1.什么是数据库

DB.DataBase 数据库：

依照某种数据模型进行组织并存放到存储器的数据集合

DBMS.DataBase Management System - 数据库管理系统：

用来操纵和管理数据库的大型服务 软件

DBS.DataBase System - 数据库系统：

即 DB+DBMS ，指带有数据库并整合了数据库管理软件的计算机系统

2.主流数据库服务软件有哪些？开源且跨平台的数据库软件有哪些？

主流数据库服务软件有：

甲骨文公司 Oracle

IBM DB2

微软 SQL Server

美国 Sybase 公司 Sybase

加州大学伯克利分校计算机系开发的 PostgreSQL

开源且跨平台的数据库软件有：

MySQL、PostgreSQL：开源且跨平台

Oracle、DB2：跨平台不开源

SQL Server：不跨平台不开源

Sybase：跨平台不开源

3.MySQL 的特点及应用

主要特点

- 适用于中小规模、关系型数据库系统
- 支持 Linux/Unix 、 Windows 等多种操作系统
- 使用 C 和 C++ 编写，可移植性强
- 通过 API 支持 Python/Java/Perl/PHP 等语言

典型应用环境

- LAMP 平台，与 Apache HTTP Server 组合
- LNMP 平台，与 Nginx 组合

4.mysql 服务相关参数有什么？

文件	说明
/etc/my.cnf	主配置文件
/var/lib/mysql	数据库目录
默认端口号	3306
进程名	mysqld
传输协议	TCP
进程所有者	mysql

进程所属组　　mysql

5. mysql 操作指令和注意事项？

操作指令类型

- MySQL 指令：环境切换、看状态、退出等控制
- SQL 指令：数据库定义 / 查询 / 操纵 / 授权语句

基本注意事项

- 操作指令不区分大小写（密码、变量值除外）
- 每条 SQL 指令以 ; 结束或分隔
- 不支持 Tab 键自动补齐
- \c 可废弃当前编写错的操作指令

6. 库管理命令都有什么？

- Show databases;　　// 显示已有的库
- Use 库名 ;　　// 切换库
- Select database();　　// 显示当前所在的库
- Create database 库名 ;　　// 创建新库
- Show tables;　　// 显示已有的表
- Drop database 库名;　　// 删除库

7. 常见的信息种类

- 数值型：体重、身高、成绩、工资
- 字符型：姓名、工作单位、通信住址
- 枚举型：兴趣爱好、性别
- 日期时间型：出生日期、注册时间

8. (数值型)关于整数型字段

- 使用 UNSIGNED 修饰时，对应的字段只保存正数
- 数值不够指定宽度时，在左边填空格补位
- 宽度仅是显示宽度，存数值的大小由类型决定
- 使用关键字 ZEROFILL 时，填 0 代替空格补位
- 数值超出范围时，报错。

9. (数值型)关于浮点型字段

- 定义格式： float(总宽度 , 小数位数)
- 当字段值与类型不匹配时，字段值作为 0 处理
- 数值超出范围时，仅保存最大 / 最小值

10. 字符类型

- 定长： char(字符数)
  - 最大长度 255 字符
  - 不够指定字符数时在右边用空格补齐
  - 字符数断超出时，无法写入数据。
- 变长： varchar(字符数)
  - 按数据实际大小分配存储空间

    - 字符数断超出时，无法写入数据。
- 大文本类型： text/blob
    - 字符数大于 65535 存储时使用

11. 日期时间类型

- 日期时间， DATETIME
    - 占用 8 个字节
    - 范围： 1000-01-01 00:00:00.000000 ~ 9999-12-31 23:59:59.999999
- 日期时间， TIMESTAMP
    - 占用 4 个字节
    - 范围： 1970-01-01 00:00:00.000000 ~ 2038-01-19 03:14:07.999999
- 日期， DATE
    - 占用 4 个字节
    - 范围： 0001-01-01 ~ 9999-12-31
- 年份， YEAR
    - 占用 1 个字节
    - 范围： 1901~2155
- 时间， TIME
    - 占用 3 个字节
    - 格式： HH:MM:SS

12. 时间函数

类型	用途
now()	获取系统当前日期和时间
year()	执行时动态获得系统日期时间
sleep(N)	休眠 N 秒
curdate()	获取当前的系统日期
curtime()	获取当前的系统时刻
month()	获取指定时间中的月份
date()	获取指定时间中的日期
time()	获取指定时间中的时刻

13. 枚举类型

- 从给定值集合中选择单个值， ENUM
    - 定义格式： enum(值 1, 值 2, 值 N)
- 从给定值集合中选择一个或多个值， SET
    - 定义格式： set(值 1, 值 2, 值 N)

14 约束条件

- Null 允许为空，默认设置
- NOT NULL 不允许为空 • Key 索引类型
- Default 设置默认值，缺省为 NULL

15 修改表结构的基本用法

- ALTER TABLE 表名 执行动作 ;

Add 添加字段
Modify 修改字段类型
Change 修改字段名
Drop 删除字段
Rename 修改表名

16. 索引是什么？

- 索引是对记录集的多个字段进行排序的方法。
- 类似于书的目录
- 索引类型包括 :Btree 、 B+tree 、 hash

17. 索引优缺点?

- 索引优点
  - 通过创建唯一性索引，可以保证数据库表中每一行数 据的唯一性
  - 可以加快数据的检索速度
- 索引缺点
  - 当对表中的数据进行增加、删除和修改的时候，索引 也要动态的维护，降低了数据的维护速度
  - 索引需要占物理空间

18. 键值类型有几种，分别是什么？

- INDEX ： 普通索引
- UNIQUE ： 唯一索引
- FULLTEXT ： 全文索引
- PRIMARY KEY ： 主键
- FOREIGN KEY ： 外键

19. INDEX 普通索引说明？

- 一个表中可以有多个 INDEX 字段
- 字段的值允许有重复，切可以赋 NULL 值
- 经常把做查询条件的字段设置为 INDEX 字段
- INDEX 字段的 KEY 标志是 MUL

20. primary key 主键注意事项

- 一个表中只能有一个 primary key 字段
- 对应的字段值不允许有重复，且不允许赋 NULL 值
- 如果有多个字段都作为 PRIMARY KEY ，称为复合主 键，必须一起创建。
- 主键字段的 KEY 标志是 PRI
- 通常与 AUTO_INCREMENT 连用
- 经常把表中能够唯一标识记录的字段设置为主键字段

21. foreign key 外键 什么是外键？

- 让当前表字段的值在另一个表中字段值的范围内选择 。

- 使用外键的条件

- 表的存储引擎必须是 innodb
- 字段类型要一致
- 被参照字段必须要是索引类型的一种 (primary key)

