问题：

MMORPG产品，特别是一些大型的MMO产品，有时候会使用无缝地图技术。就是服务器A负责地图A，服务器B负责地图B，地图A和地图B有交迭地带。无缝地图技术保证了用户在地图A可以自由移动到地图B。

1）请了解无缝地图技术相关知识

2）描述你对这种技术的认知

3）你认为这种技术的最大难点？优点，缺点

4）要求300字以上

技术的认知

无缝地图需要保证加载资源的时候绝对不卡，所以需要将加载行为分配到其他线程，并且在加载完后和主线程，渲染线程同步，需要不少专门设计。

1. 分区块的地形信息和数据加载
2. 按物件的多级精度加载
3. 贴图的逐级精度加载

此外，无缝地图除了保证以上的基本功能外，还有一个重要的部分是保证加载效率。你只有基本加载速度上去了，才可能让加载过程和游戏同步进行，否则就会让玩家看到未加载完成的地图。实际上，那些能够流畅无缝加载的游戏，真做成读条加载，加载时间也不长。

方法就是各种压缩数据，连续密集储存，让磁盘速度不再成为瓶颈。这里面其实有很多技巧。（有一个常用做法是臃余储存，就是将临近的数据复制一份单独放在一起，这样读取的时候就不用反复寻址，大大加速加载，而这样操作会导致游戏体积放大N倍，所以开放世界游戏往往会占用更大的磁盘空间）

此外，还要设置恰当的加载策略，比如室内场景从什么时候开始加载，优先加载哪一个。这个通常需要一个自动化的测试工具，控制AI在地图的各个角落跑动，并记录帧率和IO压力，然后据此调整加载时机，保证游戏的流畅度。某些游戏更新后流畅度提升，靠的也是这个。

无缝地图的实现要素

无缝地图的实现要素有两种，一种是象《魔兽3》、《天骄3》直接以表型结构将一个超大的地图不断切分并以树形结构分离数据进行保存，而另一种是制作好无数个地图，但交接处做成一样的，让玩家在视觉上感觉还是在同一地图中。无论使用哪一种手法，都有一个难题：当人物跨过一张地图时，我们如何让其相关的数据做到平滑无缝交接?答案就是利用无缝贴图技术!

无缝贴图的制作

无缝贴图是指可以循环重复拼接的图片，这样，地图制作人员就可以很方便的画出连续、无缝、交接良好的地图，实现无缝地图的游戏基本都使用了拼接地图的制作方式，这样的好处很明显，可以有效的利用图形资源，且很容易进行切分，而使用拼接地图最重要的元素就是无缝贴图。随着时代的进步和技术的不断发展，现在的游戏一般都将无缝贴图处理的很完美了，而早期美术工具落后的时候，很多游戏都是使用地板式的制作方式，这样使得场景看上去过于整齐，显的十分呆板和虚假。

超大场景的无缝交接

有了制作地图使用的无缝贴图还不行，真正重要的技术难点是超大场景的无缝交接。一般玩家的机器内存有限，不可能将整个游戏的全部场景都保存下来，而这就迫使开发者要将地图切分，但这个工作对玩家来说是透明的，具体的方法就是将一个巨大的场景按格子划分成多个小地图，当玩家在地图边缘时再读取附近的几块区域。

地图数据的切分与汇总

在地图交接处，玩家看到的是一整张地图，但设计上在后台处理的可能是多张地图，而如何对地图数据进行划分和汇总就成了另一个难题。一种解决方法就是将整个超大场景当作一张地图来编辑，将整个地图拆分成多张图形后在每张图形上加入坐标，在交接处同时载入多张图形，在保存时根据图形的坐标自动分开保存到对应的位置上去，这样一来，地图本身就是在一起编辑的，所以数据在游戏就可以更好的汇总在一起。这也就是为什么在游戏中离的很远的时候已经看到了目的地的轮廓，却无法看清的原因之一。

难点

工序难点

1：美术资源制作工序

无缝地图相对于完整地图的资源，美术人员需要对整个场景资源做切割，或者单独制作每一个地图块的资源（我说的是9宫格（2d游戏）或者8叉树结构（3d游戏））。这增加了很多工序。

另外场景是需要光照的，所以一旦切割，就要考虑对烘焙光照的影响。

场景还有地上物，所以美术人员要把地上物的归属命名都要划分好。

场景还需要网格寻路，需要考虑切割对网格寻路的影响。

一般无缝的地图，还需要考虑不同LOD级别的场景资源，也就是平时玩家觉得离自己近的景物看上去清晰，远的景物看上去模糊的原因，所以同样的场景，要制作不同LOD质量的几份。

