1，团队沟通

基本上不管做什么开发，都是一个团队来完成的，游戏也是如此，游戏团队一般由老板，总经理，CTO(技术主管)，主策划（领导一些人，包括数值策划，系统策划，特效策划），主美(领导一些人，包括原画，UI设计，特效动作设计)，客户端主程（领导一些人，客户端程序员，客户端程序员...）,服务器主程（领导一些人，包括服务器程序员，服务器程序员），外加运维。而游戏的大部分逻辑实现与逻辑数据验证都会放在服务器端，所以服务端程序需要明确了解策划的需求，要了解就需要沟通，沟通方式的正确与否，直接关系到功能的实现是否正确，由于游戏逻辑的复杂性，单纯的文档描述可能不会非常完整，不像其它行业需求文档就几百页，详细的图文并茂，制定好之后也很少变化。所以做为一名游戏程序员，一定要有良好的沟通方式和技巧。

2，[架构](http://lib.csdn.net/base/architecture)设计

这个架构设计就像盖房子打基础，基础好，房子就稳固，基础不好，房子高了就容易倒。架构设计需要结合软件工程学来搞，它需要对服务器的整个流程有足够的了解，对需求的变化有足够的认识。架构的设计一般有几个特性。

首先是易用性，架构一旦完成，在开发的时候就要方便使用，比如网络通信架构，设计好之后，其他开发者就不需要关心客户端的数据是怎么被传输到服务器端的，这个时候对于服务器开发者来说，只需要实现一些简单的接口，就可以直接对客户端发送来的请求进行处理操作。再比如说服务器端数据的存储与更新，开发者只需要写少量SQL语句或基本不用写，都由架构的底层代码完成，开发者只需要调用封装好的API，就可以把数据存入[数据库](http://lib.csdn.net/base/mysql)而不用关心数据的最终流向，只需要关心实现逻辑就可以了。

第二，可扩展性，可扩展性包括两个方面，一是代码的可扩展性，比如说游戏中的任务处理吧，一个游戏中任务可能有几十种，而且还可能不定时的增加，为了判断不同的任务类型该执行什么操作，最简单也是最差的写法是if else,想象一下，一个方法里面，有几十个if else，这简单是bug的理想诞生地呀。一种可行的做法是使用责任链模式（具体的请参考设计模式的实现），这样每种任务都有一个单独的类去处理它，而不会影响其它的类，符合开闭原则，相互关联少，越少越不容易出bug。二是部署的可扩展性，比如，如果在线人数突然增加或预期可能要增加，一台物理机器可能处理不过来这么多的请求，那怎么办？那就需要支持在不影响其它服务器运行的情况下，可以动态的添加机器。而当压力降低之后，又可以移除某些机器，合理利用资源。

第三，高吞吐量，这个是指能尽量最大化的利用计算机固定的资源，去处理更多的请求，更快速的响应客户端。这就需要在服务器架构设计的时候考虑异步处理，减少IO等待时间（比如请求[Redis](http://lib.csdn.net/base/redis)，存储数据库，和其它服务器通信）以及数据缓存。说到异步，一定会涉及到多线程，并发等相关的技术，所以架构设计的时候需要对这部分知识有足够的了解。

第四，要考虑是否所有的功能模块都放在同一个进程中。也就是需不需要分布式开发，哪些功能需要单独拿出来。对于手机游戏来说，一般要求同时在线量比较小，功能比较单一，所有功能都在一个进程中，人数大量同时在线时，可以多部署几组进程。而对于大型网页游戏或客户端游戏来说，特别是有些大区或不分区的情况，单个功能访问量大，服务器就要考虑分布式部署开发了。

架构设计一般需要有经验的开发者（项目主程）去搭建，新手可以做为了解，在接触到项目之后，可以按这个思路去理解项目的架构是怎么样构成的，如果让自己来做，能否模仿出来，有时间可以自己尝试去独立设计架构，锻炼自己的能力，有一天你是会实际去做的。

