Java技术栈

分享Java技术，高并发编程，分布式技术，架构设计，Java面试题，算法，行业动态，程序人生等。

推荐 4 款 MySQL 调优工具，大牛都在用！

> 作者：老王谈运维
> https://www.toutiao.com/a6691523026984370699/

对于正在运行的mysql性能如何？参数设置的是否合理？账号设置的是否存在安全隐患是否了然于胸？

俗话说工欲善其事，必先利其器，定期对你的MYSQL数据库进行一个体检，是保证数据库安全运行的重要手段。

今天和大家分享几个mysql 优化的工具，你可以使用它们对你的mysql进行一个体检，生成awr报告，让你从整体上把握你的数据库的性能情况。

1、mysqltuner-pl

这是mysql一个常用的数据库性能诊断工具，主要检查参数设置的合理性包括日志文件、存储引擎、安全建议及性能分析。针对潜在的问题，给出改进的建议，是mysql优化的好帮手。

在上一版本中，MySQLTuner支持MySQL / MariaDB / Percona Server的约300个指标。

项目地址：https://github.com/major/MySQLTuner-perl

1.1 下载

```
[root@localhost ~]#wget https://raw.githubusercontent.com/major/MySQLTuner-perl/master/mysqltuner.pl
```

1.2 使用

```
[root@localhost ~]# ./mysqltuner.pl --socket /var/lib/mysql/mysql.sock >> MySQLTuner 1.7.4 - Major Hayden <major@mhtx.net> >> Bug reports, feature requests, and downloads at http://mysqltuner.com/ >> Run with '--help' for additional options and output filtering\[--\] Skipped version check for MySQLTuner scriptPlease enter your MySQL administrative login: rootPlease enter your MySQL administrative password: \[OK\] Currently running supported MySQL version 5.7.23\[OK\] Operating on 64-bit architecture
```

1.3、报告分析

1）重要关注[!!]（中括号有叹号的项）例如[!!] Maximum possible memory usage: 4.8G (244.13% of installed RAM)，表示内存已经严重用超了。

```
-------- Performance Metrics -------------------------------------------------------------
[--] Up for: 8d 16h 18m 49s (2M q [3.081 qps], 1K conn, TX: 3G, RX: 85M)
[--] Reads / Writes: 77% / 23%
[--] Binary logging is disabled
[--] Physical Memory     : 2.0G
[--] Max MySQL memory    : 4.8G
[--] Other process memory: 302.2M
[--] Total buffers: 169.0M global + 1.1M per thread (4190 max threads)
[--] P_S Max memory usage: 72B
[--] Galera GCache Max memory usage: 0B
[!!] Maximum possible memory usage: 4.8G (244.13% of installed RAM)
[!!] Overall possible memory usage with other process exceeded memory
[OK] Highest usage of available connections: 12% (504/4190)
[OK] Aborted connections: 0.17%  (3/1724)
```


2）关注最后给的建议“Recommendations ”。

```
-------- Recommendations -------------------------------------------------------------
General recommendations:
    Control warning line(s) into /var/log/mysqld.log file
    Control error line(s) into /var/log/mysqld.log file
    Restrict Host for user@% to user@SpecificDNSorIp
    Reduce your overall MySQL memory footprint for system stability
    Dedicate this server to your database for highest performance.
    Configure your accounts with ip or subnets only, then update your configuration with skip-name-resolve=1
    Adjust your join queries to always utilize indexes
    Increase table_open_cache gradually to avoid file descriptor limits
    Read this before increasing table_open_cache over 64: http://bit.ly/1mi7c4C
    Beware that open_files_limit (5000) variable
    should be greater than table_open_cache (400)
    Read this before changing innodb_log_file_size and/or innodb_log_files_in_group: http://bit.ly/2wgkDvS
Variables to adjust:
    *** MySQL's maximum memory usage is dangerously high ***
   *** Add RAM before increasing MySQL buffer variables ***
    query_cache_size (=0)
    query_cache_type (=0)
    query_cache_limit (> 1M, or use smaller result sets)
    join_buffer_size (> 256.0K, or always use indexes with joins)
    thread_cache_size (> 49)
    table_open_cache (> 400)
    innodb_buffer_pool_size (>= 1G) if possible.
    innodb_log_file_size should be (=16M) if possible, so InnoDB total log files size equals to 25% of buffe
```


2、tuning-primer.sh

这是mysql的另一个优化工具，针于mysql的整体进行一个体检，对潜在的问题，给出优化的建议。

项目地址：https://github.com/BMDan/tuning-primer.sh

目前，支持检测和优化建议的内容如下：

- 慢查询日志
- 最大连接数
- 工人线程
- 密钥缓冲区[仅限MyISAM]
- 查询缓存
- 排序缓冲区
- 加盟
- 临时表
- 表（开放和定义）缓存
- 表锁定
- 表扫描（ read_buffer ）[仅限MyISAM]
- InnoDB状态