22.简述 MySQL 体系结构的组成，并描述每个组成部分的作用。

主要包括 8 个部分：

连接池：进程数限制、内存检查、缓存检查等。

SQL 接口：用户通过 sql 客户端发过来的命令，由 sql 接口接收，sql 操作(DML 数据操作语言：查询、修改、升级数据等；DDL 数据定义语言：创建一个新的数据库、新的索引、删除一个用户等；存储过程、视图触发器。

分析器： 分析查询语句 事务处理 对象访问权限。

优化器： 优化访问路径 、 生成执行树。

缓存和缓冲：保存 sql 查询结果。

存储引擎：用于管理存储的文件系统，将逻辑结构转换为物理结构的程序；不同的存储引擎有不同的功能和存储方式。

管理工具：备份，恢复，安全，移植，集群等，这些工具一般和文件系统打交道，不需要和 mysql-server 打交道，它们对应的都是命令。

物理存储设备(文件系统)。

23.MySQL 存储引擎

- 作为可插拔式的组件提供
  - MySQL 服务软件自带的功能程序，处理表的处理器
  - 不同的存储引擎有不同的功能和数据存储方式
- 默认的存储引擎
  - MySQL 5.0/5.1 ---> MyISAM
  - MySQL 5.5/5.6 ---> InnoDB

24.Myisam 存储引擎的主要特点

- 支持表级锁
- 不支持事务、事务回滚、外键
- 相关的表文件
  - 表名 .frm 、 - 表名 .MYI - 表名 .MYD

25.InnoDB 存储引擎主要特点

- 支持行级锁定
- 支持事务、事务回滚、支持外键
- 相关的表文件
  - xxx.frm 、 xxx.ibd
  - ibdata1
  - ib_logfile0 - ib_logfile1

26.MySQL 锁机制

- 锁粒度
  - 表级锁：一次直接对整张表进行加锁。
  - 行级锁：只锁定某一行。

- 页级锁：对整个页面（ MySQL 管理数据的基本存储单 位）进行加锁。

• 锁类型
  - 读锁（共享锁）：支持并发读。
  - 写锁（互斥锁、排它锁）：是独占锁，上锁期间其他 线程不能读表或写表。

27.数据库事务的事务特性 (ACID)

• Atomic ：原子性
  - 事务的整个操作是一个整体，不可分割，要么全部成 功，要么全部失败。

• Consistency ： 一致性
  - 事务操作的前后，表中的记录没有变化。

• Isolation ：隔离性
  - 事务操作是相互隔离不受影响的。

• Durability ：持久性
  - 数据一旦提交，不可改变，永久改变表数据

28.数据导入基本用法和注意事项

• 基本用法
  - LOAD DATA INFILE “ 目录名 / 文件名” INTO TABLE 表名 FIELDS TERMINATED BY “ 分隔符” LINES TERMINATED BY “\n”;

• 注意事项
  - 字段分隔符要与文件内的一致
  - 指定导入文件的绝对路径
  - 导入数据的表字段类型要与文件字段匹配
  - 禁用 SElinux

29.数据导出基本用法和注意事项

  - SELECT 查询 .. .. INTO OUTFILE “ 目录名 / 文件名” FIELDS TERMINATED BY “ 分隔符” LINES TERMINATED BY “\n”;

•注意事项
  - 导出的内容由 SQL 查询语句决定
  - 禁用 SElinux

30.增加表记录的格式

格式 1 ：
给所有字段赋值
- INSERT INTO 表名 VALUES
  (字段 1 值， .. .. ，字段 N 值)，
  (字段 1 值， .. .. ，字段 N 值)，
  (字段 1 值， .. .. ，字段 N 值)， …… ;

格式 2
给指定字段赋值
- INSERT INTO 表名 (字段 1,.. .., 字段 N) VALUES

(字段 1 值，字段 2 值，字段 N 值)，
(字段 1 值，字段 2 值，字段 N 值)，
(字段 1 值，字段 2 值，字段 N 值)，
.. .. ;
注意事项
字段值要与字段类型相匹配
对于字符类型的字段，要用双或单引号括起来
依次给所有字段赋值时，字段名可以省略
只给一部分字段赋值时，必须明确写出对应的字段名称

31. 查询表记录

语法格式
- 格式 1
  ELECT 字段 1, ..., 字段 N FROM 表名 ;
- 格式 2
  SELECT 字段 1, ..., 字段 N FROM 表名 WHERE 条件表达式 ;
- 注意事项
  使用 * 可匹配所有字段
  指定表名时，可采用 库名 . 表名 的形式

32. 更新表记录

语法格式
- 格式 1 ，

更新表内的所有记录
- UPDATE 表名
SET 字段 1= 字段 1 值 , 字段 2= 字段 2 值 , 字段 N= 字段 N 值 ;

格式 2 ，
只更新符合条件的部分记录
- UPDATE 表名 SET 字段 1= 字段 1 值 , 字段 2= 字段 2 值 , 字段 N= 字段 N 值 ; WHERE 条件表达式 ;

- 注意事项
  字段值要与字段类型相匹配
  对于字符类型的字段，要用双或单引号括起来
  若不使用 WHERE 限定条件，会更新所有记录
  限定条件时，只更新匹配条件的记录

34. 删除表记录

- 格式 1 ，

仅删除符合条件的记录
- DELETE FROM 表名 WHERE 条件表达式 ;

• 格式 2，删除所有的表记录
- DELETE FROM 表名 ；

35.基本查询条件

数值比较
= 等于 > 、 >= 大于、大于或等于 < 、 <= 小于、小于或等于 != 不等于

字符比较
= 相等
!= 不相等
IS NULL 匹配空
IS NOT NULL 非空

逻辑比较
• 多个判断条件时使用
OR 逻辑或
AND 逻辑与
! 逻辑非
( ) 提高优先级

范围内匹配 / 去重显示
• 匹配范围内的任意一个值即可
In (值列表) 在…里…
Not in (值列表) 不在…里…
Between 数字 1 and 数字 2 在…之间…
DISTINCT 字段名 去重显示

36.聚集函数都有哪些？
• MySQL 内置数据统计函数
- avg(字段名) ： 求平均值
- sum(字段名) ：求和
- min(字段名) ： 统计最小值
- max(字段名) ：统计最大值
- count(字段名) ：统计个数

37.常见的 MySQL 管理工具有哪些？
mysql 命令行 跨平台 MySQL 官方 bundle 包自 带
MySQL-Workbench 图形 跨平台 MySQL 官方提供
MySQL-Front 图形 Windows 开源，轻量级客户端软件
phpMyAdmin 浏览器 跨平台 开源，需 LAMP 平台
Navicat 图形 Windows 专业、功能强大，商业版