3d场景切割还会有其他问题，比如地图缩放接缝问题，这个需要Shader来解决，比如天气变化，雨雪效果，也需要Shader来解决，所以难点是Shader的使用。

2：前端程序控制工序

前端的难点是控制，首先前端要拿到美术给到的地图资源，做好地图过度处理，是镜头跟随还是主角跟随，简单点说就是一个视口坐标，一个人物坐标，一个世界坐标，还有每个地图块的坐标，这里说的是9宫格，8叉树的结构。

如果像魔兽那样，甚至可以考虑是非规则的结构，也就是地图由程序根据顶点和一些设定来做自定义的分隔，这样分隔出来的就是一些不规则的4棱锥，这样在人物移动，视景变化的时候，加载会更自然，不会出现卡顿，但这个算法要求极高，需要极高的图形学理论知识。

前端还要考虑时时光，需要考虑地图切换时，场景模型，地上物，人物，粒子特效，等等变化。

前端还需要考虑寻路方式，刷怪方式，资源缓存，卸载，对内存的监控。

前端需要解决大量Shader的处理，地图切割对图像也会有影响。

3：服务端程序控制工序

morpg游戏时，一般服务端都会建立一个相对应的数据化的虚拟世界，称为world，然后world里又要划分rec区域。

为什么说服务端难呢？因为前端的所有东西都可以通过工具来建模，模型的顶点，地形的网格都不需要你来手写，但后端不同，后端经常需要自己亲自手写网格算法，说白了就是前端会的后端必须会，而且要求会的十分底层，对基础基础要求太高。

世界划分区域的目的是什么？可不只是为了地图切割，服务端对性能要求极其高，如果一个超大场景里有100个或者更多的玩家，同时加载，那服务器会卡爆的，所以它必须要考虑划分区域，注册事件，贴图算法等，反正简单的理解就是需要一种算法来解决玩家移动，怪物生成，物品掉落，怪物移动，所有ai的处理。

服务器做的事就是根据玩家的输入，计算出结果返回给前端，这个结果包括了坐标，刷怪信息，物品掉落信息，其他玩家信息，npc信息，等等。前端再去加载对应的资源，包括切割地形，光照处理，怪物，npc，掉落物，粒子特效，等等。

所以这么多工序下来就要求3个部门的人员配合非常繁琐。并且对前端后端人员图形学，Shader，网格算法的要求也是极高的。至少国内现在的开发人员，没有10年以上的经验，很难驾驭这种服务端，没有5年经验的很难驾驭前端。这些会增加大量的时间成本和人员素质成本。

技术难点

1. 客户端游戏引擎的实现难点

涉及到大地图的分片的动态加载，想想你站在好几块地图的边角上，客户端要能根据你的位置，把你视野里面的小地图实时动态加载上来，并且还必须让玩家感觉不到这种加载的延迟。这对引擎的性能要求较高，另一方面，也就是对显卡的要求更高（这其实相当于另一方面限制了游戏玩家的受众群体，毕竟国内还有大量的二三线城市发展水平还比较低，没那么多好的PC机）

2. 网络服务器端的实现难点

无缝地图相当于变相限制了玩家在同一个服务器上，相当于在视野，战斗中，你需要能够快速看到其它玩家，其它的怪物。想像一个，你正处于一块无缝分割的一个小地图的边角上，需要与其它分割的小地图上面的玩家实时互动，如果这两块地图的玩家不在同一个地图服务器上，就需要涉及到复杂的网络交互及玩家，怪物NPC的数据交换（分布式上面技术上实现比较复杂），实时性是比较难保证的（个人觉得，实时性无法保证，要无缝也没什么意义）。而都同一个服务器上，硬件上限制了实时活跃玩家的总数，包括CPU限制，内存限制，甚至是网络流量的限制。

最大难点：

无缝大世界主要的难点还是在服务器上. 在这个架构下, 负载不再是跟着场景拆, 而是跟着人群, 这意味着一个场景中, 可以相互看见的人是可能在不同的服务器进程上的, 导致了玩家之间的交互都要采用异步消息的方式, 比如对目标释放技能等. 纯异步的服务器理论上是可以无限扩展的, 可以用2台主机来负载, 也可以用10台主机来负载, 但是纯异步的方式也让编程的复杂度上升了很多个等级.

