云计算复习

~~大数据4V+1C的特征：~~

1. 数据量大
2. 多样
3. 快速
4. 价值密度低
5. 复杂度

云计算定义：

长定义：云计算是一种商业计算模型，它将计算任务分布在大量计算机构成的资源池上，使各种应用系统能够根据需要获取计算力、存储空间和信息服务。
短定义：云计算是通过网络按需提供可动态伸缩的廉价计算服务。

云计算特点：

1. 特大规模
2. 虚拟化
3. 高可靠性
4. 通用性
5. 高可伸缩性
6. 按需服务
7. 极其廉价

云计算服务类型！！！

1. Saas Software as a Service ： CRM,电子商务服务FPS，DevPay,网站访问统计服务Alexa Web服务
2. PaaS Platform as a Service : Google App Engine,弹性MapReduce，简单数据库服务SimpleDB
3. IaaS Infrastructure as a Service ： EC2,S3,

~~云计算技术体系结构：~~

1. 物理资源层
2. 资源池层
3. 管理中间层
4. SOA构建层

在性价比上云计算相比传统技术为什么有压倒性的优势：！！！！

。~~使用云计算，网络，储存，管理成本更低。~~ 由于云计算有更低的硬件和网络成本、更低管理成本和电力成本，也有更高的资源利用率，两个乘起来能够将成本节省30倍以上，
~~这就是为什么云计算在性价比上相比传统技术有压倒性优势。~~

企业的IT开销主要分为三部分：硬件开销、能耗、管理成本。相比于传统技术，企业使用云计算其网络、存储、管理成本比中型数据中心低，云计算的数据中心可以选择在水电资源丰富、人烟稀少的地方，减少电力、人力成本。除此之外云计算平台提供的是有弹性的服务，它根据每个租用者的的需要来动态分配资源，大大提高了资源的利用率。这两者乘起来能够将成本节省30倍以上。

第二章Google云计算原理与应用：

GFS（分布式文件系统）系统架构

1. Client（客户端）
2. Master（主服务器）
3. Chunk Server （数据库服务器）
   Chunk Server 的个数可以为多个
   文件默认分块 固定大小：64MB 每一块称为一个chunk

GFS的特点：！！！！

1. 采用中央服务器模式
2. 不缓存数据
3. 在用户态实现
4. 只提供专用接口

~~master容错机制：~~

1. 命名空间，也就是整个文件系统的目录结构
2. Chunk与文件名的映射表
3. Chunk副本的位置信息，每一个chunk默认三个副本

~~chunk server容错机制：~~
~~校验读~~

分布式锁服务 Chubby

是提供粗力度锁服务的一个文件系统，它基于松耦合分布式系统，解决了分布的一致性问题。通过chubby的锁服务，用户可以确保数据操作过程中的一致性。
google使用它的用途：
1. 指定GFS主服务器
2. 指定Bigtable主服务器
3. 名字服务
这种锁只是一种建议性的锁而不是强制性的锁，这种选择使系统具有更大的灵活性

Paxos算法

两个阶段：

1. 准备阶段
2. 批准阶段
   三种角色：
3. proposer：提出决议
4. acceptor：批准决议
5. learner：获取并使用已经通过的协议

Bigtable架构（分布式结构化数据表）大概了解

Bigtable 是Google开发的基于GFS和Chubby的分布式存储系统

1. 客户端程序库
2. 一个主服务器
3. 多个子表服务器

分布式存储结构 Megastore 存储技术

统一sql nosql
成功的将关系型数据库和nosql的特点与优势进行了融合。

Megastore 写操作

采用预写式日志,也就是说只有当所有的操作都在日志中记录下后操作才会对数据执行修改。

一个写事务总是开始于一个current读以便确认下一个可用的日志的位置。提交操作将数据变更聚集到日志，接着分配一个比之前任意一个都高的时间戳，然后使用Paxos将数据变更聚集到日志。