3，逻辑开发

架构搭建完成之后，紧接着就是游戏服务器的逻辑开发，这时才开始真正去实现游戏需要的内容，比如注册，登陆，任务，活动，背包，组队战斗等。由于游戏逻辑可能需要的判断条件多，组合变化多，所以在游戏逻辑开发过程中，你会慢慢发现面向对象的重要性。逻辑开发是一个任重而道远的过程，同一个问题，可能有很多种实现方式，不同的实现方式对效率和吞吐量有很大的影响，所以就需要对需求功能的理解要深入，不同功能之间的关联要明确。对常用的设计模式要知道如何使用。比如像上面说的替换数量比较多的if else的方式。逻辑开发需要谨慎细心，而且一定要自己[测试](http://lib.csdn.net/base/softwaretest)才可以，不然bug在不知不觉中就产生了。

4，系统周边开发

一个游戏成功的运营，需要很多服务去支持它，比如sdk接入，充值接入，日志统计，游戏运行管理系统（一般叫后台管理系统，是内部人员为了管理游戏的而开发的系统）。比如修改某个用户的等级，封号等。管理系统一般会用web开发，与游戏服务器通信。

二，游戏类型与技术选择

游戏服务器开发使用的技术取决于游戏的类型，不同的游戏类型，需要的游戏环境不一样，所使用的技术也不一样。但是在本质上都是一样的，都是面对数据，处理数据，不同的是面对的数量大小而已。[http://www.youxijishu.com](http://www.youxijishu.com/)

1，PC类端游

这类游戏在线人数庞大，游戏中要处理的数据也非常庞大。所以对服务器性能要求非常高，一般都是采用C++做为开发语言，C++可以直接操作内存数据，与[操作系统](http://lib.csdn.net/base/operatingsystem)直接交互，减少数据之间的复制，它运行效率高，处理速度快，是这类游戏开发的首选开发语言。服务器端采用分布式架构，把不同的模块分散在多台物理机上处理。需要学习的大致有C++编程，[Linux](http://lib.csdn.net/base/linux)网络编程、TCP/IP通讯协议、多线程编程再加数据库。它一般开发周期比较长，一个游戏的上线基本上需要三到五年。

2，网页游戏

这类游戏相对于端游来说，开发周期短，因为是网页游戏，游戏的界面展示依赖于网络传输，所在在画面和特效上会次于客户端游戏很多。游戏的特点主要集中在游戏的玩法上。但是对于服务器端来说，和端游类是差不多是一样的，有些公司之前是做端游的，他们就直接把端游的服务器架构拿来就可以使用，以完成快速开发。

3，[手机游戏](http://www.youxijishu.com/)

手机类游戏目前是最火最热门的游戏，因为他的用户量大，用户占有时间长。但是手机游戏大多数是一般小游戏，功能简单，玩法单一，一般都是休闲娱乐的。现在也有一些稍微大型的MORPG游戏。所以手机游戏开发周期更短，上线更快。

目前，游戏市场竞争激烈，当前服务器主流的开发语言是C++和[Java](http://lib.csdn.net/base/java)，但是C++学习难度大，开发速度慢。为了满足游戏服务器快速开发，快速上线，所以一般来说我们都是使用Java语言来开发服务器。近年来，随着游戏市场的发展，游戏服务器开发技术因Java而生成了一套体系。可以供开发者选择。

三，使用Java开发服务器需要学习什么

Java语言，由于学习成本低，开发速度快，稳定性高，开源框架多，目前已成为网页游戏和手机游戏服务器开发的主要语言。咱们从系统的开发流程简单梳理一下服务器开发需要用到的技术。