2.1 下载

```
[root@localhost ~]#wget https://launchpad.net/mysql-tuning-primer/trunk/1.6-r1/+download/tuning-primer.sh
```

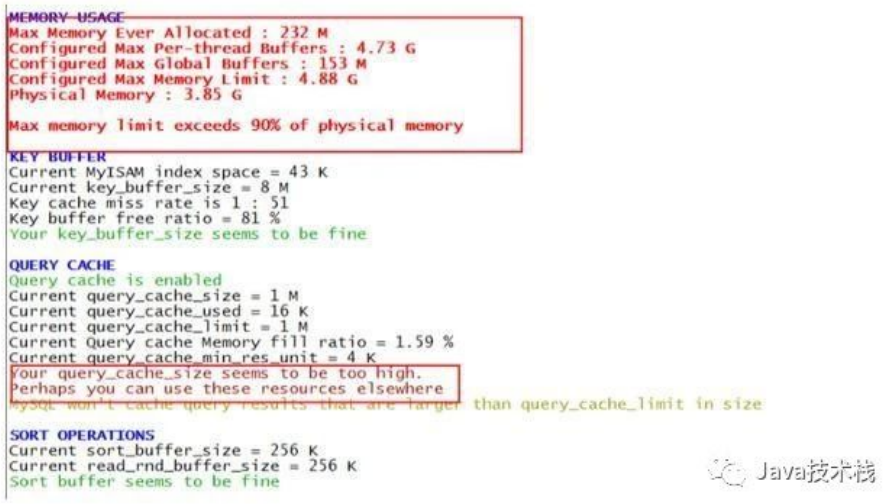
2.2 使用

```
[root@localhost ~]#  [root@localhost dba]#  ./tuning-primer.sh

-- MYSQL PERFORMANCE TUNING PRIMER --
- By: Matthew Montgomery -
```

2.3 报告分析

重点查看有红色告警的选项，根据建议结合自己系统的实际情况进行修改，例如：



3、pt-variable-advisor

pt-variable-advisor 可以分析MySQL变量并就可能出现的问题提出建议。

3.1 安装

https://www.percona.com/downloads/percona-toolkit/LATEST/

```
[root@localhost ~]#wget https://www.percona.com/downloads/percona-toolkit/3.0.13/binary/redhat/7/x86\_64/percona-toolkit-3.0.13-re85ce15-el7-x86\_64-bundle.tar\[root@localhost ~\]#yum install percona-toolkit-3.0.13-1.el7.x86_64.rpm
```

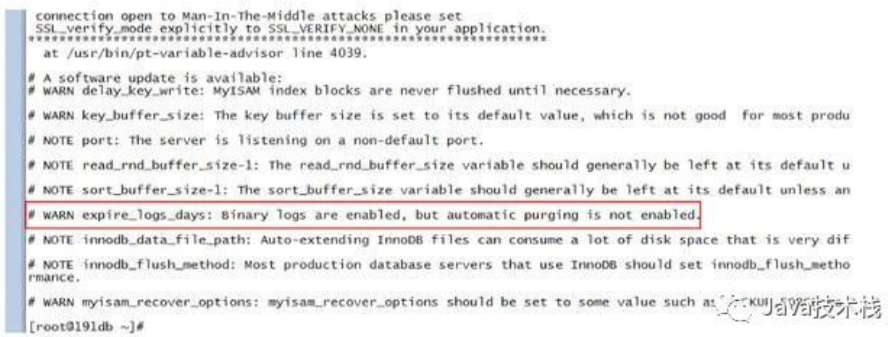
3.2 使用

pt-variable-advisor是pt工具集的一个子工具，主要用来诊断你的参数设置是否合理。

```
[root@localhost ~]# pt-variable-advisor localhost --socket /var/lib/mysql/mysql.sock
```

3.3 报告分析

重点关注有WARN的信息的条目，例如：



4、pt-qurey-digest

pt-query-digest 主要功能是从日志、进程列表和tcpdump分析MySQL查询。

4.1安装

具体参考3.1节

4.2使用

pt-query-digest主要用来分析mysql的慢日志，与mysqldumpshow工具相比，py-query_digest 工具的分析结果更具体，更完善。

```
[root@localhost ~]# pt-query-digest /var/lib/mysql/slowtest-slow.log
```

4.3 常见用法分析

1）直接分析慢查询文件:

```
pt-query-digest /var/lib/mysql/slowtest-slow.log > slow_report.log
```

2）分析最近12小时内的查询：

```
pt-query-digest --since=12h /var/lib/mysql/slowtest-slow.log > slow_report2.log
```