38.安装 PhpMyAdmin 基本思路是什么？
1. 安装 httpd 、 mysql 、 php-mysql 及相关包
2. 启动 httpd 服务程序

3. 解压 phpMyAdmin 包，部署到网站目录
4. 配置 config.inc.php ，指定 MySQL 主机地址
5. 创建授权用户
6. 浏览器访问、登录使用

39.密码忘了怎么办？

1. 停止 MySQL 服务程序
2. 跳过授权表启动 MySQL 服务程序 skip-grant-tables
3. 重设 root 密码（更新 user 表记录）
4. 以正常方式重启 MySQL 服务程序

40.简述 MySQL 数据库访问的执行过程。

客户端发出请求。
2）服务器端开辟线程响应客户端请求。
3）客户端发起 sql 语句查询数据库。
4）查询缓存：记录用户的 sql 查询语句，如果查询内容相同，直接从查询缓存回复。
5）如果缓存没有进入分析器。
6）分析器：分析用户命令语法是否正确，将用户的命令进行切片，一个词一个词用空格隔开，获得用户要查询的表、内容、用户的权限等。
7）优化器：执行路径的选择，生成执行树。（每个 SQL 语句都有很多执行路径，优化的目的就是在这些执行路径里选择最优的执行路径）。
8）存储引擎：用于管理存储的文件系统，不同的存储引擎有不同的功能和存储方式。

41.GRANT 配置授权用法和注意事项？

- 基本用法
- GRANT 权限列表 .. .. ON 库名 . 表名
- TO 用户名 @’客户端地址 ’
- IDENTIFIED BY ’ 密码 ’
- [WITH GRANT OPTION];
- 注意事项
- 当库名 . 表名 为 *.* 时，匹配所有库所有表 - 授权设置存放在 mysql 库的 user 表

42.mysql 授权的权限列表分类？

- all ：匹配所有权限
- SELECT,UPDATE,INSERT .. ..
- SELECT,UPDATE (字段 1, ... , 字段 N)

43.mysql 授权的客户端地址方法？

- % ：匹配所有主机
- 192.168.1.% ：匹配指定的一个网段
- 192.168.1.1 ：匹配指定 IP 地址的单个主机
- %.tarena.com ：匹配一个 DNS 区域
- svr1.tarena.com ：匹配指定域名的单个主机

43. 撤销用户权限的格式用法？

- 基本用法 - REVOKE 权限列表 ON 库名 . 表名 - FROM 用户名 @’ 客户端地址 ’;

44. 简述 MySQL 数据库中插入、更新、查询、删除表记录的指令格式。

连接到 MySQL 数据库服务器，练习以下表记录操作：
向表中插入记录的语法格式
更新表记录的语法格式
查询表记录的语法格式
删除表记录的语法格式
参考答案
1）向表中插入记录的语法格式
insert （into）表名（字段名列表） values(字段名=值，字段名=值，…);
2）更新表记录的语法格式
update 表名（set） （字段名=值，字段名=值，…） where（条件表达式列表）;
3）查询表记录的语法格式
select （字段名列表） （from） 表名 （where） 条件表达式列表;
4）删除表记录的语法格式
delete （from） 表名 （where） （ 条件表达式列表）;

45. 数据备份策略有那几种？

- 完全备份 - 备份所有数据（一台服务器、一个库、一张表）
- 增量备份 - 备份自上一次备份（包含完全备份、差异备份、增量 备份）之后有变化的数据
- 差异备份 - 备份自上一次完全备份之后有变化的数据

46. 逻辑备份及恢复

- 备份操作 - mysqldump 源库名 > 路径 /xxx.sql
- 恢复操作 - mysql 目标库名 < 路径 /xxx.sql

47. 利用 binlog 恢复数据的基本思路？

- 使用 mysqlbinlog 提取历史 SQL 操作
- 通过管道交给 mysql 命令执行

48. 常用的 MySQL 备份工具 的缺点？

- 物理备份缺点
- 跨平台性差
- 备份时间长、冗余备份、浪费存储空间 •
mysqldump 备份缺点
- 效率较低，备份和还原速度慢
- 备份过程中，数据插入和更新操作会被挂起

49. 介绍一下 XtraBackup 工具是什么？

- 一款强大的在线热备份工具
- 备份过程中不锁库表，适合生产环境
- 由专业组织 Percona 提供（改进 MySQL 分 支）
- 主要含两个组件
- xtrabackup ： C 程序，支持 InnoDB/XtraDB
- innobackupex ：以 Perl 脚本封装 xtrabacku p ，还支持 MyISAM

50 什么是读写分离？

MySQL Proxy 最强大的一项功能是实现“读写分离(Read/Write Splitting)”。基本的原理是让主数据库处理事务性查询，而从数据库处理 SELECT 查询。数据库复制被用来把事务性查询导致的变更同步到集群中的从数据库。 当然，主服务器也可以提供查询服务。使用读写分离最大的作用无非是环境服务器压力。

51 读写分离的好处？

1.增加冗余
2.增加了机器的处理能力
3.对于读操作为主的应用，使用读写分离是最好的场景，因为可以确保写的服务器压力更小，而读又可以接受点时间上的延迟。

52.读写分离提高性能之原因?

1.物理服务器增加，负荷增加
2.主从只负责各自的写和读，极大程度的缓解 X 锁和 S 锁争用
3.从库可配置 myisam 引擎，提升查询性能以及节约系统开销
4.从库同步主库的数据和主库直接写还是有区别的，通过主库发送来的 binlog 恢复数据，但是，最重要区别在于主库向从库发送 binlog 是异步的，从库恢复数据也是异步的
5.读写分离适用与读远大于写的场景，如果只有一台服务器，当 select 很多时，update 和 delete 会被这些 select 访问中的数据堵塞，等待 select 结束，并发性能不高。 对于写和读比例相近的应用，应该部署双主相互复制

6.可以在从库启动是增加一些参数来提高其读的性能，例如--skip-innodb、--skip-bdb、--low-priority-updates 以及--delay-key-write=ALL。当然这些设置也是需要根据具体业务需求来定得，不一定能用上

7.分摊读取。假如我们有 1 主 3 从，不考虑上述 1 中提到的从库单方面设置，假设现在 1 分钟内有 10 条写入，150 条读取。那么，1 主 3 从相当于共计 40 条写入，而读取总数没变，因此平均下来每台服务器承担了 10 条写入和 50 条读取（主库不承担读取操作）。因此，虽然写入没变，但是读取大大分摊了，提高了系统性能。另外，当读取被分摊后，又间接提高了写入的性能。所以，总体性能提高了，说白了就是拿机器和带宽换性能。MySQL 官方文档中有相关演算公式：官方文档 见 6.9FAQ 之“MySQL 复制能够何时和多大程度提高系统性能”

8.MySQL 复制另外一大功能是增加冗余，提高可用性，当一台数据库服务器宕机后能通过调整另外一台从库来以最快的速度恢复服务，因此不能光看性能，也就是说 1 主 1 从也是可以的。

53.构建主从同步的思路?

