## 试分析和比较B/S，二层C/S和三层C/S，指出各自的优点和缺点。

**二层C/S结构：**将应用一分为二，服务器负责数据管理，客户机完成与用户的交互任务。

**优点：**

C/S 体系结构具有强大的数据操作和事务处理能力，模型思想简单，易于人们理解和接受。

系统的客户应用程序和服务器构件分别运行在不同的计算机上，系统中每台服务器都可以适合各构件的要求，这对于硬件和软件的变化显示出极大的适应性和灵活性，而且易于对系统进行扩充和缩小。

在C/S体系结构中，系统中的功能构件充分隔离，客户应用程序的开发集中于数据的显示和分析，而数据库服务器的开发则集中于数据的管理，不必在每一个新的应用程序中都要对一个DBMS进行编码。将大的应用处理任务分布到许多通过网络连接的低成本计算机上，以节约大量费用

**缺点：**

开发成本较高

客户端程序设计复杂

信息内容和形式单一

用户界面风格不一，使用繁杂，不利于推广使用

软件移植困难

软件维护和升级困难

新技术不能轻易应用

**三层C/S结构：**在二层C/S结构的基础上进行了改造，并增加了一个服务器。

**优点：**

允许合理地划分三层结构的功能，使之在逻辑上保持相对独立性，能提高系统和软件的可维护性和可扩展性。

允许更灵活有效地选用相应的平台和硬件系统，使之在处理负荷能力上与处理特性上分别适应于结构清晰的三层；并且这些平台和各个组成部分可以具有良好的可升级性和开放性。

应用的各层可以并行开发，可以选择各自最适合的开发语言。

利用功能层有效地隔离开表示层与数据层，未授权的用户难以绕过功能层而利用数据库工具或黑客手段去非法地访问数据层，为严格的安全管理奠定了坚实的基础。

**缺点：**

三层C/S结构各层间的通信效率若不高，即使分配给各层的硬件能力很强，其作为整体来说也达不到所要求的性能。

设计时必须慎重考虑三层间的通信方法、通信频度及数据量。这和提高各层的独立性一样是三层C/S结构的关键问题。

**B/S体系结构：**是三层C/S结构的一种实现方式，其具体结构是：浏览器/Web服务器/数据库服务器。

**优点：**

基于B/S体系结构的软件，系统安装、修改和维护全在服务器端解决。用户在使用系统时，仅仅需要一个浏览器就可运行全部的模块，真正达到了“零客户端”的功能，很容易在运行时自动升级。

B/S体系结构还提供了异种机、异种网、异种应用服务的联机、联网、统一服务的最现实的开放性基础。

**缺点：**

B/S体系结构缺乏对动态页面的支持能力，没有集成有效的数据库处理功能。

B/S体系结构的系统扩展能力差，安全性难以控制。

采用B/S体系结构的应用系统，在数据查询等响应速度上，要远远地低于C/S体系结构。 B/S体系结构的数据提交一般以页面为单位，数据的动态交互性不强，不利于在线事务处理应用

## Hadoop原理：HDFS及MapReduce

Hadoop最主要的是Core，它又分为HDFS和MapReduce两个部分，前者提供分布式数据存储，后者提供任务的分发和归拢。其他组件都是围绕着这两个核心进行工作。

**HDFS的设计初衷：**

一次写入多次读取不支持文件并发写入不支持文件修改.

**HDFS的适用范围：**

存储并管理PB级数据处理非结构化数据注重数据处理的吞吐量，对延时不敏感应用模式为一次写入多次读取存取模式.