这个协议使用了乐观并发：尽管可能有多个写操作同时试图写同一个日志位置，但只会有1个成功。所有失败的写都会观察到成功的写操作，然后中止并重试它们的操作

快速读 快速写

快速读：如果读操作不需要副本之间的通信，那么效率一定高。利用本地读取实现快速读，其实就是保证了所有副本上的数据都一样（paxos 协调者）那是怎么保证的呢?
使用了协调者这个服务。

快速写：如果一次写成功，那么下一次写的时候就跳过准备阶段，直接进入接受阶段。因为一次写成功，那么下一次的写意味着也准确地获知了下一个日志的位置，所以不再需要准备阶段。leaders

协调者什么用?

协调者是一个服务： 该服务分布在每个副本的数据中心里面。
它的主要作用就是跟踪一个实体组集合，集合中的实体组需要具备的条件就是它们的副本都已经观察到了所有的Paxos写。只要出现在这个集合中的实体组，它们的副本就都能够进行本地读写

Dapper（监控系统）的三个重要概念：

大规模分布式系统的监控基础架构Dapper

1. 监控树
2. 区间
3. 注释

监控树：实际上就是一个同特定事件相关的所有消息，只不过这些消息是按照一定的规律以树的形式组织起来。

区间：树中的每一个节点称为一个区间，区间实际上就是一条记录。所有记录联系到一起，就构成了对某个事件的完整监控。每个区间包括以下内容：区间名、区间id、父id、监控id。

注释：注释主要用来辅助推断区间关系，也可以包含一些自定义的内容。
客户端、服务端的注释信息，用户自定义注释：

- 简单的文本方式注释
- 键–值对方式的注释，赋予开发者更多自由

Dapper的关键性技术

轻量级的核心功能库：主要是为了实现对应用层透明。

二次抽样技术：第一次抽样：在1024个请求中抽取1个进行监控
第二次抽样：发生在写入bigtabel之前，具体方法：
将监控id散列成一个标量z，其中0<=z<=1.如果某个区间的z小于实现定好的系数，则保留这个区间并将它写入bigtabel。否则丢弃
解决了低开销及广泛可部署性的问题。

Dapper是干什么用？ 理解（选择题

1. 新服务部署 Dapper
2. 定位长尾延迟
3. 推断服务间的依存关系
4. 确定不同服务的网络使用情况
5. 分层的共享式存储系统
6. 利用Dapper进行“火拼”
   火拼是指：处于危险状态的分布式系统的代表性活动。

正在“火拼”中的Dapper用户需要访问最新的数据却没有时间来编写新的DAPI代码。

Google App Engine （平台层软件

沙盒：可以保证每个应用程序能够安全地隔离运行。
沙盒对用户进行如下限制：。。。

1. 用户的应用程序只能通过Google App Engine 提供的网址抓取API和电子邮件服务API来访问互联网中其他的计算机，并且其他的计算机请求与该应用程序相连接，只能在标准接口上通过HTTP/HTTPS进行。

2. 应用程序无法对Google App Engine的文件系统进行写操作，只能读取应用程序代码上的文件，并且该程序必须使用Google 。。 的Data Store数据库来存储应用程序运行期间持续存在的数据

3. 应用程序只有在响应网络请求时才运行，并且这个响应时间必须极短，请求处理的程序不能在自己的响应发送后产生子进程或执行代码。

简而言之，沙盒给开发人员提供了一个虚拟的环境，这个环境使应用程序与其他开发者开发的程序相隔离，从而保证每个开发者安全地开发自己的程序

第三章Amazon云计算：

Dynamo的架构

是个基础存储架构
完全分布式，去中心化的架构。
只支持键–值方式的数据存储。不支持复杂的查询。
Dynamo的表格：

问题	采用的相关技术
数据均衡分布	改进的一致性哈希算法
数据备份	参数可调的quorum弱机制
数据冲突处理	向量时钟

成员资格及错误检测	基于Gossip协议成员资格和错误检测
临时故障处理	Hinted Handoff（数据回传机制）
永久性故障处理	Merkle哈希树

改进后的一致性哈希算法什么原理！！

基本过程：对于系统中的每个设备节点，为其分配一个随机的标记，这些标记可以构成一个哈希环。在存储数据时，计算出数据中键的哈希值，将其存放到哈希环顺时针方向上第一个标记大于或等于哈希值的设备节点上。

