(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 102707971 A
(43)申请公布日 2012.10.03

(21)申请号 201210125940.4

(22)申请日 2012.04.26

(71)申请人 广东电子工业研究院有限公司
地址 523808 广东省东莞市松山湖科技产业园区松科苑 10 号楼

(72)发明人 莫展鹏 季统凯 岳强

(74)专利代理机构 北京科亿知识产权代理事务所(普通合伙) 11350
代理人 汤东凤

(51)Int.Cl.
G06F 9/445(2006.01)

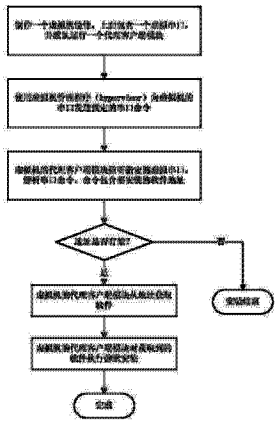
权利要求书 1 页 说明书 9 页 附图 2 页

(54)发明名称

一种在虚拟机中自动获取和安装软件的方法

(57)摘要

本发明涉及软件获取与安装技术领域，特别是一种在虚拟机中自动获取和安装软件的方法。本发明首先制作一个虚拟机镜像，里面默认开通一个虚拟串口，并且设置开机运行一个代理客户端模块；虚拟机管理程序向虚拟串口发送软件安装的命令；虚拟机上的代理客户端模块监听虚拟串口，接收并解析串口命令，命令中包含了要安装软件的获取地址；如果地址有效的话，那么代理客户端模块就获取软件并执行静默安装，否则不安装。本发明解决了为满足不同的业务需求而需要制作大量虚拟机镜像、获取软件而把病毒和恶意代码引入虚拟机及软件安装复杂等问题；可应用于虚拟机的软件获取与安装上。



CN 102707971 A

1. 一种在虚拟机中自动获取和安装软件的方法，其特征在于：包括如下步骤：

步骤 1：制作一个虚拟机镜像，上面包含一个虚拟串口，并默认运行一个代理客户端模块；

步骤 2：向虚拟机的串口发送预定的串口命令；

步骤 3：虚拟机的代理客户端模块监听指定的虚拟串口，解析包含所要安装软件地址的串口命令；如果解析出来的软件地址有效，执行步骤 4；如果解析出来的命令不正确或软件地址无效，执行步骤 6；

步骤 4：虚拟机的代理客户端模块从地址获取软件；

步骤 5：虚拟机的代理客户端模块对获取到的软件执行静默安装；

步骤 6：安装结束。

2. 根据权利要求 1 所述的在虚拟机中自动获取和安装软件的方法，其特征在于：所述代理客户端模块包括串口命令处理模块、软件获取模块和软件静默安装模块，当虚拟机进入操作系统以后自动运行；

所述串口命令处理模块是一个跨平台的模块，接收特定串口的命令，并且负责解析串口命令，判断命令是否符合约定的格式以及软件地址是否能被访问；

所述软件获取模块是一个从软件库中下载程序到虚拟机上的模块，软件库可由 FTP 或 HTTP 服务器提供；

所述软件静默安装模块是一个按照预先设置的值自动执行，不需要人工干预去安装软件的模块；

所述软件静默安装模块是一个同时满足 Windows 虚拟机与 Linux 虚拟机的静默安装模块。

3. 根据权利要求 1 所述的在虚拟机中自动获取与自动安装软件的方法，其特征在于：所述的串口命令格式为类型 + 分隔符 (::)+URI，其中类型部分预留给模块扩展使用。

4. 根据权利要求 2 所述的在虚拟机中自动获取与自动安装软件的方法，其特征在于：所述的串口命令格式为类型 + 分隔符 (::)+URI，其中类型部分预留给模块扩展使用。

5. 根据权利要求 1 至 4 任一项所述的在虚拟机中自动获取与自动安装软件的方法，其特征在于：向虚拟机串口发送的软件获取和安装的命令直接写入理机的设备文件。

6. 根据权利要求 2 或 4 所述的在虚拟机中自动获取与自动安装软件的方法，其特征在于：软件静默安装模块自动适配操作系统的类型，根据类型的不同，Windows 操作系统调用 Windows Installer 的 msiexec.exe 命令，Linux 操作系统调用 rpm 命令或 deb 命令完成软件的静默安装。