1，网络通信

这个是首要实现的，如果没有网络通信，就没有服务器存在的必要了。网络通信就需要建立网络连接。目前网络通信有两种方式，一种是短连接，比如http,一种是长连接，比如socket，当然http也是基于socket的，socket是通信的基础。所以要对tcp/ip通信的知识有所了解，明白通信的原理。基于这两种网络通信，游戏服务器也分为两种，弱联网和强联网。弱联网的游戏一般是指一些小型的游戏，比如开心消消乐，连连看，以及一些卡牌养成类游戏，这类游戏一般几秒钟或几分钟再会与服务器同步一次数据，一般会使用短连接。而像一些arpg游戏，实时战斗类游戏，以及带同屏显示玩家的游戏，这类游戏与服务器交互信息频繁，一秒钟可能几十次，会采用长连接，避免每次连接重新建立消耗系统资源，提高通信效率。

为了网络通信的效率，服务器要使用NIO（非阻塞网络通信）通信。它能支持大并发连接。[Java](http://lib.csdn.net/base/java)NIO是多路复用IO，在多路复用IO模型中，会有一个线程不断去轮询多个socket的状态，只有当socket真正有读写事件时，才真正调用实际的IO读写操作。因为在多路复用IO模型中，只需要使用一个线程就可以管理多个socket，系统不需要建立新的进程或者线程，也不必维护这些线程和进程，并且只有在真正有socket读写事件进行时，才会使用IO资源，所以它大大减少了资源占用。目前基于此技术有很多开源框架，最常用的有两种，Netty和Mina。

所以在网络通信这一块，如果是弱联网游戏，可以使用web那一套来开发游戏服务器，需要学习的技术一般有http原理，Json格式协议，servlet，Tomcat（也可以是其它web容器）,[spring](http://lib.csdn.net/base/javaee)等。如果是强联网游戏，要学习的技术有Netty或Mina可以选择一种，多线程以及线程池的应用。这是网络通信所必须掌握的。只要能把客户端发送的信息接收到，并解析成代码使用的明文，就是成功了一半了，剩下的事就是把代码封装好，方便逻辑开发调用！

通信这块还要考虑消息的并发，长连接情况下，怎么处理断包，粘包问题，每个用户的消息处理的是不是有序的，如果有序会不会阻塞消息，如果无序会不会造成处理混乱，比如后到的消息先处理了，这些问题都要处理好，目前一般是保证同一个用户的消息要有序处理！

2，数据存储

网络通信调试好之后，不要急着做逻辑开发，还需要把数据如何存储理清楚！因为服务器端操作的全是数据，如果处理的不好，容易出bug，丢数据，这对游戏玩家来说是致命的，不可接受的！数据存储要考虑，一，数据如何存到数据库，是同步存储，还是异步存储！同步存储即将数操作完之后立刻写入数据库，异步操作即数据操作完之后先存储到内存缓存，然后由另外的线程或进程再同步到数据库！游戏中一般都是采用的异步存储方式，因为游戏并发量大，必须低延时，快速响应客户端！如果直接操作数据库太慢，会造成消息阻塞！内存缓存可先择的框架有[redis](http://lib.csdn.net/base/redis)，memcache，具体怎么同步到数据库，需要自己去设计了！二，数据接口如何设计，能不能用工作生成这些数据操作的代码，能不能不用写SQL语句，需是封装在底层，或由工具生成。编程是门艺术，在这就体现出来了，当然是仁者见仁，智者见智了！三，大并发情况下数据的一致性，像这类可能多线程操作的数据，一般是放在内存中，由锁来控制并发！所以对锁的使用要熟悉，不要出现死锁，或锁粒度过大，造成线程的长时间等待的情况！四，当数据量太大，一个数据库存储不了，数据该怎么分库分表！一种是水平划分，一种是垂直划分！具体的划分方式其它资料已有详细介绍，请自行查找阅读！目前有一个开源的分库框架mycat，是用JAVA写的，大家可以研究一下！