若加一个节点，顺时针的 前面一个节点有数据的话，部分移到新加的节点上面去
若删一个节点，有数据的话，数据移到，顺时针的前面节点上面去。

- **改进后的一致性哈希算法**： 因为每个物理节点的存储能力不同，将物理节点虚拟为一个或多个虚拟节点，分布在哈希环上，各个虚拟节点的能力基本相同。 数据对象先按照哈希值分配到虚拟节点上，并存在该虚拟节点对应的物理节点上。

删、改节点，数据怎么挪动

Dynamo的容错处理：

1. 临时故障处理机制
   数据放到临时空间（最新的节点）然后恢复之后，数据就拿回来
2. 永久性故障处理机制
   从其他副本进行数据备份，采用了Merkle哈希树技术来加快检测和减少数据传输量

N个数据副本采用参数可调的弱quorum的机制

R+W>N
W:成功的写操作至少需要写入的副本数
R：表示一次成功读操作须由服务器返回给用户的最小副本数
N:表示每个数据存储的副本数

EC2的基本架构

Elastic Compute Cloud

1. Amazon机器映像 AMI：包含了操作系统、服务器程序、应用程序等软件配置等模板，用于启动不同的实例。
2. 实例 Instance：由AMI启动，可以分为计算、存储、微型等不同类型的实例
3. 存储模块等组成部分 ：
   弹性存储EBC (Elastic Block Store)进行实例的数据存储。否则实例存储模块中的数据仅仅在实例的生命周期内存在。如果实例出现故障或被终止，数据将会丢失。

公共ip，私有ip，弹性ip

EC2的容错机制，使用一种弹性ip地址：
在创建实例的时候，系统会分配一个公共ip和一个私有ip。
弹性ip和用户的账号进行绑定而不是某个特定的实例。每个账号默认五个弹性ip地址

当系统正在使用的实例出现故障时，用户只需要将弹性ip地址通过网络地址NAT转换为新实例所在对应的私有ip地址，这样就将弹性ip地址与新的实例关联起来。
通过弹性ip地址改变映射关系总可以保证有实例可用。

简单存储服务S3 桶和对象

Simple Storage Services

- 桶：
  桶是存储对象的容器。类似文件夹但是
  桶不可以嵌套，桶不能创建桶。
  amazon限制了用户可以创建的桶数，但是没有限制每个桶里面的对象数。
  桶的命名要求是在整个amazon S3 服务器中是全局唯一的

- 对象：对象是S3的基本储存单元

由数据和**元数据**组成。 数据可以是任意类型但是大小受到对象最大容量限制
元数据是数据内容的附加描述信息，通过名称–值集合的形式来定义。

元数据名称	名称含义
last–modified	对象最后被修改的时间

ETag	利用MD5哈希算法的出对象值
Conten-Type	对象的MIME数据类型
Content—Length	对象数据长度

关系型数据库服务RDS

Relational Database Service
Share-Nothing 架构， 每台数据库服务器都是完全独立的计算机系统
通过网络连接，不共享任何资源
RDS将MYSQL数据库移植到集群，一定范围内解决了关系型数据库的可拓展性问题。
集群MYSQL通过

1. 主从备份
2. 读副本技术
   提高可靠性和数据处理能力
   数据库升级的时候，先升级副本，副本变成主服务器，然后升级另外的，又变成主服务器。

简单队列服务 SQS （Simple Queue Service）

基本组件：

- 系统组件 是SQS的服务对象
- 队列：队列是存放消息的容器，类似桶，队列的数目是任意的，但是创建时必须是SQS账户名称唯一的；会尽量实现“先进先出”
- 消息：是发送者创建的具有一定格式的文本数据，接受对象可以是一个或多个组件，消息的大小是限制的，目前amazon的每一条消息不得超过8kB，消息数未限制