7. 根据权利要求 5 所述的所述的在虚拟机中自动获取与自动安装软件的方法，其特征在于：软件静默安装模块自动适配操作系统的类型，根据类型的不同，Windows 操作系统调用 Windows Installer 的 msiexec.exe 命令，Linux 操作系统调用 rpm 命令或 deb 命令完成软件的静默安装。

一种在虚拟机中自动获取和安装软件的方法

[0001]

技术领域

[0002] 本发明涉及软件获取与安装技术领域，特别是一种在虚拟机中自动获取和安装软件的方法。

背景技术

[0003] 云计算对于虚拟机的应用是通过预先安装软件，制作好虚拟机镜像模板以达到快速、批量生成虚拟机的目的。在面临操作系统与软件众多，且两者之间可通过自由搭配来提供服务的应用场景时，需要预先制作的虚拟机模板数量就比较多，且会带来以下问题：

一是镜像扩展不方便，增加一个软件时，那么可能就需要重新制作与操作系统数量相匹配的虚拟机镜像；

二是耗时较长，制作镜像时需要在物理机上相应的操作系统里先做好对应软件的安装，然后通过 P2V 的方式导出虚拟机镜像，在 P2V 成功率无法达到 100% 的情况下，这样做耗费的时间无疑是比较长的；

三是用户获取软件的途径各种各样，获取的软件无法保证其安全性，软件中可能含有恶意代码或病毒，一旦爆发，对云计算中心的网络将造成极大的影响；

四是让用户自行安装软件时可能会发生误操作而导致软件无法正确安装，或者配置有误导致软件无法正常工作。

[0004] 为了提升虚拟机应用的效率，增强虚拟机镜像制作的可扩展性，需要一种通过预先定制虚拟机镜像和相应软件，在虚拟机运行起来后，进行软件自动安装的方式。

发明内容

[0005] 本发明解决的技术问题在于提供一种在虚拟机中自动获取和安装软件的方法，解决了为满足不同的业务需求而需要制作大量虚拟机镜像、获取软件而把病毒和恶意代码引入虚拟机及软件安装复杂等问题。

[0006] 本发明解决上述技术问题的技术方案是：包括如下步骤：

步骤 1：制作一个虚拟机镜像，上面包含一个虚拟串口，并默认运行一个代理客户端模块；

步骤 2：向虚拟机的串口发送预定的串口命令；

步骤 3：虚拟机的代理客户端模块监听指定的虚拟串口，解析包含所要安装软件地址的串口命令；如果解析出来的软件地址有效，执行步骤 4；如果解析出来的命令不正确或软件地址无效，执行步骤 6；

步骤 4：虚拟机的代理客户端模块从地址获取软件；

步骤 5：虚拟机的代理客户端模块对获取到的软件执行静默安装；

步骤 6：安装结束。

[0007] 所述代理客户端模块包括串口命令处理模块、软件获取模块和软件静默安装模块，当虚拟机进入操作系统以后自动运行；

所述串口命令处理模块是一个跨平台的模块，接收特定串口的命令，并且负责解析串口命令，判断命令是否符合约定的格式以及软件地址是否能被访问；

所述软件获取模块是一个从软件库中下载程序到虚拟机上的模块，软件库可由 FTP 或 HTTP 服务器提供；

所述软件静默安装模块是一个按照预先设置的值自动执行，不需要人工干预去安装软件的模块；

所述软件静默安装模块是一个同时满足 Windows 虚拟机与 Linux 虚拟机的静默安装模块。

[0008] 所述的串口命令格式为类型 + 分隔符 (::)+URI，如 10::ftp://192.168.1.2/office.rar，其中类型部分预留给程序扩展使用，目前默认为 10，代表软件地址。

