本申请涉及一种基于版本管理的数据打包更新方法，通过终端发送获取数据打包更新的请求指令；通过打包任务管理模块接收所述请求指令，根据所述请求指令执行打包更新任务，实时反馈并储存所述打包更新任务的执行状态数据；获取所述执行状态数据，发送并展示所述执行状态至所述终端。能够根据移动端项目从内部测试到上线过程中的实际需要，在测试阶段就按照发布基础包加多个更新包的模式，灵活地在打包机之间完成通信，存储，日志记录，版本管理，分支管理，自定义显示等功能，降低部署的难度，在构建过程中发生的错误可以及时反馈处理，提高如游戏项目迭代开发的效率。

指定说明书附图1为摘要附图

1、一种基于版本管理的数据打包更新方法，其特征在于，包括如下技术步骤：

S100、通过终端发送获取数据打包更新的请求指令；

S200、通过打包任务管理模块接收所述请求指令，根据所述请求指令执行打包更新任务，实时反馈并储存所述打包更新任务的执行状态数据；

S300、获取所述执行状态数据，发送并展示所述执行状态至所述终端。

2、根据权利要求1所述的基于版本管理的数据打包更新方法，其特征在于，在步骤S200中，所述通过打包任务管理模块接收所述请求指令，并根据所述请求指令执行打包更新任务，实时反馈并储存所述打包更新任务的执行状态数据，包括：

S210、预设打包更新任务管理规则，并将所述打包更新任务管理规则配置于所述打包任务管理模块；

S220、所述打包任务管理模块基于所述打包更新任务管理规则，执行所述请求指令对应的打包更新任务；

S230、获取所述打包更新任务的执行状态，并将所述执行状态储存至MySql数据库。

3、根据权利要求2所述的基于版本管理的数据打包更新方法，其特征在于，在步骤S200中，所述通过打包任务管理模块接收所述请求指令，并根据所述请求指令执行打包更新任务，实时反馈并储存所述打包更新任务的执行状态数据，还包括：

S211、预设访问规则，并将所述访问规则配置于所述打包任务管理模块；

S212、所述打包任务管理模块基于所述访问规则，发送所述打包更新任务的执行状态请求指令至所述MySql数据库；

S213、所述MySql数据库返回所述打包更新任务的执行状态，发送至所述终端并展示。

4、根据权利要求2所述的基于版本管理的数据打包更新方法，其特征在于，在步骤S200中，所述通过打包任务管理模块接收所述请求指令，并根据所述请求指令执行打包更新任务，实时反馈并储存所述打包更新任务的执行状态数据，还包括：

S221、预设调用规则，并将所述调用规则配置于所述打包任务管理模块；

S222、所述打包任务管理模块基于所述调用规则，调用执行所述请求指令的任务执行接口，其中，所述任务执行接口为Python的SVN操作接口、Python的文件操作接口、Python的远程管理接口或Python的FTP接口中的至少一种；

S223、通过调用的任务执行接口，执行所述请求指令所请求的打包更新任务。

5、根据权利要求4所述的基于版本管理的数据打包更新方法，其特征在于，在步骤S200中，所述通过打包任务管理模块接收所述请求指令，并根据所述请求指令执行打包更新任务，实时反馈并储存所述打包更新任务的执行状态数据，还包括：

S224、获取执行打包更新任务过程中的任务执行日志数据，以此作为所述打包更新任务的执行状态数据；

S225、将所述打包更新任务的执行状态数据，记录到所述打包任务管理模块。

6、根据权利要求4所述的基于版本管理的数据打包更新方法，其特征在于，在步骤S222中，所述调用执行所述请求指令的任务执行接口，包括：

S2221、根据所述请求指令，获取所述请求指令的请求类型；

S2222、根据所述调用规则，调用并获得与所述请求类型对应匹配的任务执行接口；

S2223、通过所匹配的任务执行接口，完成所述打包任务管理模块与任务执行模块之间的连接。

7、根据权利要求6所述的基于版本管理的数据打包更新方法，其特征在于，在步骤S200中，所述通过打包任务管理模块接收所述请求指令，并根据所述请求指令执行打包更新任务，实时反馈并储存所述打包更新任务的执行状态数据，还包括：

