基于BSDiff的AssetBundle差异更新方案

功能简介：

在更新AssetBundle资源时，不用完整下载整个ab包。只需要下载服务器上的ab包和本地ab包的差异文件，本地再合成一个新的ab包，这样可以减少下载量。

举个例子，比如一个90M的ab文件，在版本变更了之后它变成了100M，按照常规的逻辑来说，玩家只要下载这100M文件就可以了。本方案是将这两个ab包在CDN上计算好差异，差异可能在10M左右，那么玩家只需要下载10M的补丁，这个10M的补丁会和之前的这个90M的文件做一次合并，然后进入游戏。

BSDiff简介：

BSDiff是一个差量更新算法，它在服务器端运行BSDiff算法产生patch包，在客户端运行BSPatch算法，将旧文件和patch包合成新文件

算法原理：

http://www.daemonology.net/bsdiff/

https://blog.csdn.net/add_ada/article/details/51232889

https://cloud.tencent.com/developer/article/1008518

BsDiff.Native.DiffApply的返回结果：0代表两个文件合并成功，1代表两个文件一样无需合并，-1代表合并失败

两种BSDiff的算法：

File-BSDiff：基于File做差异和合并

> 🏆 DiffCreate(string oldpath,string newpath, string patchpath, IntPtr progresscontrol)
> DiffApply(string oldpath, string patchpath, string newpath, IntPtr progresscontrol)

FileStream-BSDiff：基于FileStream做差异和合并

🏆 Create(byte[] oldData, byte[] newData, Stream output, ProgressDelegate pd = null)

Apply(Stream input, Func<Stream> openPatchStream, Stream output)

合并时间：

测试设备时小米9。一共76个文件，共计590M大小，基于FileStream的补丁包是48M，基于File的差异补丁包是67M左右

	常规热更，不做差异	FileStream-BSDiff补丁包	File-BSDiff补丁包
差异文件数量	76	76	76
热更下载大小	590M	48M	67M
真机上差异合并时间(主线程)		38s	15s
真机上差异合并时间(主线程+1个子线程)		25s	
真机上差异合并时间(主线程+3个子线程)		20s	
真机上差异合并时间(主线程+4个子线程)		17s	
更新优点	没有合并时间，cdn也不需要提前计算差异	更新文件非常小，大概有10到50倍的效益。基于文件流的形式合并，不用担心堆内存溢出的问题	更新文件小。CDN计算差异较快，手机上合并差异也较快。
更新缺点	更新文件太大，AssetBundle包的粒度控制和依赖加载很难做到平衡	合并文件需要花点时间，多线程可以缓解耗时。CDN上计算差异较慢，可以通过提高机器配置来减少时间，以及引入并行计算减少两个版本之间的差异计算时间。	对大文件不友好，一个100M的文件和它的补丁包合并时，分配的堆内存过大(堆内存溢出)，模拟器有百分之20的概率崩溃(手机未发现，可能是模拟器性能较弱)。

5个版本的差异计算：

版本差异包大小：

1.0.141.526.0：93.0MB

1.0.141.527.1：94.2MB

1.0.141.532.0：90.2MB

1.0.141.536.0：38.8MB

1.0.141.538.0：2.93MB

1.0.141.539.0：

机器配置	串行计算	版本并行计算	版本并行计算+文件并行计算
Intel Core Processor 2.6GHz + 32G RAM	120分钟	40分钟	
Intel Core i7-8700 3.2GHz + 32G RAM	88分钟	28分钟	18分钟
Intel Core i9-9900k 3.6GHz + 64G RAM		18分钟	10分钟

总结：通过修改计算方式和提升机器配置，可以大幅度减少计算时间

后续优化：优化差异算法可以再降低不少时间。

整包和小包两种情况下的补丁合并流程：