[0009] 向虚拟机串口发送的软件获取和安装的命令直接写入理机的设备文件。

[0010] 软件静默安装模块自动适配操作系统的类型，根据类型的不同，Windows 操作系统调用 Windows Installer 的 msiexec.exe 命令，Linux 操作系统调用 rpm 命令(红帽发行版)或 deb 命令(debian 发行版)命令完成软件的静默安装。

[0011] 本发明通过串口命令按需控制虚拟机自动获取与自动安装软件，可以从一个基本的虚拟机镜像自动按需生成满足不同业务需求的虚拟机，而不需要去手动制定大量的镜像，同时在大量虚拟机需要统一安装补丁的时候，本发明也提供了很大的便利性。本发明的虚拟机从指定软件库获取所需软件，各种软件可以在一个地方统一管理，降低了虚拟机中安装软件而引入恶意代码和病毒的风险。本发明向虚拟机串口发送的软件获取和安装的命令直接通过写物理机的设备文件来实现，不受网络环境的影响，区别于一般的软件安装和升级程序受限于网络的缺点。本发明能自动适配操作系统的类型，区别于一般的软件安装和升级程序只能单一用于 Windows 或 Linux 的限制。

附图说明

[0012] 下面结合附图对本发明进一步说明：

图 1 为本发明方法流程图；

图 2 为本发明代理客户端模块结构框图。

具体实施方式

[0013] 如图 1 所示，先制作一个虚拟机镜像，上面包含一个虚拟串口，并默认运行一个代理客户端模块，本过程可以由 libvirt.xml 中定义的一个虚拟串口来实现，定义如下：

```
<console type='pty' tty='/dev/pts'>
      <source path='/dev/pts'/>
      <target type='serial' port='0'/>
    </console>
```

这里定义了一个串口，端口号是 0，表示 COMM1，并且映射到了物理机的 /dev/pts 设备文件上；

代理客户端模块中的串口命令处理模块接收串口命令并解析其内容，本程序可由 Java Communications API 实现，代码如下：

```
        /**
         * 初始化串口监听
         *
         * @throws Exception
         */
        public SerialMonitor init() throws Exception
        {
              try {
                    portId = CommPortIdentifier.getPortIdentifier(“COMM1”);
                      serialPort = (SerialPort) portId.open(“SerialMonitor”,
2000);
                    System.out.println("Start to monitor data from serial port
"
                                + portName);
                      log.info("Start to monitor data from serial port " +
portName);
                    inStream = serialPort.getInputStream();
                     isPortOpen = true;
              }
              catch (NoSuchPortException ex) {
                    throw new Exception(ex.toString());
              }
              catch (PortInUseException ex) {
                    throw new Exception(ex.toString());
              }
              return this;
        }
    /**
          * 处理接收到的数据，分为 3 部分，一是类别，标识符分别为 10（软件）、11（数
据）
         * 二是软件或数据源地址，三是软件的默认安装命令，若没有则以 N 替代
         */
        public void readData()
        {
              try {
                    byte[] readBuffer = new byte[512];
                    int i = 0;
```

```
                while (inStream.available() > 0) {
                      i = inStream.read(readBuffer);
                }
                String command = new String(readBuffer, 0, i - 1);
                log.info("Receive Command:\n" + command);
                String[] comms = command.split("::");
                        DataType type = DataType.getType(Integer.
parseInt(comms[0]));
                String params=comms[2];
                DataRetrieve dataRetieve = new DataRetrieve(type,comms[1],p
arams );
                dataRetieve.retrieveData();
            }
            catch (Exception e) {
                log.error(e);
                e.printStackTrace();
            }
        }
    /**
         * 解析 URI,分解出协议头、地址、端口、文件名、账号等信息
         */
        private void parseURI()
        {
            //若为 ftp
            if(uri.startsWith("ftp"))
            {
                    String account=uri.substring(uri.indexOf("//")+2, uri.
indexOf("@"));
                String[] accountArr=account.split(":");
                userName=accountArr[0];
                password=accountArr[1];
                String path=uri.substring(uri.indexOf("@")+1);
                String url=path.substring(0,path.indexOf("/"));
                String[] urlArr=url.split(":");
                host=urlArr[0];
                if(urlArr.length>1)
                {
                    PORT=Integer.parseInt(urlArr[1]);
                }
```

```
                ftPath=path.substring(path.indexOf("/")+1);
                fileName=path.substring(path.lastIndexOf("/")+1);
                localPath=Main.Default_Directory+fileName;
            }
            //处理 HTTP 协议请求的数据
            else if(uri.startsWith("http"))
            {
            }
        }
```