S231、根据所述请求指令，获取所述请求指令的任务信息，其中，所述任务信息包括任务的版本号、资源版本号、执行任务的log、任务的命令行参数或运行机器的信息中的至少一种；

S232、将所述请求指令的任务信息记录并保存到所述MySql数据库中；

S233、定时获取打包任务执行过程中的log，并返回至所述终端。

8、一种实现权利要求1-7中任一项所述的基于版本管理的数据打包更新方法的装置，其特征在于，包括：

终端，用于发送获取数据打包更新的请求指令；

打包任务管理模块，用于接收所述请求指令，根据所述请求指令执行打包更新任务，实时反馈所述打包更新任务的执行状态数据至所述终端；

MySql数据库，用于实时储存所述打包更新任务的执行状态数据。

9、根据权利要求8所述的装置，其特征在于，所述终端，还用于接收所述执行状态数据，并展示所述执行状态。

10、一种数据更新系统，其特征在于，包括：

处理器；

用于存储处理器可执行指令的存储器；

其中，所述处理器被配置为执行所述可执行指令时实现权利要求1至8中任意一项所述的基于版本管理的数据打包更新方法。

基于版本管理的数据打包更新方法

技术领域

本公开涉及数据处理技术领域，尤其涉及一种基于版本管理的数据打包更新方法、装置和数据更新系统。

背景技术

项目的数据打包和更新系统，通常使用Jenkins作为打包工具，与某种版本管理工具如(Git，Svn等)配合使用。需要在指定机器上部署Jenkins节点，在指定的分支上获取项目的代码和资源，运行带参数的命令，执行打包过程，远程客户端可以通过一个网页界面查看生成结果。

在具体的移动项目实践中，Jenkins能满足一部分远程打包的需求，但也存在几个不足：

（1）版本管理：Jenkins的打包记录以递增数字的方式呈现，无法和项目的版本号管理建立直接联系；

（2）对更新包的支持：移动游戏一般分为基础包加多个更新包的开发模式，更新包指的是游戏的增量资源，随着游戏的运营陆续发出由玩家自动下载；而Jenkins不能直接区分基础包和更新包；

（3）自定义的构建需求：在构建中，通常需要一些自定义的需求比如机器之间通过Ftp，curl等上传和下载，PC和Mac机器通过Python，Shell脚本通信等等，这些过程在执行中可能由于某些因素中断，在中断时需要停止整个构建过程并给出详细日志记录；这种多样化的构建需求Jenkins支持起来较为困难；

（4）对打包执行和结果显示的需求：在打包过程中，客户端有时需要停止某个更新版本或者增加某个空更新版本，并希望完全不记录某次构建数据，以保持版本号的连续。

发明内容

有鉴于此，本公开提出了一种基于版本管理的数据打包更新方法、装置和数据更新系统。

根据本公开的一方面，提供了一种基于版本管理的数据打包更新方法，包括如下技术步骤：

S100、通过终端发送获取数据打包更新的请求指令；

S200、通过打包任务管理模块接收所述请求指令，根据所述请求指令执行打包更新任务，实时反馈并储存所述打包更新任务的执行状态数据；

S300、获取所述执行状态数据，发送并展示所述执行状态至所述终端。

在一种可能的实现方式中，可选地，在步骤S200中，所述通过打包任务管理模块接收所述请求指令，并根据所述请求指令执行打包更新任务，实时反馈并储存所述打包更新任务的执行状态数据，包括：

S210、预设打包更新任务管理规则，并将所述打包更新任务管理规则配置于所述打包任务管理模块；

S220、所述打包任务管理模块基于所述打包更新任务管理规则，执行所述请求指令对应的打包更新任务；

S230、获取所述打包更新任务的执行状态，并将所述执行状态储存至MySql数据库。

在一种可能的实现方式中，可选地，在步骤S200中，所述通过打包任务管理模块接收所述请求指令，并根据所述请求指令执行打包更新任务，实时反馈并储存所述打包更新任务的执行状态数据，还包括：