1. 确保数据相同 - 从库必须要有主库上的数据。
2. 配置主服务器 - 启用 binlog 日志及设置格式，设置 server_id, 授权用户

3. 配置从服务器 - 设置 server_id ，指定主数据库服务器信息
4. 测试配置 - 客户端连接主库，写入的数据，在连接从库的时候也能够访问到。

54.复制模式都有哪些？

步复制（ Asynchronous replication ） - 主库在执行完客户端提交的事务后会立即将结果返给 客户端，并不关心从库是否已经接收并处理。

全同步复制（ Fully synchronous replication ） - 当主库执行完一个事务，所有的从库都执行了该事务 才返回给客户端。

半同步复制（ Semisynchronous replication ） - 介于异步复制和全同步复制之间，主库在执行完客户 端提交的事务后不是立刻返回给客户端，而是等待至 少一个从库接收到并写到 relay log 中才返回给客户端

55.MySQL 主从复制原理？

复制的基本过程如下：

1)、Slave 上面的 IO 进程连接上 Master，并请求从指定日志文件的指定位置（或者从最开始的日志）之后的日志内容；

2)、Master 接收到来自 Slave 的 IO 进程的请求后，通过负责复制的 IO 进程根据请求信息读取制定日志指定位置之后的日志信息，返回给 Slave 的 IO 进程。返回信息中除了日志所包含的信息之外，还包括本次返回的信息已经到 Master 端的 bin-log 文件的名称以及 bin-log 的位置；

3)、Slave 的 IO 进程接收到信息后，将接收到的日志内容依次添加到 Slave 端的 relay-log 文件的最末端，并将读取到的 Master 端的 bin-log 的文件名和位置记录到 master-info 文件中，以便在下一次读取的时候能够清楚的告诉 Master“我需要从某个 bin-log 的哪个位置开始往后的日志内容，请发给我”；

4)、Slave 的 Sql 进程检测到 relay-log 中新增加了内容后，会马上解析 relay-log 的内容成为在 Master 端真实执行时候的那些可执行的内容，并在自身执行。

Mysql 为了解决这个风险并提高复制的性能，将 Slave 端的复制改为两个进程来完成。提出这个改进方案的人是 Yahoo!的一位工程师“Jeremy Zawodny”。这样既解决了性能问题，又缩短了异步的延时时间，同时也减少了可能存在的数据丢失量。当然，即使是换成了现在这样两个线程处理以后，同样也还是存在 slave 数据延时以及数据丢失的可能性的，毕竟这个复制是异步的。只要数据的更改不是在一个事物中，这些问题都是会存在的。如果要完全避免这些问题，就只能用 mysql 的 cluster 来解决了。不过 mysql 的 cluster 是内存数据库的解决方案，需要将所有数据都 load 到内存中，这样就对内存的要求就非常大了，对于一般的应用来说可实施性不是太大。

复制常用架构

Mysql 复制环境 90%以上都是一个 Master 带一个或者多个 Slave 的架构模式，主要用于读压力比较大的应用的数据库端廉价扩展解决方案。因为只要 master 和 slave 的压力不是太大（尤其是 slave 端压力）的话，异步复制的延时一般都很少很少。尤其是自 slave 端的复制方式改成两个进程处理之后，更是减小了 slave 端的延时。而带来的效益是，对于数据实时性要求不是特别的敏感度的应用，只需要通过廉价的 pc server 来扩展 slave 的数量，将读压力分散到多台 slave 的机器上面，即可解决数据库端的读压力瓶颈。这在很大程度上解决了目前很多中小型网站的数据库压力瓶颈问题，甚至有些大型网站也在使用类似方案解决数据库瓶颈。

56.什么是多实例 ，为什么要使用多实例

在一台物理主机上运行多个数据库服务就叫作多实例。

节约运维成本 - 提高硬件利用率

57.多实例配置步骤?

- 安装支持多实例服务的软件包
- 修改主配置文件
- 根据配置文件做相应设置
- 初始化授权库
- 启动服务
- 客户端访问

58.MySQL 性能调优的作用?

提高 MySQL 系统的性能、响应速度

- 替换有问题的硬件（ CPU/ 磁盘 / 内存等）
- 服务程序的运行参数调整
- 对 SQL 查询进行优化

59.介绍下 MHA 集群?

- 由日本 DeNA 公司 youshimaton （现就职于 Facebo ok 公司）开发
- 是一套优秀的作为 MySQL 高可用性环境下故障切换和主从提升的高可用软件。
- 目前在 MySQL 高可用方面是一个相对成熟的解决方案 。
- 在 MySQL 故障切换过程中，MHA 能做到在 0~30 秒之内自动完成数据库的故障切换操作
- 并且在进行故障切换的过程中，MHA 能在最大程度上保证数据的一致性，以达到真正意义上的高可用。

60.MHA 的组成由哪些?

MHA Manager （管理节点）

- 可以单独部署在一台独立的机器上管理多个 master-sl ave 集群，也可以部署在一台 slave 节点上。

• MHA Node （数据节点）

- 运行在每台 MySQL 服务器上。

61.MHA 的工作过程?

MHA Manager 会定时探测集群中的 master 节点， 当 master 出现故障时，它可以自动将最新数据的 sl ave 提升为新的 master ，然后将所有其他的 slave 重新指向新的 master 。整个故障转移过程对应用程 序完全透明。

- （ 1 ）从宕机崩溃的 master 保存二进制日志事件（ binlog events ）
- （ 2 ）识别含有最新更新的 slave
- （ 3 ）应用差异的中继日志（ relay log ）到其他的 slave
- （ 4 ）应用从 master 保存的二进制日志事件（ binlog events ）

- （5）提升一个 slave 为新的 master ；
- （6）使其他的 slave 连接新的 master 进行复制；

62. 什么是视图（View）

- 虚拟表 - 内容与真实的表相似，包含一系列带有名称的列和行 数据。
- 视图并不在数据库中以存储的数据的形式存在。
- 行和列的数据来自定义视图时查询所引用的基表，并 且在具体引用视图时动态生成。
- 更新视图的数据，就是更新基表的数据
- 更新基表数据，视图的数据也会跟着改变

63. 视图优点有哪些？

- 简单
  - 使用视图的用户完全不需要关心视图中的数据是通过 什么查询得到的。
  - 视图中的数据对用户来说已经是过滤好的符合条件的 结果集。
- 安全
  - 用户只能看到视图中的数据。
- 数据独立
  - 一旦视图的结构确定了，可以屏蔽表结构变化对用户 的影响。