代理客户端模块的软件获取模块从串口命令输入的软件地址获取软件，本程序可由 Netty Client API 实现，代码如下：

```
    /**
     * 获取数据
     * @throws Exception
     */
    public String retrieveData() throws Exception
    {
        try {
                client.download(ftPath, new File(localPath),new
DataTransferListener());
            if(type==DataType.SOFTWARE)
            {
                //执行下一步安装操作
                  new SoftwareDeploy(localPath,fileName,installParam
s);
            }
            return localPath;
        }
        catch (Exception e) {
            e.printStackTrace();
            throw new Exception(e);
        }
    }
```

代理客户端的软件静默安装模块对下载的软件进行自动安装，针对 Linux 系统，调用 rpm 命令或 deb 命令进行静默安装；针对 Windows 系统，分别针对 zip 压缩文件与 exe 文件进行不同的处理，代码如下：

```
    /**
     * 处理压缩文件
     */
```

```
        private void processCompressedFile(File destDir)
        {
              StringBuffer comm = new StringBuffer("\"").append(
                        Main.Default_Directory).append("WinRar\\");
              //.zip 和 .rar 压缩文件解压命令稍有不同
              if (filename.endsWith(".zip")) {
                    comm.append("WinRar.exe\" ");
              }
              else if(filename.endsWith(".rar")) {
                    comm.append("Rar.exe\" ");
              }
              comm.append("x -o+ \"").append(path).append("\" \"")
                        .append(destDir.getAbsolutePath()).append("\\\"");
              System.out.println(comm.toString());
              int exitVal = 0;
              try {
                    Runtime tr = Runtime.getRuntime();
                    Process proc = tr.exec(comm.toString());
                    InputStream es = proc.getErrorStream();
                    InputStreamReader reader = new InputStreamReader(es);
                    BufferedReader br = new BufferedReader(reader);
                    StringBuffer sb = new StringBuffer();
                    String l = "";
                    while ((l = br.readLine()) != null) {
                          sb.append(l + "\n");
                    }
                    log.error(sb.toString());
                    exitVal = proc.waitFor();
                    if (proc.waitFor() == 0) {
                          if (proc.exitValue() == 0) {
                                log.info("Uncompressing complete...");
                          }
                    }
                    // 查找并启动 exe 文件
                    File exeFile=findExeFile(destDir);
                              Process proc2=tr.exec("\""+exeFile.
getAbsolutePath()+"\"");
                    proc2.waitFor();
              }
```

```
            catch (IOException e) {
                  log.error(e);
                  e.printStackTrace();
            }
            catch (InterruptedException e) {
                  e.printStackTrace();
                  log.error(e);
            }
            System.out.println(exitVal);
      }
      /**
       * 处理可执行文件
       */
      private void processExecutableFile() {
            String command=path+" "+installParams;
            try {
                  Runtime rt=Runtime.getRuntime();
                  Process proc=rt.exec(command);
                  proc.waitFor();
                  log.info("Software "+filename+" install succefully!");
            }
            catch (Exception e) {
                  log.error(e);
                  e.printStackTrace();
            }
      }
      /**
       * 查找给定目录中的第一个 exe 文件,若无则返回 null
       * @param dir
       * @return
       */
      private File findExeFile(File dir) {
            File[] children=dir.listFiles();
            boolean isFind=false;
            for(File file:children) {
                  if(isFind) {
                        break;
                  }
                  if(file.isDirectory()) {
```

```
                    findExeFile(file);
                }
                else if(file.isFile())
                {
                    if(file.getAbsolutePath().endsWith(".exe")) {
                        exeFile=file;
                    }
                }
            }
            return exeFile;
        }
```