SQS是组件之间沟通的桥梁。

内容推出服务 CloudFront

基于amazon云计算平台实现的内容分发网络。（CDN）

智能的DNS（cdn）负载均衡系统。
CDN技术通过将网站内容发布到靠近用户的边缘节点，使不同地域的用户在访问相同的网页时可以就近获取。这样既可以减轻源服务器的负担，也可以减少整个网络中流量分布不均的情况，进而改善整个网络的性能

- 原理：DNS在对域名进行解析时不再向用户返回网站服务器的ip，而是返回由智能CDN负载均衡系统选定的某个边缘节点ip，用户利用这个ip来访问边缘节点，然后该节点通过其内部的DNS解析出真实网站的ip并发出请求来获取用户所需的网页。

优点：

1. 将网站的流量比较均匀的分散到各个边缘节点，减轻了网站源服务器的负担
2. 由于边缘节点与用户的地理位置进，访问的速度更快
3. 智能DNS负载均衡系统和各个边缘节点之间始终保持着通信，可以确保分配给用户的边缘节点始终可用并且在允许的流量范围内。

OpenStack开源虚拟化平台！！！：

提供一个部署云操作平台或工具集
使用OpenStack易于构建虚拟计算或存储服务的云

- Rackspace公司贡献了“云文件”（Swift）作为 对象存储部分
- 美国宇航局NASA贡献了“星云“（Nova）

OpenStack是管理计算、存储和网络资源的数据中心云计算开放平台，通过一个仪表板，为管理员提供了所有的管理控制，同时通过web界面为其用户提供资源。

- 主要目标：是管理数据中心的资源，简化资源分配
  主要服务成员：

1. 计算服务（Nova）是OpenStack云计算架构的控制器，支持OpenStack云内的实例的生命周期所需的活动由Nova处理，Nova作为管理平台管理着OpenStack云里的计算资源、网络、授权、扩展需求。Nova本身不能提供虚拟化功能，它使用Libvirt的api来支持虚拟机管理程序交互。

2. 对象存储服务（Swift）提供对象存储服务，允许对文件进行存储或者检索。Swift为OpenStack提供了分布式的、最终一致的虚拟对象存储，通过分布式的存储节点，Swift有能力存储数十亿计的对象。具有内置冗余、容错管理、存档、流

媒体的功能。Swift是高度拓展的

3. 镜像服务（Glance）提供一个虚拟磁盘的目录和存储仓库，可以提供对虚拟机镜像的存储和检索；能够进行多个数据中心的镜像管理和租户私有云管理。
4. 身份验证（keystone）为所有服务提供身份验证和授权。
5. 网络管理服务（Quantum）在接口设备之间提供“网络连接即服务”的服务
6. 存储管理服务（Cinder）虚拟机的存储管理
7. 仪表盘（Horizon）演示作用，前端界面。

Libvirt是个什么东西?

Libvirt的主要目标是为各种虚拟化工具提供一套方便、可靠的编程接口，用一种单一的方式管理多种不同的虚拟化提供了方式。

支持的功能：

1. 虚拟机管理
2. 远程机器支持
3. 存储管理
4. 网络接口管理
5. 虚拟NAT和基于路由的网络：

hadoop编程：

第一题： WordCountAndSort !!!!

```
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext

object WordCountAndSort {
  def main(args: Array[String]) {
    val conf = new SparkConf().setAppName("WordCountAndSort").setMaste
```

```
r("local")
    // 创建Spark配置，应用程序的名称   创建Spark程序执行的入口
    //conf.setAppName("wordcount")
    //conf.setMaster("local")
    val sc = new SparkContext(conf)
    val line = sc.textFile(args(1)) //  "/Users/vincent/vincent/learn/
云计算/wordcount.txt"
    val reduce = line.flatMap(_.split(" ")).map((_, 1)).reduceByKey(_
+ _)
    val sort = reduce.sortBy(_._2, false)//false降序;
    val result = sort.take(args(0).toInt) //   5
    for (i <- 0 until result.length)
      println(result(i)._2)
    sort.saveAsTextFile(args(2)) //"/Users/vincent/vincent/output"
    sc.stop()
  }
}
```