**HDFS：**

HDFS 采用Master/Slave的架构来存储数据，这种架构主要由四个部分组成，分别为HDFS Client、NameNode、DataNode和Secondary NameNode。下面我们分别介绍这四个组成部分。

**Client：就是客户端。**

1、文件切分。文件上传 HDFS 的时候，Client 将文件切分成 一个一个的Block，然后进行存储。

2、与 NameNode 交互，获取文件的位置信息。

3、与 DataNode 交互，读取或者写入数据。

4、Client 提供一些命令来管理 HDFS，比如启动或者关闭HDFS。

5、Client 可以通过一些命令来访问 HDFS。

**NameNode：就是 master，它是一个主管、管理者。**

1、管理 HDFS 的名称空间。

2、管理数据块（Block）映射信息

3、配置副本策略

4、处理客户端读写请求。

**DataNode：就是Slave。NameNode 下达命令，DataNode 执行实际的操作。**

1、存储实际的数据块。

2、执行数据块的读/写操作。

**Secondary NameNode：并非 NameNode 的热备。当NameNode 挂掉的时候，它并不能马上替换 NameNode 并提供服务。**

1、辅助 NameNode，分担其工作量。

2、定期合并 fsimage和fsedits，并推送给NameNode。

3、在紧急情况下，可辅助恢复 NameNode。

## MapReduce处理过程：

Map-Reduce的目标是在面对节点失效的情况时能够保证大量的文件和数据依然可用。

初始化时数据被分割成许多分块的小数据，这些数据都是以<key,value>的形式存储。用户程序会分配一个master进程和许多worker进程。

任务开始时，Master将用户程序的工作分成两种类型的任务（map 任务和reduce任务），并将这些任务分配给相应的工人（workers）。Master的责任如下：

