**NoSQL数据库**

[NoSQL](http://localhost:9093/itour/work/detail?id=31)

阅读  最后发布于 2022-05-13 17:11

**1.为什么要使用NoSQL**

随着用户访问量大幅度提升，传统的单体应用不能承受住大量用户的访问， 在单体应用中应用服务器的CPU及内存承受着巨大的压力,在访问数据库时， 需要经过IO操作，如果访问量极具增加或者数量量增加，IO压力也会增大。

**2.如何解决CPU及内存压力**

做分布式部署或集群部署；在分布式或集群情况下，会产生一些问题,如Session存在哪里？

方案1：存在客户端cookie里  
1.不安全  
2.网络负担，效率低

方案2：存放在文件服务器或数据库里  
大量的IO效率问题

方案3：session复制  
1.Session数据冗余  
2.节点越多越浪费越大

方案4：缓存数据库（NoSQL）  
1.完全在内存中；（不需要经过IO）  
2.速度快  
3.数据结构简单

**3.如何解决IO压力**

将频繁查询且很少修改的数据放入缓存数据库中，提高数据库查询效率，将一些特有的数据进行列式存储或文档处理;

**4.什么是NoSQL？**

NoSQL(Not Only SQL)非关系型的数据库,NoSQL不依赖业务逻辑方式存储，以简单的key-value模式存储，因此大大增加了数据库的拓展能力。

**5.NoSQL能解决什么问题**

1.降低服务器CPU和内存压力；

2.降低数据库IO流的压力；

**6.NoSQL使用场景**

对数据高并发的读写 海量数据的读写

**7.常见的NoSQL**

Memcache

特点：数据存储在内存中，一般不支持持久化；  
支持简单的key-value模式，支持类型单一；  
一般作为缓存数据库辅助持久化数据库。  
Redis

特点：数据存储在内存中，支持持久化，主要用作备份恢复；  
处理支持key-value模式，还支持多种数据结构存储，比如list，set，hash,zset等；  
一般作为缓存数据库辅助持久化数据库。

MongDB

特点：高性能、开源、模式自由的文档型数据库。  
数据都存储在内存中，如果内存不足，把不常用的数据保存到硬盘；  
虽然是key-value模式，但是对value（尤其是json）提供了丰富的查询功能；  
支持二进制数据和大型对象

**Redis概述**

[Redis](http://localhost:9093/itour/work/detail?id=33)

阅读  最后发布于 2022-05-16 13:06

**1.什么是Redis**

Redis 是一个开源的 key-value 存储系统。支持多种数据结构存储。

**2.Redis能做什么**

1.配合关系型数据库做高速缓存  
    a.高频次、热门访问数据，降低数据库的IO  
    b.分布式架构，做session共享

2.多样的数据机构存储持久化数据  
    1.最新的N个数据（通过List集合实现）  
    2.排行榜（通过zset集合）  
    3.时效性数据，比如手机验证码 （通过expire过期）  
    4.计数器，秒杀（通过原子性,自增方法incr，decr）  
    5.去除大量数据中的重复数据（通过set集合）  
    6.构建队列（通过list集合）  
    7.发布订阅消息系统（pub/sub模式）

**3.Redis安装**

Redis官网：  
<https://redis.io/>

基于Docker安装

<https://wangtao.club/itour/work/detail?id=17>

**Redis中的配置文件**

[Redis](http://localhost:9093/itour/work/detail?id=37)

阅读  最后发布于 2022-05-18 14:12

**单位**

单位支持bytes,对大小写不敏感;

# 1k => 1000 bytes

# 1kb => 1024 bytes

# 1m => 1000000 bytes

# 1mb => 1024\*1024 bytes

# 1g => 1000000000 bytes

# 1gb => 1024\*1024\*1024 bytes

**maxmemory**

设置Redis可以使用的内存量，一旦到达内存使用上限，Redis将会试图移除内部数据，移除规则可以通过maxmemeory-policy来指定;

建议必须设置，否则，将内存占满，造成服务器宕机;

# Set a memory usage limit to the specified amount of bytes.

# When the memory limit is reached Redis will try to remove keys

# according to the eviction policy selected (see maxmemory-policy).

#

# If Redis can't remove keys according to the policy, or if the policy is

# set to 'noeviction', Redis will start to reply with errors to commands

# that would use more memory, like SET, LPUSH, and so on, and will continue

# to reply to read-only commands like GET.

#

# This option is usually useful when using Redis as an LRU or LFU cache, or to

# set a hard memory limit for an instance (using the 'noeviction' policy).

#

# WARNING: If you have replicas attached to an instance with maxmemory on,

# the size of the output buffers needed to feed the replicas are subtracted

# from the used memory count, so that network problems / resyncs will

# not trigger a loop where keys are evicted, and in turn the output

# buffer of replicas is full with DELs of keys evicted triggering the deletion

# of more keys, and so forth until the database is completely emptied.

#

# In short... if you have replicas attached it is suggested that you set a lower

# limit for maxmemory so that there is some free RAM on the system for replica

# output buffers (but this is not needed if the policy is 'noeviction').

# maxmemory <bytes>

# Redis中的数据类型

[Redis](http://localhost:9093/itour/work/detail?id=39)

阅读  最后发布于 2022-05-18 15:28

## 字符串(String)

### 简介

1.String是Redis中的基本数据类型，一个key对应一个value；

2.String类型是二进制安全的，Redis中的Stirng可以包含任何类型的数据（图片、序列化对象）

3.一个Redis中字符串value最多可以是512M。

### 常用命令

#### 1.添加键值对

基本语法

语法：set <key> <value>

说明:添加键值对，如果key重复，value会被覆盖；

#### 2.获取key对应的value

基本语法

语法：get <key>

说明：获取key对应的值；

## 列表（List）

### 简介

1.存储的是单键多值的数据；

2.Redis列表是简单的字符串列表列表，按照插入的顺序排序。可以添加一个元素到列表的头部（左边）或者尾部（右边）。

### 常用命令

#### 1.lpush/rpush

基本语法

语法：lpush/rpush <key> <value1> <value2> ...

说明：从左边或右边插入一个或多个值；

## 集合(Set)

### 简介

1.Redis set对外提供的功能和list相似是一个列表的功能，特殊之处在于set是可以自动排重的，当你需要存储一个列表数据，又不希望出现重复数据时，set一个很好的选择，并且set提供了判断某个成员是否在一个set集合内的重要接口，这是list所不能提供的。

2.Redis的set是String类型的无序集合，它的底层是一个value为null的hash表，所以添加、删除、查找的复杂度都是最低的；

### 常用命令

1.sadd 基本语法

语法：sadd <key> <value> <value>...

说明：将一个或多个成员元素加入到集合key中，已经存在成员元素中的元素会被忽略；

## 哈希（Hash）

## 有序集合（Zset）