S211、预设访问规则，并将所述访问规则配置于所述打包任务管理模块；

S212、所述打包任务管理模块基于所述访问规则，发送所述打包更新任务的执行状态请求指令至所述MySql数据库；

S213、所述MySql数据库返回所述打包更新任务的执行状态，发送至所述终端并展示。

在一种可能的实现方式中，可选地，在步骤S200中，所述通过打包任务管理模块接收所述请求指令，并根据所述请求指令执行打包更新任务，实时反馈并储存所述打包更新任务的执行状态数据，还包括：

S221、预设调用规则，并将所述调用规则配置于所述打包任务管理模块；

S222、所述打包任务管理模块基于所述调用规则，调用执行所述请求指令的任务执行接口，其中，所述任务执行接口为Python的Svn操作接口、Python的文件操作接口、Python的远程管理接口或Python的Ftp接口中的至少一种；

S223、通过调用的任务执行接口，执行所述请求指令所请求的打包更新任务。

在一种可能的实现方式中，可选地，在步骤S200中，所述通过打包任务管理模块接收所述请求指令，并根据所述请求指令执行打包更新任务，实时反馈并储存所述打包更新任务的执行状态数据，还包括：

S224、获取执行打包更新任务过程中的任务执行日志数据，以此作为所述打包更新任务的执行状态数据；

S225、将所述打包更新任务的执行状态数据，记录到所述打包任务管理模块。

在一种可能的实现方式中，可选地，在步骤S222中，所述调用执行所述请求指令的任务执行接口，包括：

S2221、根据所述请求指令，获取所述请求指令的请求类型；

S2222、根据所述调用规则，调用并获得与所述请求类型对应匹配的任务执行接口；

S2223、通过所匹配的任务执行接口，完成所述打包任务管理模块与任务执行模块之间的连接。

在一种可能的实现方式中，可选地，在步骤S200中，所述通过打包任务管理模块接收所述请求指令，并根据所述请求指令执行打包更新任务，实时反馈并储存所述打包更新任务的执行状态数据，还包括：

S231、根据所述请求指令，获取所述请求指令的任务信息，其中，所述任务信息包括任务的版本号、资源版本号、执行任务的log、任务的命令行参数或运行机器的信息中的至少一种；

S232、将所述请求指令的任务信息记录并保存到所述MySql数据库中；

S233、定时获取打包任务执行过程中的log，并返回至所述终端。

根据本公开的另一方面，提供了一种实现上述所述的基于版本管理的数据打包更新方法的装置，包括：

终端，用于发送获取数据打包更新的请求指令；

打包任务管理模块，用于接收所述请求指令，根据所述请求指令执行打包更新任务，实时反馈所述打包更新任务的执行状态数据至所述终端；

MySql数据库，用于实时储存所述打包更新任务的执行状态数据。

在一种可能的实现方式中，可选地，所述终端，还用于接收所述执行状态数据，并展示所述执行状态。

根据本公开的另一方面，还提供了一种数据更新系统，包括：

处理器；

用于存储处理器可执行指令的存储器；

其中，所述处理器被配置为执行所述可执行指令时实现上述所述的基于版本管理的数据打包更新方法。

本申请的技术效果：

本发明通过终端发送获取数据打包更新的请求指令；通过打包任务管理模块接收所述请求指令，根据所述请求指令执行打包更新任务，实时反馈并储存所述打包更新任务的执行状态数据；获取所述执行状态数据，发送并展示所述执行状态至所述终端。能够根据移动端项目从内部测试到上线过程中的实际需要，在测试阶段就按照发布基础包加多个更新包的模式，灵活地在打包机之间完成通信，存储，日志记录，版本管理，分支管理，自定义显示等功能，降低部署的难度，在构建过程中发生的错误可以及时反馈处理，提高如游戏项目迭代开发的效率。