64. 使用视图的限制有哪些？

- 不能在视图上创建索引
- 在视图的 FROM 子句中不能使用子查询
- 以下情形中的视图是不可更新的
  - 包含以下关键字的 SQL 语句：聚合函数（SUM 、 MIN 、 MAX 、 COUNT 等 ）、DISTINCT 、 GROUP BY 、 HAVING 、 UNION 或 UNION ALL
  - 常量视图
  - JOIN
  - FROM 一个不能更新的视图
  - WHERE 子句的子查询引用了 FROM 子句中的表
  - 使用了临时表，视图是不可更新

65. 视图的语法格式是什么？

- create view 视图名称 as SQL 查询;
- create view 视图名称 （字段名列表） as SQL 查 询;

66. 使用试图的语法格式？

- 查询记录
  - Select 字段名列表 from 视图名 where 条件;
- 插入记录
  - Insert into 视图名 (字段名列表) values(字段值列 表) ;
- 更新记录
  - Update 视图名 set 字段名 = 值 where 条件;
- 删除记录
  - Delete from 视图名 where 条件;

注意：对视图操作即是对基本操作，反之亦然！！！

删除视图
- 语法格式
  - drop view 视图名;

67. 什么存储过程?
- 数据库中保存的一系列 sql 命令的集合
- 编写存储过程时，可以使用变量、条件判断、流程控 制等
- 存储过程，就是 MySQL 中的脚本

68. 存储过程优点?
- 提高性能
- 可减轻网络负担
- 可以防止对表的直接访问
- 避免重复的 sql 操作

69. 存储过程变量类型?

会话变量 会话变量和全局变量叫系统变量 使用 set 命令定 义;

全局变量的修改会影响到整个服务器，但是对会 话变量的修改，只会影响到当前的会话。 全局变量

用户变量 在客户端连接到数据库服务的整个过程中都是有 效的。当当前连接断开后所有用户变量失效。 定义 set @ 变量名 = 值; 输出 select @ 变量名;

局部变量 存储过程中的 begin/end 。其有效范围仅限于该 语句块中，语句块执行完毕后，变量失效。 declare 专门用来定义局部变量。

70. 什么是分库分表?
- 通过某种特定条件，将存放在一个数据库（ 主机 ）中的 数据，分散存放到多个数据库（主机 ）中。
- 已达到分散单台设备负载的效果，即分库分表
- 数据的切分根据其切分规则的类型，分为 2 种切分模 式 - 垂直分割（ 纵向 ）和 水平分割（ 横向 ）

71. 什么是垂直分割?
- 纵向切分
  - 把单一的表，拆分成多个表，并分散到不同的数据库（ 主机 ）上。
  - 一个数据库由多个表构成，每个表对应不同的业务， 可以按照业务对表进行分类，将其分布到不同的数据 库（ 主机 ）上，实现专库专用，让不同的库（ 主机 ）分 担不同的业务。

72. 什么是水平分割?
- 横向切分
  - 按照表中某个字段的某种规则，把向表中写入的记录 分散到多个库（ 主机 ）中。
  - 简单来说，就是按照数据行切分，将表中的某些行存 储到指定的数据库（ 主机 ）中

73.Mycat 介绍?

Mycat 是基于 Java 的分布式数据库系统中间层，为 高并发下的分布式提供解决方案

- 支持 JDBC 形式连接
- 支持 MySQL 、 Oracle 、 Sqlserver 、 Mongodb 等
- 提供数据读写分离服务
- 可以实现数据库服务器的高可用
- 提供数据分片服务
- 基于阿里巴巴 Cobar 进行研发的开源软件
- 适合数据大量写入数据的存储需求

74.mycat 服务提供 10 种分片规则?

- 1 枚举法 sharding-by-intfile
- 2 固定分片 rule1
- 3 范围约定 auto-sharding-long
- 4 求模法 mod-long
- 5 日期列分区法 sharding-by-date
- 6 通配取模 sharding-by-pattern
- 7ASCII 码求模通配 sharding-by-prefixpattern
- 8 编程指定 sharding-by-substring
- 9 字符串拆分 hash 解析 sharding-by-stringhash
- 10 一致性 hash sharding-by-murmur

75.mycat 工作工程?

当 Mycat 收到一个 SQL 时

- 会先解析这个 SQL 查找涉及到的表，然后看此表的定 义
- 如果有分片规则，则获取到 SQL 里分片字段的值，并 匹配分片函数，得到该 QL 对应的分片列表
- 然后将 SQL 发往这些分片去执行，最后收集和处理所 有分片返回的结果数据，并输出到客户端 以 select * from Orders where prov=? 语句为例，查到 prov=wuhan ， 按照分片函数， wuhan 返回 dn1 ，于是 SQL 就发给了 MySQL1 ，去 取 DB1 上的查询结果，并返回给用户。 如果上述 SQL 改为 elect * from Orders where prov in (‘wuhan’,‘beijing’) ，那么， SQL 就会发给 ySQL1 与 MySQL2 去执行， 然后结果集合并后输出给用户。但通常业务中我们的 SQL 会有 Order By 以及 Limit 翻页语法，此时就涉及到结果集在 Mycat 端的二次处理。

76.什么是 RDBMS?

关系数据库管理系统

- Relational Database Management System
- 数据按照预先设置的组织结构，存储在物理存储介质 上。
- 数据之间可以做关联操作

77.什么是 nosql?

• NoSQL(NoSQL = Not Only SQL)

- 意思是 ″ 不仅仅是 SQL “
- 泛指非关系型数据库
- 不需要预先定义数据存储结构

- 表的每条记录都可以有不同的类型和结构

78. NoSQL 服务软件主流软件

- Redis
- MongoDB
- Memcached
- CouchDB
- Neo4j
- FlockDB

79. Redis 特点是什么?

- 支持数据持久化，可以把内存里数据保存到硬盘中
- 不仅仅支持 key/values 类型的数据，同时还支持 list hash set zset 类型
- 支持 master-salve 模式数据备份

80. Redis 介绍

- Remode DIctionary Server(远程字典服务器)
- 使用 C 语言编写的，遵守 BSD 的开源软件
- 是一款高性能的 (Key/Values) 分布式内存数据库
- 并支持数据持久化的 NoSQL 数据库服务软件
- 中文网站 www.redis.cn

81. redis 常用操作指令

- Set keyname keyvalue 存储
- get keyname 获取
- Select 数据库编号 0-15 切换库
- Keys * 打印所以变量
- Keys a? 打印指定变量
- Exits keyname 测试是否存在
- ttl keyname 查看生存时间
- type keyname 查看类型