第二题：整合文件！！！！

```
public class FileMerge {
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();//获得程序参数 放到othe
rArgs中
        //利用命令行args 进行传参
        //  args[0], args[1], args[2]
//        if (args.length < 2) {//没有指定参数
//            System.err.println("请指定输入、输出路径");
//            System.exit(2);
//        }
        Job job = Job.getInstance(conf, "FileMerge");//设置环境参数
        job.setJarByClass(FileMerge.class);//设置整个程序的类名
        job.setMapperClass(FileMerge.Map.class);//添加Mapper类
        //job.setCombinerClass(FileMerge.Reduce.class);//添加combiner类
        job.setReducerClass(FileMerge.Reduce.class);//添加reducer类
        job.setOutputKeyClass(Text.class);//设置输出类型
        job.setOutputValueClass(Text.class);//设置输出类型
        for (int i = 0; i < args.length - 1; ++i) {
            FileInputFormat.addInputPath(job, new Path(args[i]));//设
置输入文件
```

```
        }

        Path path = new Path(args[args.length-1]);// 取最后一个表示输出目录参数（第0个参数是输入目录）
//        FileSystem fileSystem = path.getFileSystem(conf);// 当前路径path的文件系统
//        if (fileSystem.exists(path)) {//如果存在output文件夹则删除
//            fileSystem.delete(path, true);// true的意思是，就算output有东西，也一带删除
//        }

        FileOutputFormat.setOutputPath(job, new Path(args[args.length - 1]));//设置输出文件
        // 程序完成退出
        System.exit(job.waitForCompletion(true) ? 0 : 1);
        System.exit(job.waitForCompletion(true)?0:1);
    }

    //实现map函数
    public static class Map extends Mapper<Object, Text, Text, Text> {
//继承mapper父类  //接收一个文件名作为key，该文件的每行内容作为value
        //private Text line = new Text();//每一行作为一个数据
        public void map(Object key, Text value, Mapper<Object, Text, Text, Text>.Context context) throws IOException, InterruptedException {
            context.write(value, new Text(""));//把每一行都放到text中 唯一的key实现了去重
        }
    }

    // reduce将输入中的key复制到输出数据的key上，并直接输出，这是数据去重思想
    public static class Reduce extends Reducer<Text, Text, Text, Text>
 {
        public void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {
            context.write(key, new Text("")); //map传给reduce的数据已经做完数据去重，输出即可
        }
    }
}
```