根据下面参考附图对示例性实施例的详细说明，本公开的其它特征及方面将变得清楚。

附图说明

包含在说明书中并且构成说明书的一部分的附图与说明书一起示出了本公开的示例性实施例、特征和方面，并且用于解释本公开的原理。

图1示出为本发明基于版本管理的数据打包更新方法的实施流程示意图；

图2示出为本发明实施基于版本管理的数据打包更新方法时的硬件结构示意图。

具体实施方式

以下将参考附图详细说明本公开的各种示例性实施例、特征和方面。附图中相同的附图标记表示功能相同或相似的元件。尽管在附图中示出了实施例的各种方面，但是除非特别指出，不必按比例绘制附图。

在这里专用的词“示例性”意为“用作例子、实施例或说明性”。这里作为“示例性”所说明的任何实施例不必解释为优于或好于其它实施例。

另外，为了更好的说明本公开，在下文的具体实施方式中给出了众多的具体细节。本领域技术人员应当理解，没有某些具体细节，本公开同样可以实施。在一些实例中，对于本领域技术人员熟知的方法、手段、元件和电路未作详细描述，以便于凸显本公开的主旨。

实施例1

本技术为一种线上数据的打包更新处理方法，本实施例，优先用作游戏项目的线上数据打包更新。

如图1所示，根据本公开的一方面，提供了一种基于版本管理的数据打包更新方法，包括如下技术步骤：

S100、通过终端发送获取数据打包更新的请求指令；

用户在需要更新游戏数据，进行版本更新时，首先需要通过终端发送更新的请求。

其中，本实施例，采用前端Web网页的方式，以HTML+PHP网页形式显示给用户，用户通过Web界面执行新建任务和显示任务列表等操作，发送获取数据打包更新的请求指令。

S200、通过打包任务管理模块接收所述请求指令，根据所述请求指令执行打包更新任务，实时反馈并储存所述打包更新任务的执行状态数据；

本技术，本系统使用Svn版本管理工具来管理游戏开发过程中的资源和代码版本，设置Svn的分支和版本号，再设置是打基础包还是更新包，如果是基础包填写App版本号，就可以开始一个打包任务。

例如：要打一个App版本为1.0.0的基础包，用Svn号111，生成的就是1.0.0版本的基础包app和1.0.0.0的资源版本的更新资源，对应Svn号111下一次需要打一个更新包，用Svn号112，生成的就是1.0.0.1的资源版本的更新包，对应Svn号112(基础包app通过更新资源包的前三位版本号获取)。

对一次打包任务，如果是基础包则产出为:基础包app+基础资源版本号的资源；如果是更新包则产出为：更新资源版本号的资源(app使用更新版本号对应的基础包app)。

在建立一个新任务时，可以选择是重新打一次基础包，也可以选择在之前版本的基础上打更新包。

结合图2可知，打包任务管理模块接收用户在Web页面的请求，可以使用PHP脚本通过访问MySql数据库来获取任务的状态信息返回给用户。打包任务管理模块负责对所有的打包任务进行管理，除了通过PHP脚本访问MySql数据库来获取任务的状态以外，还使用Python语言调用接口来执行具体的打包任务。

Python提供的接口有Svn操作，文件操作，远程管理，Ftp操作这几类。具体的打包任务则是由Unity命令实现的打包模块，也就是游戏具体的打包规则，如何生成美术打包资源和App。

在执行时，此外还有一个用来存储备份的远程机器，在打包过程中用来实现打包生成资源的存储和之前历史版本资源的获取。

S300、获取所述执行状态数据，发送并展示所述执行状态至所述终端。

打包任务管理模块根据调用的接口，执行完所述请求指令的任务后，系统会实时将执行过程的执行状态数据，进行记录，并保存在MySql数据库。

在打包过程中，打包状态和打包机的状态都可以通过前端Web页面显示，如果用户不想使用某个版本的构建结果，方便重新使用某个版本号，可以通过Web页面操作来删除某个打包记录，也就是操作数据库中记录的版本信息。