82. redis 主从复制工作原理

• 工作原理

- Slave 向 maste 发送 sync 命令
- Master 启动后台存盘进程，同时收集所有修改数据命 令
- Master 执行完后台存盘进程后，传送整个数据文件到 slave 。
- Slave 接收数据文件后，将其存盘并加载到内存中完成 首次完全同步
- 后续有新数据产生时， master 继续将新的所以收集到 的修改命令依次传给 slave ，完成同步。

83. redis 主从复制缺点

- 网络繁忙，会产生数据同步延时问题
- 系统繁忙，会产生数据同步延时问题

84.RDB 介绍

- 全称 Reids DataBase
  - 数据持久化方式之一
  - 在指定时间间隔内，将内存中的数据集快照写入硬盘 。
  - 术语叫 Snapshot 快照。
  - 恢复时，将快照文件直接读到内存里。

85.RDB 优点缺点有哪些？

- RDB 优点
  - 持久化时， Redis 服务会创建一个子进程来进行持久 化，会先将数据写入到一个临时文件中，待持久化过 程都结束了，再用这个临时文件替换上次持久化好的 文件；整个过程中主进程不做任何 IO 操作，这就确保 了极高的性能。 -
  如果要进程大规模数据恢复，且对数据完整行要求不 是非常高，使用 RDB 比 AOF 更高效。
- RDB 的缺点
  - 意外宕机，最后一次持久化的数据会丢失。

86.AOF 介绍

- 只追加操作的文件
  - Append Only File
  - 记录 redis 服务所有写操作。
  - 不断的将新的写操作，追加到文件的末尾。
  - 使用 cat 命令可以查看文件内容

87.AOF 优点 / 缺点

- RDB 优点
  - 可以灵活的设置同步持久化 appendfsync alwayls 或 异步持久化 appendfsync verysec
  - 宕机时，仅可能丢失 1 秒的数据
- RDB 的缺点 -
  AOF 文件的体积通常会大于 RDB 文件的体积。执行 fs ync 策略时的速度可能会比 RDB 慢。

88.MongoDB 软件介绍？

介于关系数据库和非关系数据库之间的产品

- 一个基于分布式文件存储的数据库。
- 由 C++ 语言编写。旨在为 WEB 应用提供可扩展的高 性能数据存储解决方案。
- MongoDB 将数据存储为一个文档，数据结构由键值 (key=>value) 对组成。
- MongoDB 文档类似于 JSON 对象。字段值可以包 含其他文档，数组及文档数组。

89.MongoDB 软件特点 ？

- 安装简单
- 面向文档存储，操作比较简单容易
- 支持丰富的查询表达
- 可以设置任何属性的索引
- 支持主流编程语言 RUBY|PYTHON|JAVA|PHP|C++
- 支持副本集，分片

90.MongoDB 常用管理命令？

- 库管理命令： 查看 创建 切换 删除

- show dbs 查看已有的库
- db 显示当前所在的库
- use 库名 切换库，若库不存在的话 自动延时创建库
- show collections 或 show tables 查看库下已有的 集合
- db.dropDatabase() 删除当前所在的库

数据库名可以是满足以下条件的任意 UTF-8 字符串。
- 不能是空字符串（""）。
- 不得含有’’（空格）、 .、 $、 /、 \ 和 \0（空字符）。
- 应全部小写。
- 最多 64 字节。

91.集合管理命有哪些？

查看 创建 删除
- show collections 或 show tables # 查看集合
- db. 集合名 .drop() # 删除集合
- db. 集合名 .save({'','’}) # 创建集合，集合不存 在时，创建并添加文档

92.MongoDB 文档基本管理？

文档管理命令：
查看 统计 添加 删除
- db. 集合名 .find()
- db. 集合名 .count()
- db. 集合名 .insert({“name”:” jim”})
- db. 集合名 .find({ 条件 })
- db. 集合名 .findOne() # 返回一条文档
- db. 集合名 .remove({}) # 删除所有文档
- db. 集合名 .remove({ 条件 }) # 删除与条件匹配的所有文 档

93.MongoDB 基本数据类型 ？

字符 string/ 布尔 bool/ 空 null
- 字符串 string
  - UTF-8 字符串都可以表示为字符串类型的数据
  - {name:” 张三” } 或 { school:“tarena”}
- 布尔 bool
  - 布尔类型有两个值 true 和 false ， {x:true}
- 空 null
  - 用于表示空值或者不存在的字段， {x:null}

数值 / 数组 array
- 数值
  - shell 默认使用 64 为浮点型数值。 {x ： 3.14} 或 {x ： 3} 。
  - NumberInt （ 4 字节整数） {x:NumberInt(3)}
  - NumberLong （ 8 字节整数） {x:NumberLong(3)}
- 数组 array
  - 数据列表或数据集可以表示为数组

- {x ： [“a“ ， “ b”,” c”]}

代码 / 日期 / 对象
• 代码
- 查询和文档中可以包括任何 JavaScript 代码
- {x: function( ){/* 代码 */}}
• 日期
- 日期被存储为自新纪元依赖经过的毫秒数，不存储时区
- {x:new Date( )}
• 对象
- 对象 id 是一个 12 字节的字符串，是文档的唯一标识
- {x: ObjectId() }

内嵌 / 正则表达式
• 内嵌
- 文档可以嵌套其他文档，被嵌套的文档作为值来处理
- {tarena: {address:“Beijing”,tel:“888888”,perso n:”hanshaoyun” - }}
• 正则表达式
- 查询时，使用正则表达式作为限定条件
- {x:/ 正则表达式 /}

94 MongoDB 数据导出？

• 语法格式 1 - #mongoexport [--host IP 地址 --port 端口] -d 库名 -c 集合名 -f 字段名 1, 字段名 2 --type=csv > 目录名 / 文件名 .csv
• 语法格式 2 - #mongoexport --host IP 地址 --port 端口 - 库名 -c 集合名 -q ‘{ 条件 }’ -f 字段名 1 ，字 段名 2 --type=csv > 目录名 / 文件名 .csv
• 语法格式 3 #mongoexport [--host IP 地址 --port 端口] -d 库名 -c 集合名 [-q ‘{ 条件 }’ - f 字段列表] --type=json > 目录名 / 文件名 .json

95.MongoDB 数据导入？

• 语法格式 1 - #mongoimport - host IP 地址 - port 端口 -d 库名 - c 集合名 --type=json 目录名 / 文件名 .json
• 语法格式 2 - #mongoimport - host IP 地址 - port 端口 -d 库名 - c 集合名 --type=csv --headerline [--drop] 目录名 / 文件名 .c sv

96.数据备份

• 备份数据所有库到当前目录下的 dump 目录下
mongodump [--host ip 地址 --port 端口]
• 备份时指定备份的库和备份目录
mongodump [--host ip 地址 --port 端口] -d 数据库名 -c 集合名 -o 目录
目录无需事先创建 备份时指定即可！！！
• 查看 bson 文件内容 #bsondump ./dump/bbs/t1.bson

97.副本集是什么?