hdfs常用命令：

```
# 1. Print the Hadoop version
hadoop version
# 2. List the contents of the root directory in HDFS
#
hadoop fs -ls /
# 3. Report the amount of space used and
# available on currently mounted filesystem
#
hadoop fs -df hdfs:/
# 4. Count the number of directories,files and bytes under
# the paths that match the specified file pattern
#
hadoop fs -count hdfs:/
# 5. Run a DFS filesystem checking utility
#
hadoop fsck – /
# 6. Run a cluster balancing utility
#
hadoop balancer
# 7. Create a new directory named “hadoop” below the
# /user/training directory in HDFS. Since you’re
# currently logged in with the “training” user ID,
# /user/training is your home directory in HDFS.
#
hadoop fs -mkdir /user/training/hadoop
# 8. Add a sample text file from the local directory
# named “data” to the new directory you created in HDFS
# during the previous step.
#
hadoop fs -put data/sample.txt /user/training/hadoop
# 9. List the contents of this new directory in HDFS.
#
hadoop fs -ls /user/training/hadoop
# 10. Add the entire local directory called “retail” to the
# /user/training directory in HDFS.
#
hadoop fs -put data/retail /user/training/hadoop
# 11. Since /user/training is your home directory in HDFS,
```

```
# any command that does not have an absolute path is
# interpreted as relative to that directory. The next
# command will therefore list your home directory, and
# should show the items you've just added there.
#
hadoop fs -ls
# 12. See how much space this directory occupies in HDFS.
#
hadoop fs -du -s -h hadoop/retail
# 13. Delete a file 'customers' from the "retail" directory.
#
hadoop fs -rm hadoop/retail/customers
# 14. Ensure this file is no longer in HDFS.
#
hadoop fs -ls hadoop/retail/customers
# 15. Delete all files from the "retail" directory using a wildcard.
#
hadoop fs -rm hadoop/retail/*
# 16. To empty the trash
#
hadoop fs -expunge
# 17. Finally, remove the entire retail directory and all
# of its contents in HDFS.
#
hadoop fs -rm -r hadoop/retail
# 18. List the hadoop directory again
#
hadoop fs -ls hadoop
# 19. Add the purchases.txt file from the local directory
# named "/home/training/" to the hadoop directory you created in HDFS
#
hadoop fs -copyFromLocal /home/training/purchases.txt hadoop/
# 20. To view the contents of your text file purchases.txt
# which is present in your hadoop directory.
#
hadoop fs -cat hadoop/purchases.txt
# 21. Add the purchases.txt file from "hadoop" directory which is pres
ent in HDFS directory
# to the directory "data" which is present in your local directory
#
hadoop fs -copyToLocal hadoop/purchases.txt /home/training/data
# 22. cp is used to copy files between directories present in HDFS
```

```
#
hadoop fs -cp /user/training/*.txt /user/training/hadoop
# 23. ‘-get’ command can be used alternaively to ‘-copyToLocal’ comman
d
#
hadoop fs -get hadoop/sample.txt /home/training/
# 24. Display last kilobyte of the file “purchases.txt” to stdout.
#
hadoop fs -tail hadoop/purchases.txt
# 25. Default file permissions are 666 in HDFS
# Use ‘-chmod’ command to change permissions of a file
#
hadoop fs -ls hadoop/purchases.txt
sudo -u hdfs hadoop fs -chmod 600 hadoop/purchases.txt
# 26. Default names of owner and group are training,training
# Use ‘-chown’ to change owner name and group name simultaneously
#
hadoop fs -ls hadoop/purchases.txt
sudo -u hdfs hadoop fs -chown root:root hadoop/purchases.txt
# 27. Default name of group is training
# Use ‘-chgrp’ command to change group name
#
hadoop fs -ls hadoop/purchases.txt
sudo -u hdfs hadoop fs -chgrp training hadoop/purchases.txt
# 28. Move a directory from one location to other
#
hadoop fs -mv hadoop apache_hadoop
# 29. Default replication factor to a file is 3.
# Use ‘-setrep’ command to change replication factor of a file
#
hadoop fs -setrep -w 2 apache_hadoop/sample.txt
# 30. Copy a directory from one node in the cluster to another
# Use ‘-distcp’ command to copy,
# -overwrite option to overwrite in an existing files
# -update command to synchronize both directories
#
hadoop fs -distcp hdfs://namenodeA/apache_hadoop hdfs://namenodeB/hado
op
# 31. Command to make the name node leave safe mode
#
hadoop fs -expunge
sudo -u hdfs hdfs dfsadmin -safemode leave
```

```
# 32. List all the hadoop file system shell commands
#
hadoop fs
# 33. Last but not least, always ask for help!
#
hadoop fs -help
```

重点部分：

GFS有哪些特点？！！！！

性价比高？！！！！！ 3/4

编程题有个难的 编程 2/3

MapReduce的缺点：

- MR算法少，不适合复杂的数据处理过程
- 每次Reduce都需要读写磁盘，速度慢
- MR需要成对出现
- Master节点调度慢，无法实时看到结果
- 迭代数据处理能力较差

copyright@Vincent.