一次打包任务，可以看作是：远程控制Unity机器，取代码资源，构建App和资源，从Ftp获取对应版本资源，比较差异产生更新包，上传到Ftp存储结果，数据库存储打包版本记录的过程。每次打包任务的日志是整个串行化流程的日志，在执行远程控制，执行Unity命令，Ftp操作等过程中，每个步骤都记录执行命令参数，执行过程，执行结果等信息，如果某一步骤执行失败时，命令会记录失败原因并中断整个打包流程。在打包过程中，打包状态和打包机的状态都可以通过前端Web页面显示，如果不想使用某个版本的构建结果，方便重新使用某个版本号，可以通过Web页面操作来删除某个打包记录，也就是操作数据库中记录的版本信息。

在一种可能的实现方式中，可选地，在步骤S200中，所述通过打包任务管理模块接收所述请求指令，并根据所述请求指令执行打包更新任务，实时反馈并储存所述打包更新任务的执行状态数据，包括：

S210、预设打包更新任务管理规则，并将所述打包更新任务管理规则配置于所述打包任务管理模块；

对于用户发送的请求打包更新的指令，需要打包任务管理模块对不同的请求指令进行管理，因此，需要提前管理方或者项目运营方，提前对打包任务管理模块配置打包更新任务管理规则，便于根据打包更新任务管理规则对用户的更新请求进行管理、处理。

S220、所述打包任务管理模块基于所述打包更新任务管理规则，执行所述请求指令对应的打包更新任务；

本实施例，打包任务管理模块负责对所有的打包任务进行管理，使用Python语言调用接口来执行具体的打包任务。具体的打包更新任务管理规则，是由Unity命令实现的打包模块，也就是游戏具体的打包更新任务管理规则。

S230、获取所述打包更新任务的执行状态，并将所述执行状态储存至MySql数据库。

MySql数据库负责存储打包任务的状态，包括任务的版本号，资源版本号，执行任务的log，任务的命令行参数，运行机器的信息等。执行log通常需要定时写入数据库，以便向前端用户及时返回任务执行的状态信息。

在一种可能的实现方式中，可选地，在步骤S200中，所述通过打包任务管理模块接收所述请求指令，并根据所述请求指令执行打包更新任务，实时反馈并储存所述打包更新任务的执行状态数据，还包括：

S211、预设访问规则，并将所述访问规则配置于所述打包任务管理模块；

S212、所述打包任务管理模块基于所述访问规则，发送所述打包更新任务的执行状态请求指令至所述MySql数据库；

S213、所述MySql数据库返回所述打包更新任务的执行状态，发送至所述终端并展示。

打包任务管理模块需要通过PHP脚本访问MySql数据库来获取任务的状态，因此，需要为打包任务管理模块配置相应的访问MySql数据库的访问规则。访问规则根据项目运营方根据项目或者游戏更新规则进行设定即可。

在执行打包任务，得到更新包后，上传到Ftp存储，MySql数据库存储打包版本记录的过程。

每次打包任务的日志是整个串行化流程的日志，在执行远程控制，执行Unity命令，Ftp操作等过程中，每个步骤都记录执行命令参数，执行过程，执行结果等信息，如果某一步骤执行失败时，命令会记录失败原因并中断整个打包流程。

在一种可能的实现方式中，可选地，在步骤S200中，所述通过打包任务管理模块接收所述请求指令，并根据所述请求指令执行打包更新任务，实时反馈并储存所述打包更新任务的执行状态数据，还包括：

S221、预设调用规则，并将所述调用规则配置于所述打包任务管理模块；

S222、所述打包任务管理模块基于所述调用规则，调用执行所述请求指令的任务执行接口，其中，所述任务执行接口为Python的SVN操作接口、Python的文件操作接口、Python的远程管理接口或Python的FTP接口中的至少一种；