- MongoDB 复制是将数据同步在多个服务器的过程。
- 复制提供了数据的冗余备份，并在多个服务器上存储 数据副本，提高了数据的可用性， 并可以保证数据的 安全性。

- 复制还允许您从硬件故障和服务中断中恢复数据

98. 副本集工作过程?

- mongodb 的复制至少需要两个节点。其中一个是主 节点，负责处理客户端请求，其余的都是从节点，负 责复制主节点上的数据。
- mongodb 各个节点常见的搭配方式为：一主一从、 一主多从。
- 主节点记录在其上的所有操作 oplog ，从节点定期轮 询主节点获取这些操作，然后对自己的数据副本执行 这些操作，从而保证从节点的数据与主节点一致。

##
##

1. 存储的目标的是什么？

- 存储是根据不同的应用环境通过采取合理、安全、有效
  的方式将数据保存到某些介质上并能保证有效的访问
- 一方面它是数据临时或长期驻留的物理媒介
- 另一方面，它是保证数据完整安全存放的方式或行为
- 存储就是把这两个方面结合起来，向客户提供一套数据 存放解决方案

2. 存储技术分类？

- SCSI 小型计算机系统接口
- DAS 直连式存储
- NAS 网络技术存储
- SAN 存储区域网络
- FC 光纤通道

3. SCSI 技术

- Small Computer System Interface 的简称
- 作为输入/输出接口
- 主要用于硬盘、光盘、磁带机等设备

4. DAS 技术

- Direct-Attached Storage 的简称
- 将存储设备通过 SCSI 接口或光纤通道直接连接到计算机上
- 不能实现数据与其他主机的共享
- 占用服务器操作系统资源，如 CPU、IO 等
- 数据量越大，性能越差

5. NAS 技术

- Network-Attached Storage 的简称
- 一种专用数据存储服务器，以数据为中心，将存储设 备与服务器彻底分离，集中管理数据，
  从而释放带宽、 提高性能、降低总拥有成本、保护投资
- 用户通过 TCP/IP 协议访问数据
  - 采用标准的 NFS/HTTTP/CIFS 等

6. SAN 技术

- Storage Area Network 的简称
  - 通过光纤交换机、光纤路由器、光纤集线器等设备将 磁盘阵列、磁带等存储设备与相关服务器连接起来， 形成高速专网网络
- 组成部分
  - 如路由器、光纤交换机
  - 接口：如 SCSI、FC – 通信协议：如 IP、SCSI

7. Fibre Channel

- 一种适合于千兆数据传输的、成熟而安全解决方案
- 与传统的 SCSI 相比，FC 提供更高的数据传输速率、更 远的传输距离、更多的设备连接支持以及更稳定的性 能、更简易的安装

FC 主要组件

- 光纤
- HBA（主机总线适配置器）
- FC 交换机

8. iSCSI 技术的优势？

- 基于 IP 协议技术的标准
- 允许网络在 TCP/IP 协议上传输 SCSI 命令
- 相对 FC SAN，iSCSI 实现的 IP SAN 投资更低
- 解决了传输效率、存储容量、兼容性、开放性、安全 性等方面的问题
- 没有距离限制

9. iSCSI 操作流程是什么？

Target 端

- 选择 target 名称
- 安装 iSCSI target
- 准备用于 target 的存储
- 配置 target – 启用服务

- Initiator 端
  - 安装 initiator
  - 配置 initiator 并启动服务

10. udev 的作用是什么？

- 从内核收到添加/移除硬件事件时，udev 将会分析：
  - /sys 目录下信息
  - /etc/udev/rules.d 目录中的规则
- 基于分析结果，udev 会：
  - 处理设备命名
  - 决定要创建哪些设备文件或链接
  - 决定如何设置属性
  - 决定触发哪些事件

11 文件系统的类型有那些？

- 本地文件系统 – EXT3/4、SWAP、NTFS、……
- 伪文件系统 – /proc、/sys、……
- 网络文件系统 – NFS（Network File System）

12.什么是多路径？

- 当服务器到某一存储设备有多条路径时，每条路径都 会识别为一个单独的设备
- 多路径允许您将服务器节点和储存阵列间的多个 I/O 路径配置为一个单一设备
- 这些 I/O 路径是可包含独立电缆、交换器和控制器的 实体 SAN 链接
- 多路径集合了 I/O 路径，并生成由这些集合路径组成 的新设备

13. 多路径主要功能

- 冗余 – 主备模式，高可用
- 改进的性能 – 主主模式，负载均衡

14. 什么是集群？

- 一组通过高速网络互联的计算组，并以单一系统的模 式加以管理
- 将很多服务器集中起来一起，提供同一种服务，在客 户端看来就象是只有一个服务器
- 可以在付出较低成本的情况下获得在性能、可靠性、 灵活性方面的相对较高的收益
- 任务调度是集群系统中的核心技术

15. 集群的目的是什么？

- 提高性能
  - 如计算密集型应用，如：天气预报、核试验模拟
- 降低成本
  - 相对百万美元级的超级计算机，价格便宜
- 提高可扩展性
  - 只要增加集群节点即可
- 增强可靠性
  - 多个节点完成相同功能，避免单点失败

16. 集群分类有那些？

- 高性能计算集群 HPC
  - 通过以集群开发的并行应用程序，解决复杂的科学问题
- 负载均衡（LB）集群
  - 客户端负载在计算机集群中尽可能平均分摊
- 高可用（HA）集群
  - 避免单点故障，当一个系统发生故障时，可以快速迁移

17. LVS 介绍？

- Linux 虚拟服务器（LVS）是章文嵩在国防科技大学 就读博士期间创建的
- LVS 可以实现高可用的、可伸缩的 Web、Mail、 Cache 和 Media 等网络服务
- 最终目标是利用 Linux 操作系统和 LVS 集群软件实现一 个高可用、高性能、低成本的服务器应用集群

18. LVS 集群的组成？

- 前端：负载均衡层
  - 由一台或多台负载调度器构成
- 中间：服务器群组层
  - 由一组实际运行应用服务的服务器组成
- 底端：数据共享存储层
  - 提供共享存储空间的存储区域

19. LVS 术语

- Director Server：调度服务器
  - 将负载分发到 Real Server 的服务器
- Real Server：真实服务器
  - 真正提供应用服务的服务器
- VIP：虚拟 IP 地址
  - 公布给用户访问的虚拟 IP 地址
- RIP：真实 IP 地址
  - 集群节点上使用的 IP 地址
- DIP：调度器连接节点服务器的 IP 地址