然后，向虚拟机的串口发送预定的串口命令，在宿主机上向虚拟机发送命令，本程序可用 C 语言在 Linux 下写串口设备的文件，代码如下：

```
/** 打开串口，dev 串口设备名， mode 打开方式，**/
    int opendev(char *dev,mode_t mode)
    {
        int fd;
        fd = open(dev, mode);
        if (-1 == fd){
            perror("Can't Open Serial Port");
            return -1;
        }
        else{
            fcntl(fd, F_SETFL, FNDELAY);
            return fd;
        }
    }
/** 写串口 **/
    #define RSDEV_NAME "/dev/ttyS1"
    int main(void)
    {
        int rsfd = 0;
        int nwrite;
        char input_buf[64];
        rsfd = opendev(RSDEV_NAME,O_RDWR | O_NOCTTY | O_NDELAY);
        if(rsfd < 0){
            printf("open error:/n");
            exit(-1);
        }
```

```
                set_speed(rsfd,9600);     /* 设置速率 B9600*/
                if (set_parity(rsfd,8,1,'N') == FALSE){ /*8位数据位,一位停止位
*/
                        printf("Set Parity Error/n");
                        exit (-1);
                }
                while(1)
                {
                        fgets(input_buf,sizeof(input_buf),stdin);
                        printf("input_buf = %s", input_buf);
                        nwrite = write(rsfd, input_buf, strlen(input_buf));
                        if ( nwrite == -1 ) {
                             perror("ERROR!");
                             exit(1);
                        }
                        else {
                             printf("ret=%d/n", nwrite);
                        }
                }
        }
```

然后,虚拟机的代理客户端模块监听指定的虚拟串口,解析串口命令,此操作由上述串口命令处理程序完成;

然后,虚拟机的代理客户端模块从地址获取软件,此操作由上述软件获取模块完成;