S223、通过调用的任务执行接口，执行所述请求指令所请求的打包更新任务。

如图2所示，本技术，需要通过打包任务管理模块负责对所有的打包任务进行管理，除了通过PHP脚本访问MySql数据库来获取任务的状态以外，还使用Python语言调用接口来执行具体的打包任务。Python提供的接口有SVN操作，文件操作，远程管理，FTP操作这几类。具体的打包任务则是由Unity命令实现的打包模块，也就是游戏具体的打包规则，如何生成打包资源和App。

每次打包任务的日志是整个串行化流程的日志，在执行远程控制，执行Unity命令，Ftp操作等过程中，每个步骤都记录执行命令参数，执行过程，执行结果等信息，如果某一步骤执行失败时，命令会记录失败原因并中断整个打包流程。

运营方，提前为打包任务管理模块配置所述调用规则，便于在打包任务管理模块获知请求指令时，能够基于所述调用规则，调用与该请求指令向匹配的任务执行端口，通过该匹配的任务执行端口，连接对应的任务执行模块，比如Unity打包模块。

在一种可能的实现方式中，可选地，在步骤S200中，所述通过打包任务管理模块接收所述请求指令，并根据所述请求指令执行打包更新任务，实时反馈并储存所述打包更新任务的执行状态数据，还包括：

S224、获取执行打包更新任务过程中的任务执行日志数据，以此作为所述打包更新任务的执行状态数据；

S225、将所述打包更新任务的执行状态数据，记录到所述打包任务管理模块。

如图2所示，在每次打包更新任务执行结束，需要通过打包任务管理模块记录执行任务的日志数据，以此来记录打包更新任务的执行状态，这些数据将作为执行状态数据，并通过打包任务管理模块传输至MySql数据库进行保存。

可以通过打包任务管理模块从MySql数据库中带偶去数据，并返回给用户终端进行展示，向用户展示更新任务的执行进度。

在一种可能的实现方式中，可选地，在步骤S222中，所述调用执行所述请求指令的任务执行接口，包括：

S2221、根据所述请求指令，获取所述请求指令的请求类型；

S2222、根据所述调用规则，调用并获得与所述请求类型对应匹配的任务执行接口；

S2223、通过所匹配的任务执行接口，完成所述打包任务管理模块与任务执行模块之间的连接。

在调用任务执行接口时，需要获取用户所发送的所述请求指令的任务请求类型比如是游戏更新请求或者游戏下载打包请求，打包任务管理模块根据获取的任务请求类型，在根据配置的调用规则调用并获得与所述请求类型对应匹配的任务执行接口；通过所匹配的任务执行接口，完成所述打包任务管理模块与任务执行模块之间的连接，并驱动执行该请求的任务。

在一种可能的实现方式中，可选地，在步骤S200中，所述通过打包任务管理模块接收所述请求指令，并根据所述请求指令执行打包更新任务，实时反馈并储存所述打包更新任务的执行状态数据，还包括：

S231、根据所述请求指令，获取所述请求指令的任务信息，其中，所述任务信息包括任务的版本号、资源版本号、执行任务的log、任务的命令行参数或运行机器的信息中的至少一种；

S232、将所述请求指令的任务信息记录并保存到所述MySql数据库中；

S233、定时获取打包任务执行过程中的log，并返回至所述终端。

MySql数据库存储模块负责存储打包任务的状态，包括任务的版本号，资源版本号，执行任务的log，任务的命令行参数，运行机器的信息等。执行log通常需要定时写入数据库，以便向前端用户及时返回任务执行的状态信息。

因此，在执行任务前，需要把这个打包任务的版本号，参数，记录到MySql数据库中，并定时将打包执行的log写入数据库中，返回给用户。

在打包执行过程中，如果打包执行失败，则打包任务停止运行，返回结果和执行log给用户界面，并写入MySql数据库；如果打包执行成功，则调用Python的FTP操作模块，将生成的打包结果按版本号命名，上传到用来存储备份的远程机器上。