给map 工人和reduce工人分配相应的任务，检测是否有工人进程死掉，将Map任务处理后的结果通知给Reduce任务。

## 根据自己的经验，谈谈对软件危机的看法？（结合表现、原因、如何克服描述）

答：软件危机是指软件生产方式无法满足迅速增长的计算机需求，开发和维护过程出现的一系列问题。以下几个原因导致：

（1）软件自身特点

（2）开发人员的弱点

（3）用户需求不明

（4）缺乏正确理论指导

（5）开发规模越来越大

（6）开发复杂度越来越高

可以通过软件生命周期的模型和软件工具的使用来缓解危机，通过程序自动化和软件工业化生产的方法实现软件标准化的目标，进一步缓解软件危机带来的影响。

软件危机有利有弊，除了带来许多麻烦，也给我们带来许多挑战，克服危机的过程，我们在技术上和创新上都有了一个提升，也算是间接为软件产业的发展做了贡献。

## 3、如何克服软件危机?

人们面临的不光是技术问题，更重要的是管理问题。管理不善必然导致失败 。

要提高软件开发效率，提高软件产品质量，必须采用工程化的开发方法与工业化的生产技术。

在技术上，应该采用基于重用的软件生产技术；在管理上，应该采用多维的工程管理模式。

或：

可以通过软件生命周期的模型和软件工具的使用来缓解危机，通过程序自动化和软件工业化生产的方法实现软件标准化的目标，进一步缓解软件危机带来的影响。

## 1、什么是Web服务体系结构？与传统的结构相比，使用Web服务有哪些好处？

答：Web服务作为一种新兴的Web应用模式，是一种崭新的分布式计算模型，是Web上数据和信息集成的有效机制。

Web服务就像Web上的构件编程，开发人员通过调用Web应用编程接口，将Web服务集成进他们的应用程序，就像调用本地服务一样。

这主要包括一下模块内容:数据层、数据访问层、业务层、业务面、监听者；

**Web服务的不同描述：**

应用的风不是；应用到应用的交互；平台无关性；

**Web服务开发生命周期：**

构建、部署、运行、管理；

**Web服务的特点：**

使用标准协议规范；使用协越的规范性；高度集成能力；完好的封装性；松散耦合；

**与传统的结构相比，具有以下几个优点：**

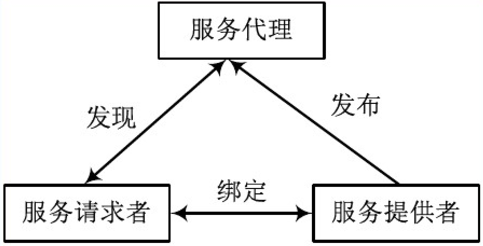
高度的通用性和易用性；完全的平台、语言独立性；高度的集成性；容易部署和发布；

## 2、在Web服务中，如何实现其松散耦合的特点？

答：C/S 结构是松散耦合系统，它们通过消息传递机制进行通话，由客户端发出请求给服务器，服务器进行相应处理后经传递机制送回客户端。

## 3、试分析SOA服务提供者、服务请求者和服务代理三者的作用，以及它们之间的工作流程。

答：服务请求者与服务提供者通过语义进行交互，服务提供者提交web服务描述给服务代理者，服务代理者返回web服务描述给服务请求者。



## 4、试解释Web服务栈的层次结构。

 答： XML(可扩展标记语言)、SOAP(简单对象访问协议)、WSDL(web服务定义语言)、

UDDI(统一描述发现和集成)。



## 5、Web服务有哪些核心技术，这些技术是如何在Web服务中发挥作用的。

答：Web服务技术核心基于可扩展标记语言XML的标准，包括简单对象访问协议，Web服务描述语言和统一描述，发现和集成协议。

SOAP定义了三部分：定义了描述消息和如何处理消息的框架的封装，表达应用程序定义的数据类型实例的编码规则以及描述远程调用和应答的协议和SOAP编订。

WSDL为服务者提供以XML格式描述的WEB服务请求的标准格式，经网络服务描述为能够进行消息交换的通信端点集合，以表达一个Web服务能做什么，他的位置在哪里，以及如何调用等信息。

UDDL规范描述了Web的概念，同时也定义了一种编程接口。通过UDDL提供的标准接口，企业可以发布自己的Web服务供其他企业调用和查询，业可以查询特地服务的描述信息，并动态的绑定到该服务上，通过UDDL，Web服务可以真正实现信息的“一次注册到处访问”。

## 6、从管理的角度看，SOA有什么优点？

答：（1）更易于维护；服务提供者和服务和服务使用者的松散耦合关系及对开放标准的采用确保了该特性的实现。

（2）更高的可用性；该特性在服务提供者和服务使用者的松散耦合关系上得以体现。使用者无需了解提供者的实现细节。

（3）更好的伸缩性；依靠服务设计、开发和部署所采用的架构模型实现伸缩性。服务提供者可以彼此独立调整，以满足服务需求。

## 7、在实际开发中，如何实现Web服务和SOA结构？

答：声明技术：J2EE 编程模型就是使用声明技术提供应用程序逻辑和中间件配置分离的一个例子。

     抽象：在某些情况下，SOA 基础结构中可以提供 API，以用于特定的用途。例如，SOA 基础结构可以提供错误报告和审核机制。在设计此类 API 时应非常小心，要注意其易用性。我们应优先考虑声明技术，而不是对这些机制进行编程配置。同样，在标准 API 可用时，我们应通过这些标准 API 公开 SOA 基础结构功能，而不是采用自己开发编写的方式。

   代码生成：在无法避免代码复杂性的地方，可以使用代码生成技术。例如，Web 服务描述语言（Web Services Definition Language，WSDL）就可以为开发人员隐藏 SOAP、HTTP 和 JMS 的复杂细节。这是通过组合用 WSDL 表示的可由计算机处理的接口定义和可从 WSDL 生成相关调用代码的语言特定实现的工具来实现的。

   工具：在不可避免 SOA 基础结构的细节进入开发人员代码的情况下，我们可以通过使用合适的工具扩展开发环境来减少开发人员工作的复杂性。

IBM Rational® Software Development Platform 产品所提供的基于 Eclipse 的环境可使用自定义插件、代码片段和用户指南轻松地进行扩展。

     模型驱动的开发：模型驱动的开发技术可以被视为前面两种方法的特定复杂组合，同时利用了工具和代码生成功能来简化开发体验。开发人员生成统一建模语言（Unified Modeling Language，UML）模型，此类模型可转换为相应的代码，其中包含利用 SOA 基础结构所必需的代码。

总之，在定义面向服务的体系结构及其基础结构时，我们必须特别注意开发人员的需求。当为开发人员提供指南，以告知他们应如何开发或使用服务时，我们应该寻找可促进这些指导方针遵循的机制。SOA 内的控制对其成功甚为关键。