20. LVS 工作模式

- VS/NAT – 通过网络地址转换实现的虚拟服务器
  - 大并发访问时，调度器的性能成为瓶颈
- VS/DR – 直接使用路由技术实现虚拟服务器
  - 节点服务器需要配置 VIP，注意 MAC 地址广播
- VS/TUN
  - 通过隧道方式实现虚拟服务器

21. 负载均衡调度算法有哪些？

- 轮询（Round Robin）
- 加权轮询（Weighted Round Robin）
- 最少连接（Least Connections）
- 加权最少连接（ Weighted Least Connections ）
  - **基于局部性的最少链接**
  - **带复制的基于局部性最少链接**
  - **目标地址散列**（Destination Hashing）
  - **最短的期望的延迟**
  - **最少队列调度**
  - **源地址散列**（Source Hashing）

22. HAProxy 介绍？

- 它是免费、快速并且可靠的一种解决方案
- 适用于那些负载特大的 web 站点，这些站点通常又需 要会话保持或七层处理
- 提供高可用性、负载均衡以及基于 TCP 和 HTTP 应用 的代理

23. 衡量负责均衡器性能的因素

- Session rate 会话率
  - 每秒钟产生的会话数
- Session concurrency 并发会话数
  - 服务器处理会话的时间越长，并发会话数越多
- Data rate 数据速率
  - 以 MB/s 或 Mbps 衡量
  - 大的对象导致并发会话数增加
  - 高会话数、高数据速率要求更多的内存

24. HAProxy 工作模式？

- mode http
  - 客户端请求被深度分析后再发往服务器
- mode tcp
  - 客户端与服务器之间建立会话，不检查第七层信息
- mode health
  - 仅做健康状态检查，已经不建议使用

25. HTTP 事务模型？

- HTTP 协议是事务驱动的
- 每个请求（Request）仅能对应一个响应（Response）
- 常见模型：
  - HTTP close
  - Keep-alive
  - Pipelining

- HTTP close
  - 客户端向服务器建立一个 TCP 连接
  - 客户端发送请求给服务器
  - 服务器响应客户端请求后即断开连接
  - 如果客户端到服务器的请求不只一个，那么就要不断的 去建立连接
  - TCP 三次握手消耗相对较大的系统资源，同时延迟较大

- Keep-alive
  - 一次连接可以传输多个请求
  - 客户端需要知道传输内容的长度，以避免无限期的等 待传输结束
  - 降低两个 HTTP 事务间的延迟 - 需要相对较少的服务器资源

- Pipelining
  - 仍然使用 Keep-alive
  - 在发送后续请求前，不用等前面的请求已经得到回应
  - 适用于有大量图片的页面
  - 降低了多次请求之间的网络延迟

26. Keepalived 概述？

- 调度器出现单点故障，如何解决？
- Keepalived 实现了高可用集群
- Keepalived 最初是为 LVS 设计的，专门监控各服务器 节点的状态
- Keepalived 后来加入了 VRRP 功能，防止单点故障

27. Keepalived 运行原理？

- Keepalived 检测每个服务器节点状态
- 服务器节点异常或工作出现故障， Keepalived 将故障节 点从集群系统中剔除
- 故障节点恢复后，Keepalived 再将其加入到集群系统中
- 所有工作自动完成，无需人工干预

28. 说一下 Keepalived+LVS 拓扑？

- 使用 Keepalived 高可用解决调度器单点失败问题
- 主、备调度器上配置 LVS
- 主调度器异常时，Keepalived 启用备用调度器调度器

29. 分析 Nginx 优缺点？

- 优点
  - 工作在 7 层，可以针对 http 做分流策略
  - 正则表达式比 HAProxy 强大
  - 安装、配置、测试简单，通过日志可以解决多数问题
  - 并发量可以达到几万次 – Nginx 还可以作为 Web 服务器使用
- 缺点
  - 仅支持 http、https、mail 协议，应用面小
  - 监控检查仅通过端口，无法使用 url 检查

30. 分析 LVS 优缺点？

- 优点
  - 负载能力强，工作在 4 层，对内存、CPU 消耗低
  - 配置性低，没有太多可配置性，减少人为错误
  - 应用面广，几乎可以为所有应用提供负载均衡
- 缺点
  - 不支持正则表达式，不能实现动静分离
  - 如果网站架构庞大，LVS-DR 配置比较繁琐

31. 分析 HAProxy 优缺点？

- 优点
  - 支持 session、cookie 功能
  - 可以通过 url 进行健康检查
  - 效率、负载均衡速度，高于 Nginx，低于 LVS – HAProxy 支持 TCP，可以对 MySQL 进行负载均衡
  - 调度算法丰富
- 缺点
  - 正则弱于 Nginx
  - 日志依赖于 syslogd，不支持 apache 日志

32. 什么是分布式文件系统？

- 分布式文件系统（Distributed File System）是指文 件系统管理的物理存储资源不一定直接连接在本地节 点上，而是通过计算机网络与节点相连
- 分布式文件系统的设计基于客户机/服务器模式

33. 常用分布式文件系统有哪些？

- Lustre
- Hadoop
- FastDFS
- Ceph
- GlusterFS

34. 什么是 ceph ？

- ceph 是一个分布式文件系统
- 具有高扩展、高可用、高性能的特点
- ceph 可以提供对象存储、块存储、文件系统存储
- ceph 可以提供 PB 级别的存储空间(PBàTBàGB) – 1024G*1024G=1048576G
- 软件定义存储(Software Defined Storage)作为存储 行业的一大发展趋势，已经越来越受到市场的认可

35. ceph 组件有哪些？

- OSDs – 存储设备
- Monitors – 集群监控组件
- MDSs – 存放文件系统的元数据（对象存储和块存储不需要该组件）
- Client – ceph 客户端

36. 什么是 CephFS？

- 分布式文件系统（Distributed File System）是指文 件系统管理的物理存储资源不一定直接连接在本地节 点上，而是通过计算机网络与节点相连
- CephFS 使用 Ceph 集群提供与 POSIX 兼容的文件系统
- 允许 Linux 直接将 Ceph 存储 mount 到本地

37. 什么是元数据？

- 元数据（Metadata）
  - 任何文件系统中的数据分为数据和元数据。
  - 数据是指普通文件中的实际数据
  - 而元数据指用来描述一个文件的特征的系统数据
  - 比如：访问权限、文件拥有者以及文件数据块的分布信息 (inode...)等

38. 什么是对象存储？

- 对象存储
  - 也就是键值存储，通其接口指令，也就是简单的 GET、 PUT、DEL 和其他扩展，向存储服务上传下载数据
  - 对象存储中所有数据都被认为是一个对象，所以，任 何数据都可以存入对象存储服务器，如图片、视频、 音频等
- RGW 全称是 Rados Gateway
- RGW 是 Ceph 对象存储网关，用于向客户端应用呈现 存储界面，提供 RESTful API 访问接口