下面将简述下几个打包场景的执行过程：

场景一、

当打包任务管理模块接收到前端的一个"我要执行一个Android平台打包任务"的命令时：

（1）把这个打包任务的版本号，参数，记录到MySql数据库中，并定时将打包执行的log写入数据库中，返回给用户；

（2）调用Python的SVN操作接口，取得客户端指定版本的美术资源和程序代码；

（3）调用Python的文件操作接口，清空要生成打包文件的目标目录和临时目录；

（4）调用Unity命令，执行Unity的打包模块功能；

（5）如果打包执行失败，则打包任务停止运行，返回结果和执行log给用户界面，并写入MySql数据库；

（6）如果打包执行成功，则调用Python的FTP操作模块，将生成的打包结果按版本号命名，上传到用来存储备份的远程机器上。

场景二、

当打包任务管理模块接收到前端的一个"我要执行一个IOS平台打包任务"的命令时：

（1）把这个打包任务的版本号，参数，记录到MySql数据库中，并定时将打包执行的log写入数据库中，返回给用户；

（2）由于IOS打包需要MacOS和XCode，所以调用Python的远程管理模块，远程登录MacOS机器；

（3）在登录的MacOS机器上，通过Shell命令执行SVN和文件操作，调用Unity命令执行Unity的打包模块功能；

（4）如果打包执行失败，则退出MacOS登录，打包任务停止运行，返回结果和执行log给用户界面，并写入MySql数据库；

（5）如果打包执行成功，则退出MacOS登录，调用Python的FTP操作模块，将生成的打包结果按版本号命名，上传到用来存储备份的远程机器上。

场景三、

当打包任务管理模块接收到前端的一个"我要执行一次资源热更新的任务"的命令时：

（1）在MySql数据库中查找此App版本下上一个资源版本的信息，如c版本号等；

（2）将本次打包的SVN版本号和上一个资源版本的SVN版本号进行比对，如果版本一致不需要更新则返回，任务结束；

（3）如果需要执行热更新，调用Unity命令，执行Unity的打包资源的功能，生成新的打包资源；

（4）调用Python的FTP命令，从远程备份机器上获取上一个版本的打包资源；

（5）将生成的打包资源和上一个版本的打包资源进行对比，调用Python的文件操作执行将增量文件拷贝到指定目录，作为更新的美术资源；

（6）调用其他压缩工具生成最终的增量更新文件。

其中，实施本技术的部署环境可以是:Windows，MacOS(用作Unity的ios版本打包)；

1、前端Web页面:WampServer(ApacheWeb服务器、PHP解释器以及MySQL数据库的软件包) Laravel(一个PHP的MVC框架)

2、MySql数据库，存储打包版本信息。

3、打包功能模块:

bat批处理在Windows下实现Unity资源打包和App打包(PC和Android平台)；

Shell脚本实现App打包(IOS平台)。

4、Python打包功能主模块，负责打包任务管理，Svn操作，Ftp操作，文件操作，和SSH远程管理。

其中，打包机上部署的Python节点系统，执行Unity的命令实现打包的方法，其他打包系统也可以使用。

需要说明的是，尽管以游戏打包更新作为示例介绍了如上本技术的实施，但本领域技术人员能够理解，本公开应不限于此。事实上，用户完全可根据个人喜好和/或实际应用场景灵活设定数据处理类型，只要能够实现线上打包更新即可。

本技术为一种打包更新的数据处理方式，不仅仅可以用在游戏数据的处理上，还可以用在其他线上任务的更新项目上，具体不做限制。

这样，根据移动端项目从内部测试到上线过程中的实际需要，在测试阶段就按照发布基础包加多个更新包的模式，灵活地在打包机之间完成通信，存储，日志记录，版本管理，分支管理，自定义显示等功能，降低部署的难度，在构建过程中发生的错误可以及时反馈处理，提高游戏项目迭代开发的效率。

