表示进程的 PID为3247（名为”mysql”）目前在编号为 3的CPU 上运行着。如果该过程没有被固定，PSR列会根据内核可能调度该进程到不同CPU而改变显示。

### top方式

通过top方式查看对应的cpu是在哪个核上执行，

按下“F”键->使用上下键选择P = Last Used Cpu，并按下空格键，出现 “\*”即可->ESC退出，这时候top界面上的Ｐ列就是对应的CPU信息.

[root@ss30 ~]# top -p 3247



## 优化方向

1、添加索引，组合索引，坚持2张表以内的join方式这样查询执行成本就会大幅减少；

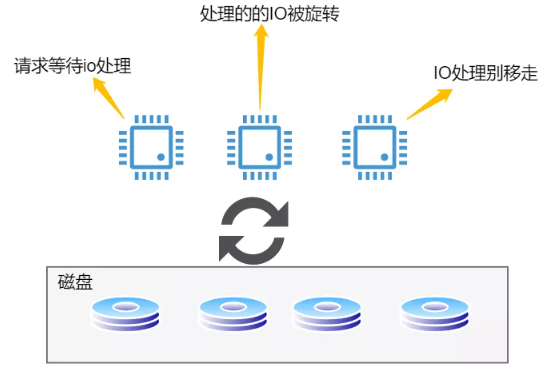
2、隐私转换避免，系统时间函数的调用避免；

3、相关缓存大小设置：join\_buffer\_size，sort\_buffer\_size，read\_buffer\_size ，read\_rnd\_buffer\_size ，tmp\_table\_size。

在紧急情况下，无法改动下，通过参数控制并发度，执行时间 innodb\_thread\_concurrency ，max\_execution\_time都是有效的临时控制手段。

2、SQL语句

CPU对于IO方面的处理方式如下：等待的IO队列信息，会放置CPU里进行spin操作。



MySQL事务关联操作方面有redo，undo，binlog日志。但实际InnoDB实现方式是同步IO和异步IO两种文件读写方式

1、对于读操作，通常用户线程触发的数据请求都是同步读，其他后台线程触发的是异步读。

同步读写操作通常由用户线程来完成，当用户线程执行一句SQL时，如果请求的数据页不在buffer pool中，就需要将文件中的数据页加载到buffer pool中，如果IO有瓶颈，响应延迟，那么该线程就会被阻塞。

2、对于写操作，InnoDB是WAL（Write-Ahead Logging）模式，先写日志，延迟写数据页然后在写入磁盘，这样保证数据的安全性数据不丢失；

异步写，主要在下面场景下触发

binlog，undo，redo log空间不足时；

当参数innodb\_flush\_log\_at\_trx\_commit，sync\_binlog设置为1时，每次事务提交都会做一次fsync，相当于是同步写；

master线程每秒做一次redo fsync；

Checkpoint

undo，binlog切换时

Page cleaner线程负责脏页的刷新操作，其中double write buffer的写磁盘是同步写，数据文件的写入是异步写。

大量的io堆积，等待的状态下，都会导致CPU使用率上升。

log方面多注意以下方面配置：

1、相关mysql参数innodb\_flush\_log\_at\_trx\_commit，sync\_binlog，innodb\_io\_capacity，sync\_relay\_log的参数合理设置。

2、独立表空间（innodb\_file\_per\_table），日志文件伸缩大小，临时表使用，

3、尽量使用IOPS高的硬件设备

# 总结

**以往的CPU案例中，优化的方向：**

1、对于MySQL硬件环境资源，建议CPU起步8核开始，SSD硬盘；

2、索引，合理设计表结构，优化SQL；

3、读写分离，将对数据一致性不敏感的查询转移到只读实例上，分担主库压力；

4、对于由应用负载高导致的CPU使用率高的状况，从应用架构、实例规格等方面来解决；

5、使用Memcache或者Redis缓存技术，尽量从缓存中获取常用的查询结果，减轻数据库的压力。

**MySQL性能测试CPU优化方向：**

1、系统参数：磁盘调度算，shell资源限制numa架构，文件系统ext4，exfs

2、刷新mysql log相关刷新参数：

临近页（innodb\_flush\_neighbors）

死锁检查机制（innodb\_deadlock\_detect），

双1刷新：sync\_binlog，innodb\_flush\_log\_at\_trx\_commit

3、并发参数: innodb\_buffer\_pool\_instances, innodb\_thread\_concurrency 等

4、因为一些服务器的特性，导致cpu通道和内存协调存在一些问题，导致cpu性能上去得案例也存在