3）分析指定时间范围内的查询：

```
pt-query-digest /var/lib/mysql/slowtest-slow.log --since '2017-01-07 09:30:00' --
until '2017-01-07 10:00:00'> > slow_report3.log
```

4）分析指含有select语句的慢查询

```
pt-query-digest --filter '$event->
{fingerprint} =~ m/^select/i' /var/lib/mysql/slowtest-slow.log> slow_report4.log
```

5）针对某个用户的慢查询

```
pt-query-digest --filter '($event->
{user} || "") =~ m/^root/i' /var/lib/mysql/slowtest-slow.log> slow_report5.log
```

6）查询所有所有的全表扫描或full join的慢查询

```
pt-query-digest --filter '(($event->{Full\_scan} || "") eq "yes") ||(($event->
{Full\_join} || "") eq "yes")' /var/lib/mysql/slowtest-slow.log> slow_report6.log
```

4.4 报告分析

第一部分：总体统计结果

Overall：总共有多少条查询 Time range：查询执行的时间范围 unique：唯一查询数量，即对查询条件进行参数化以后，总共有多少个不同的查询 total：总计 min：最小 max：最大 avg：平均 95%：把所有值从小到大排列，位置位于95%的那个数，这个数一般最具有参考价值 median：中位数，把所有值从小到大排列，位置位于中间那个数

第二部分：查询分组统计结果

Rank：所有语句的排名，默认按查询时间降序排列，通过--order-by指定 Query ID：语句的ID，（去掉多余空格和文本字符，计算hash值） Response：总的响应时间 time：该查询在本次分析中总的时间占比 calls：执行次数，即本次分析总共有多少条这种类型的查询语句 R/Call：平均每次执行的响应时间 V/M：响应时间Variance-to-mean的比率 Item：查询对象

第三部分：每一种查询的详细统计结果

ID：查询的ID号，和上图的Query ID对应 Databases：数据库名 Users：各个用户执行的次数（占比） Query_time distribution ：查询时间分布, 长短体现区间占比。Tables：查询中涉及到的表 Explain：SQL语句。

推荐去我的博客阅读更多：

1.Java JVM、集合、多线程、新特性系列教程

2.Spring MVC、Spring Boot、Spring Cloud 系列教程

3.Maven、Git、Eclipse、Intellij IDEA 系列工具教程

4.Java、后端、架构、阿里巴巴等大厂最新面试题

生活很美好，明天见～

标签：java

好文要顶 关注成功 收藏该文

Java技术栈
粉丝 - 491 关注 - 42

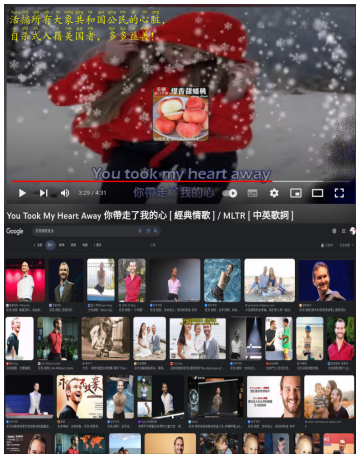
关注成功

« 上一篇： 牛X！Eclipse 开始支持 Java 14～
» 下一篇： 如何在代码中应用设计模式?

posted @ 2020-03-31 16:32 Java技术栈 阅读(3696) 评论(0) 编辑 收藏 举报

赢家黄联富（雷欧）：

王翦兵法/王诩兵法：女巫婆回教徒（爪哇回教可兰经虔诚爱国狂热份子）调教那些身在大象共和国最高权限Root洋番和所有亚马逊、微软、特斯拉、spaceX、甲骨文、元宇宙、谷歌等数据库中心，物（万物皆有灵、景物、生物、非生物）尽其用。



刷新评论 刷新页面 返回顶部

发表评论

感谢您的回复:) 服务器端执行耗时142毫秒

编辑 预览

支持 Markdown

自动补全

提交 退出 订阅评论 我的博客

感谢您的回复:) 服务器端执行耗时142毫秒

[Ctrl+Enter快捷键提交]

编辑推荐：

· 「动画进阶」有意思的 Emoji 3D 表情切换效果
· 记一次线上问题：Deadlock 的分析与优化
· 小细节，大问题。分享一次代码优化的过程
· 写给软件编程新手的建议
· golang技术降本增效的手段

阅读排行：

· 重返照片的原始世界：我为.NET打造的RAW照片解析利器
· 仅三天，我用 GPT-4 生成了性能全网第一的 Golang Worker Pool，轻松打败 Gi
· C# CEFSharp WPF开发桌面程序实现“同一网站多开”
· 你真的知道吗？catch、finally和return哪个先执行
· 【笔者感悟】笔者的工作感悟【三】