实施例2

基于实施例1的实施，本实施，根据本公开的另一方面，提供了一种实现上述所述的基于版本管理的数据打包更新方法的装置，包括：

终端，用于发送获取数据打包更新的请求指令；

打包任务管理模块，用于接收所述请求指令，根据所述请求指令执行打包更新任务，实时反馈所述打包更新任务的执行状态数据至所述终端；

MySql数据库，用于实时储存所述打包更新任务的执行状态数据。

在一种可能的实现方式中，可选地，所述终端，还用于接收所述执行状态数据，并展示所述执行状态。

各个模块/硬件的功能和实施原理，具体参见上述实施例的描述，本处不再赘述。

如图2所示，本实施例，终端采用前端Web页面的方式实施。其具体实施方案不做限制。

显然，本领域的技术人员应该明白，上述的本发明的各模块或各步骤可以用通用的计算装置来实现，它们可以集中在单个的计算装置上，或者分布在多个计算装置所组成的网络上，可选地，它们可以用计算装置可执行的程序代码来实现，从而，可以将它们存储在存储装置中由计算装置来执行，或者将它们分别制作成各个集成电路模块，或者将它们中的多个模块或步骤制作成单个集成电路模块来实现。这样，本发明不限制于任何特定的硬件和软件结合。

实施例3

更进一步地，根据本公开的另一方面，根据本公开的另一方面，还提供了一种数据更新系统，包括：

处理器；

用于存储处理器可执行指令的存储器；

其中，所述处理器被配置为执行所述可执行指令时实现上述所述的基于版本管理的数据打包更新方法。

本公开实施例数据更新系统包括处理器以及用于存储处理器可执行指令的存储器。其中，处理器被配置为执行可执行指令时实现前面任一所述的一种基于版本管理的数据打包更新方法。

此处，应当指出的是，处理器的个数可以为一个或多个。同时，在本公开实施例的数据更新系统中，还可以包括输入装置和输出装置。其中，处理器、存储器、输入装置和输出装置之间可以通过总线连接，也可以通过其他方式连接，此处不进行具体限定。

存储器作为一种计算机可读存储介质，可用于存储软件程序、计算机可执行程序和各种模块，如：本公开实施例的一种基于版本管理的数据打包更新方法所对应的程序或模块。处理器通过运行存储在存储器中的软件程序或模块，从而执行数据更新系统的各种功能应用及数据处理。

输入装置可用于接收输入的数字或信号。其中，信号可以为产生与设备/终端/服务器的用户设置以及功能控制有关的键信号。输出装置可以包括显示屏等显示设备。

以上已经描述了本公开的各实施例，上述说明是示例性的，并非穷尽性的，并且也不限于所披露的各实施例。在不偏离所说明的各实施例的范围和精神的情况下，对于本技术领域的普通技术人员来说许多修改和变更都是显而易见的。本文中所用术语的选择，旨在最好地解释各实施例的原理、实际应用或对市场中的技术的技术改进，或者使本技术领域的其它普通技术人员能理解本文披露的各实施例。

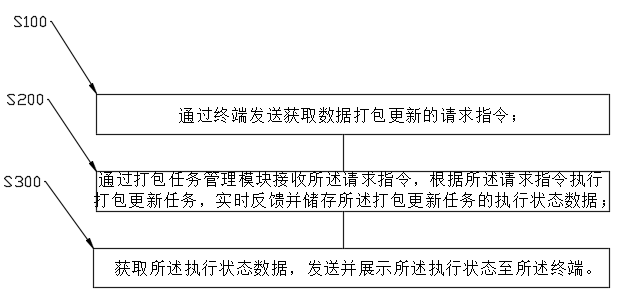


图1

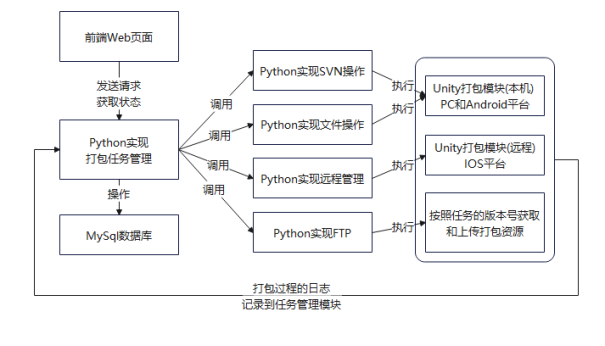


图2