最后,虚拟机的代理客户端模块对获取到的软件执行静默安装,此操作由上述软件静默安装模块完成。

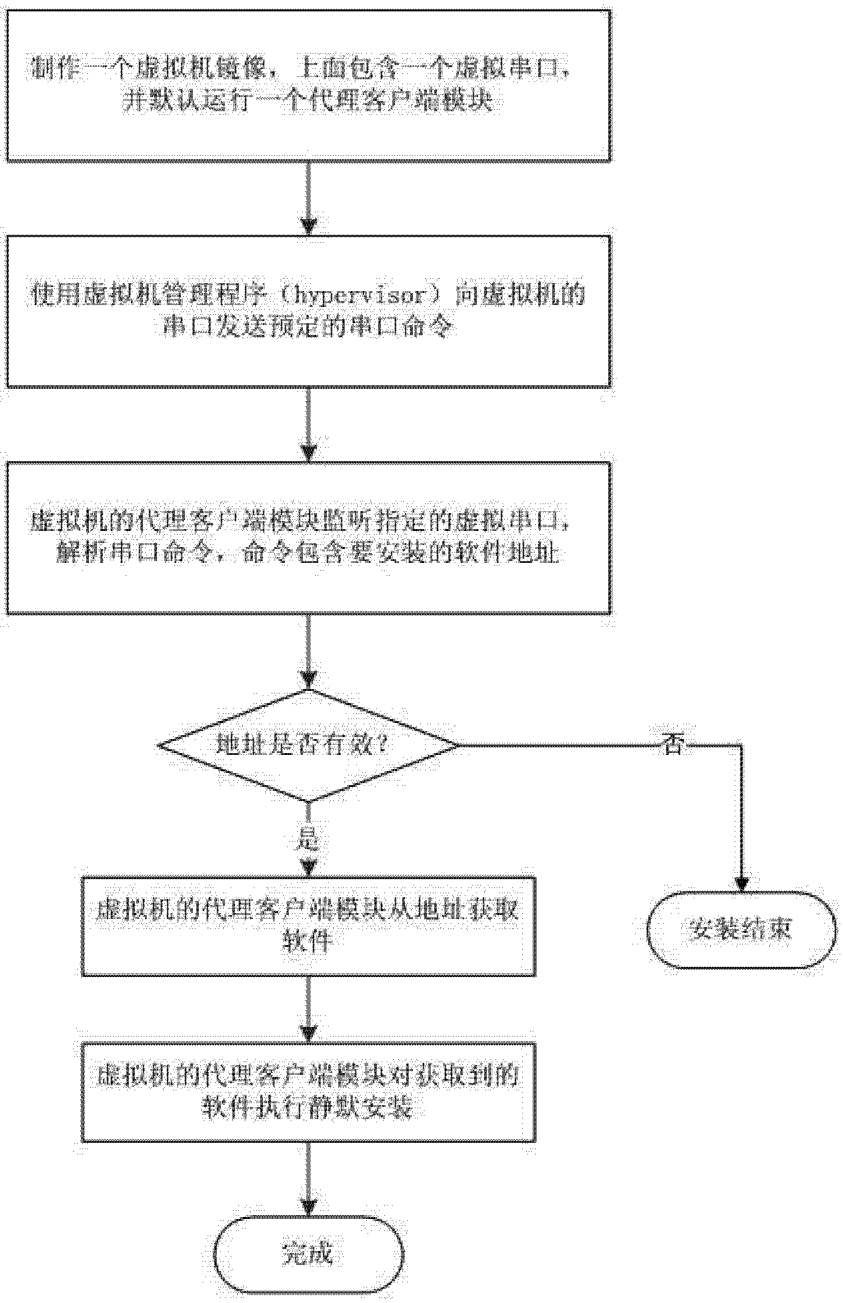

制作一个虚拟机镜像，上面包含一个虚拟串口，并默认运行一个代理客户端模块
使用虚拟机管理程序（hypervisor）向虚拟机的串口发送预定的串口命令
虚拟机的代理客户端模块监听指定的虚拟串口，解析串口命令，命令包含要安装的软件地址
地址是否有效？
否
是
安装结束
虚拟机的代理客户端模块从地址获取软件
虚拟机的代理客户端模块对获取到的软件执行静默安装
完成

图 1

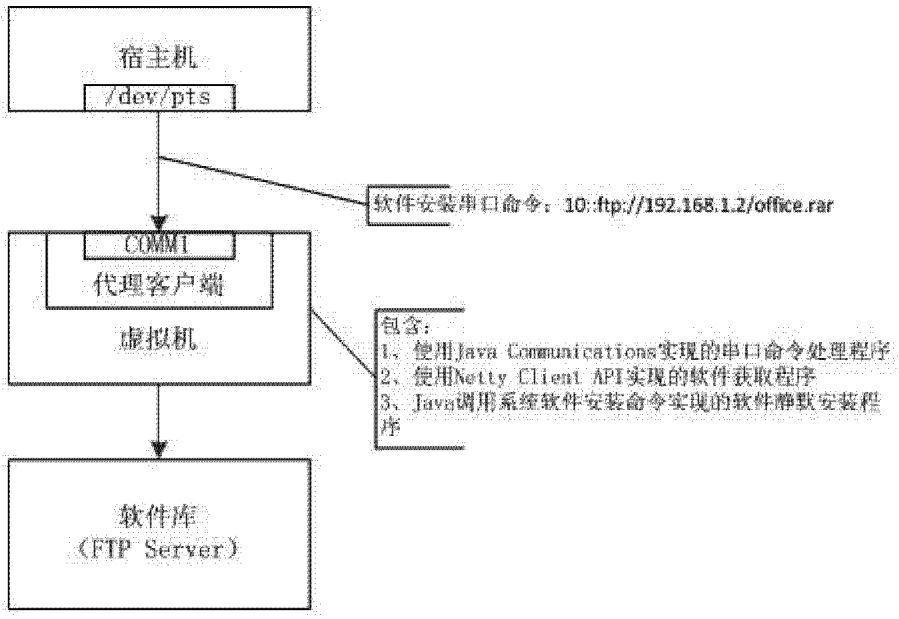

宿主机
/dev/pts
软件安装串口命令：10:ftp://192.168.1.2/office.rar
COMM1
代理客户端
虚拟机
包含：
1、使用Java Communications实现的串口命令处理程序
2、使用Netty Client API实现的软件获取程序
3、Java调用系统软件安装命令实现的软件静默安装程序
软件库
(FTP Server)

图 2