3，逻辑开发

逻辑开发就是实现游戏策划想象的各种游戏功能，比如，登录，物品使用，战斗结算等！逻辑开发代码量巨大，相互之间有很紧密的耦合性，所以每个功能模块一定要划分好！最好是接触下单元测试，写之前考虑一下是否方便单元测试，这样设计的代码会更加清晰，每个方法责任明确，不容易出bug！正是因为逻辑代码复杂，为了更好的管理代码，前辈们给我们总结了一些经验，就是著名的设计模式，所以学习一下设计模式对代码的管理有很大的好处！

逻辑开发一般遇到的问题有：

3.1，数据同步

一说到数据同步或资源共享的时候，一般都会考虑到锁的使用。因为一份资源同时只能被一个线程访问才是安全的。Java的JDK中提供了一些锁，比如：synchronized，以及java.util.concurrent.lock包中的Lock对象，java.util.concurrent包中还提供了其它的一些原子操作的类，我们知道i++操作不是线程安全的，但是可以使用AtomicInteger中的getAndIncrement();方法代替，还有线程安全的ConcurrentHashMap哈稀Map。以及阻塞队列LinkedBlockingQueue等。都是逻辑开发中常用的处理数据同步的类。

3.2，设计模式的使用

使用设计模式，可以让代码更加清晰，可扩展性更强，维护性更佳，比如，任务系统，任务会有很多种类型，要获得任务数据时，在一开始写这个系统的时候，我是这样写的if(type == 1)做什么，else if(type == 2)做什么，else if(type == 3)......else if(type == 35) else等。如果需要添加新的类型，又要添加else，这些if else都在同一个方法中。最后都不敢动一块，就怕出bug。其实当一个方法中出现三个以上的if else将来还可能增加时，就应当考虑设计是不是有问题了，后来改成责任链模式或状态模式，就解决了这个问题。还有一个例子是，当一个值变化，要影响多个任务完成状态时，可以使用观察者模式或监听模式或订阅模式去实现，这样功能之间完全解耦，出问题的机率会很小很小。

3.3，数据缓存框架的API使用

目前主流使用的数据缓存框架有redis和memcache，虽然在逻辑开发前，主程会对这些进行一些封装，但是作为使用者还是需要对这些框架的客户端的使用要有所了解的。这些可以去阅读相关的文档。不是太难。

4，程序部署与运行

目前，大多数Java项目都采用maven管理 ，可以使用maven打包开发好的程序，程序一般运行在远程服务器上，比如云服务器。一般运行Java程序的远程服务器都是[linux](http://lib.csdn.net/base/linux)系统，需要使用Linux命令操作，或写一些shell脚本去自动化部署管理一些程序。

5，艰苦奋斗的精神

首先，一定要让自己对这一行有兴趣，明确自己在这一行的技术选择，人生选择。很多人都知道，程序员加班是常有的事，坚持的住就做，坚持不了就再换一家公司做。

综上所述，想做Java游戏服务器方面的开发要掌握的技术有以下一些：

1，网络通信框架，Mina或Netty必须熟悉一种。而且自己必须要亲自搭建过，并明白其它原理。

2，通信协议制定和处理断包粘包，这一般属于网络通信框架要解决的问题。

3，数据缓存框架，redis或memcache选择一个，能熟练使用其客户端的命令。

4，Java基础，[java](http://lib.csdn.net/base/java)NIO通信原理，Java集合的使用，Java多线程开发，Java锁的使用

5，了解一些设计模式。最好能把23种设计模式都看一遍，并结合自己的开发经验，看哪些可以用到设计模式，但也不能死套设计模式，要灵活运用。

6，熟悉使用[MySQL](http://lib.csdn.net/base/mysql)数据库

7，了解数据库连接池的一些框架，比如Mybatis,[hibernate](http://lib.csdn.net/base/javaee)

8，对Http协议熟悉，熟悉一种web容器，比如tomcat，了解其配置。

9，对常用的一些Linux命令要熟悉使用。

10，热爱学习，不断的充实自己，上面所说的只是入门技能而已，真正做起来要复杂的多，一定要让自己喜欢游戏这个行业，这样才能有动力做下去，做自己喜欢的工作还是比为了工作要好的！