优缺点

无缝世界模式的优点

1．拥有更大的游戏空间无缝世界在大小上具有明显优势。

无缝游戏世界可以有更大的连续区域来进行游戏，而一个分区游戏世界所能支持的区域大小，受限于单个服务器所能处理的玩家数量。对玩家来说，无缝游戏世界可以营造出一个更加引人入胜的环境，并且游戏仿真也可以变得更为真实。对于某些游戏设计来说，把游戏世界分解为独立的区域甚至是不可行的，这时必须使用无缝游戏世界。

2．可伸缩性和可靠性无缝游戏世界可以给游戏带来一个更为实际的好处，那就是可以增加可伸缩性。

随着游戏世界中的玩家越来越多，服务器需要进行调整以处理不断增加的负载，更不用说那些能在运行时自行平衡的动态服务器边界了。即使服务器边界是静态的，也可以根据玩家的数量和分布，在停机时间对它们进行调整来达到同样的效果。 无缝世界还具有更高的可靠性。如果某个服务器中止服务或是某个服务器进程崩溃了，就可以调整服务器边界来把负载分散到其余的服务器中。一个更强大的动态服务器边界实现甚至可以在检测到某个服务器崩溃时自动地做到这点。

3．不确定的边界线游戏中有很多错误与在服务器间来回移动数据相关，动态服务器边界的另一个优点是它们可以降低玩家对这些错误进行滥用的可能。

理由很简单：玩家对于究竟哪里是服务器边界并没有一个确切的概念，当玩家发现这些边界后，只需要对服务器边界进行调整就可以避免他们对此滥用。

4．不需要载入地图无缝游戏世界甚至对玩家也是有好处的。

“载入地图”所需要的时间会大大减少（除了初始化以外），这些载入操作会被平摊到游戏世界中每一个分块上，这些分块仅在需要时才会被载入。

无缝世界模式的缺点

（和分区服务器设计相比，无缝服务器会使游戏开发和维护上的所有工作都变得更为困难）

1．不确定法则：

不仅仅是客户端分区服务器和无缝服务器在复杂度上的区别类似于单人游戏和多人游戏之间的区别。在一个无缝服务器环境中，由于表示远程对象的代理是不精确的，因此在服务端存在着相同类型的不确定性。游戏设计必须考虑到，代理对象不可能和它们所对应的真实对象完全同步，除非付出巨大的代价。这意味着游戏系统必须对很多很简单的玩家交互进行异步处理（譬如说，使用消息传递）实现以避免使用可能失效的数据。异步处理的本质导致我们需要解决大量的失败情况。不仅如此，代理对象缺乏同步是很多错误的根源，而这些错误可能会被玩家滥用

2．对设计的影响

实践中，几乎每个游戏系统（移动、战斗、角色升级以及制造）在无缝世界中都会变得更为复杂并且更容易发生错误。这会直接导致开发周期延长或是使游戏变得更不稳定，更可能是两者都有。

4．构筑游戏世界构筑游戏世界就是创造游戏发生的物理世界的过程。

构筑游戏世界的任务包括地形建模、贴图绘制和对象放置等。显而易见，建造一个无缝世界比建造一个分区世界要困难很多。刚开始的时候，整个世界可能太大了，以至于无法把它作为一个整体进行编辑，因此，游戏编辑人员需要能够在更小的分块上工作，再加上多个游戏世界构筑人员需要同时进行工作，显然他们需要使用一个基于分块的版本控制系统来管理对游戏世界的并发访问。 游戏编辑人员需要花费额外的时间来“整理”那些由不同人员编辑的区域边界。在放置大型对象时需要格外小心，因为它们可能会跨越服务器边界而导致问题的产生。如果假设在同一建筑物中的对象应该在同一服务器上（可能是为了有效地进行裁减），那么和潜在服务器边界相交的建筑物必须改变位置。 与之相反，在分区游戏世界中，编辑人员可以在一个既没有边界区域也没有放置限制的连续区域中工作。虽然这仍然可能需要某种形式的锁定方式，但是它可以在一个更高的粒度（譬如说地图文件）上进行，而不是在整个游戏世界的任意子集上进行。

5．美工

令人难以置信的是，在无缝世界中就连美工制作也会变得更为复杂。创建像建筑物那样的大型对象会受到服务器边界区域大小的影响；任何对象的最大尺寸必须小于共享边界的大小；否则，这样的对象可能会由于跨越边界区域而导致问题。

6．运营和扩展

要在一个无缝游戏世界中创造更多的内容仍然需要游戏开发人员更多的努力。此外，如果游戏运营团队中的人员不是出自原始的开发团队，他们就可能会忘记无缝服务器带来的大量问题和限制，而这会给运营团队带来很大的麻烦。 不仅如此，大量对游戏的侵入行为会在服务器边界上发生。即使在分区世界中，物品复制也是一个比较常见的问题。然而，在服务器边界上更容易进行物品复制的根本原因是竞争条件。代理和相应的对象不能保持完全同步会导致大量的失效状态的错误。发现并且修复这样的错误将是运营团队的主要职责之一。