(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
30 January 2014 (30.01.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/015753 A1

(51) International Patent Classification:
G06F 11/00 (2006.01)

(21) International Application Number:
PCT/CN2013/079482

(22) International Filing Date:
16 July 2013 (16.07.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
201210264230.X 27 July 2012 (27.07.2012) CN

(71) Applicant: **TENCENT TECHNOLOGY (SHENZHEN) COMPANY LIMITED** [CN/CN]; Room 403, East Block 2, SEG Park, Zhenxing Road, Futian District, Shenzhen, Guangdong 518044 (CN).

(72) Inventor: **LI, Weijie**; Room 403, East Block 2, SEG Park, Zhenxing Road, Futian District, Shenzhen, Guangdong 518044 (CN).

(74) Agent: **ADVANCE CHINA I.P.LAW OFFICE**; Suite 918-920, Dongshan Plaza, No.69 Xianlie Central Road, Guangzhou City, Guangdong 510095 (CN).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

(54) Title: METHOD AND APPARATUS FOR INTERCEPTING OR CLEANING-UP PLUGINS

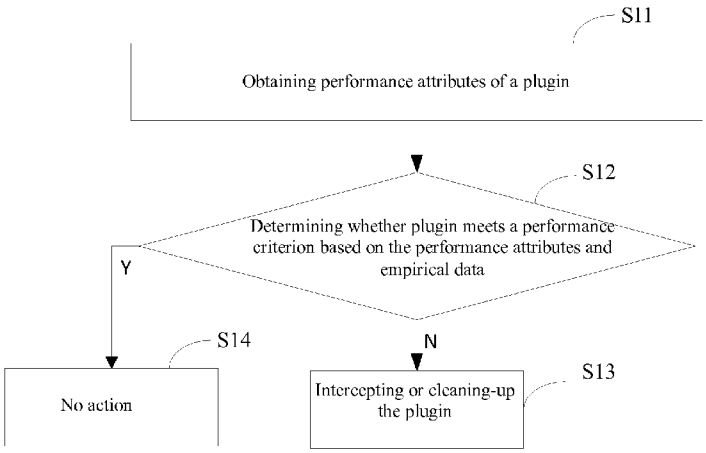

Figure 1

(57) Abstract: A method and apparatus for intercepting or cleaning-up plugins are provided. The method may include: obtaining performance attributes of a plugin (S11); determining if the plugin meets a performance criterion based on the obtained performance attributes and empirical data(S12); and intercepting or cleaning-up the plugin if the plugin does not meet the performance criterion(S13). In accordance with embodiments of the present invention, the performance attributes of a plugin can be evaluated to determine whether the plugin meets a preset performance criterion, and the plugin can then be processed according to the result of the determination.

WO 2014/015753 A1

METHOD AND APPARATUS FOR INTERCEPTING OR CLEANING-UP PLUGINS

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims the benefit and priority of Chinese Patent Application No. 201210264230.X, entitled "Method and Apparatus for Intercepting or Cleaning-up Plugins," filed on July 27, 2012. The entire disclosures of each of the above applications are incorporated herein by reference.

TECHNICAL FIELD

The present invention relates to plugin detection technologies, and more particularly to a method and apparatus for intercepting or cleaning-up plugins.

BACKGROUND

In the prior art, plugin interception or cleanup is generally based on the malicious acts or reputations of the plugins. For example, malicious plugin like Adware or Spyware monitors the internet activities of user terminals, and sends the recorded data to a remote monitoring center for the purposes of advertising or stealing game or bank account passwords, which causes severe adverse effects on user terminals.

However, some plugins are not malicious, have some useful functions, but may consume large amount of resources at the user terminal when they are running. Such plugins are usually not blocked or cleaned-up by security software, and are not easily found by the user terminal.

Since the current security software only blocks or cleans-up plugin based on maliciousness, it may miss some plugins, which could affect the stability of the user terminal system.

SUMMARY OF THE INVENTION

The present invention aims to provide a method for intercepting or cleaning-up plugins to ensure a more complete interception and cleanup of the plugins, and enhances the stability of the system.

In accordance with embodiments of the present invention, a method for intercepting or cleaning-up a plugin is provided, the method comprising: obtaining performance attributes of a plugin; determining whether the plugin meets a performance criterion based on the performance

attributes and empirical data; and intercepting or cleaning-up the plugin if the plugin does not meet the performance criterion.

Preferably, the method further comprises monitoring activity of the plugin in a user terminal, wherein the activity is selected from a group comprising installing, loading, and running of the plugin; identifying the plugin based on the monitored activity; and obtaining performance attributes of the plugin when the plugin is identified.

Preferably, the method further comprises, after identifying the plugin based on the monitored activity, evaluating performance attributes of the plugin when the plugin is not identified; determining whether the plugin meets a performance criterion based on the performance attributes and empirical data; and intercepting or cleaning-up the plugin if the plugin does not meet the performance criterion.

Preferably, the method further comprises, prior to obtaining performance attributes of the plugin, obtaining empirical data of the plugin, wherein the empirical data comprises plugin importance data, plugin performance data, and/or end user review data; and determining whether the plugin meets the performance criterion based on the empirical data.

Preferably, the empirical data obtained through plugin testing, backend collection, and/or web crawling, backend collection comprises frontend collection and end user review, frontend collection comprises hook mode and notification mode, plug testing comprises testing the plugin in an automatic module to obtain information on the plugin's CPU usage, load time and/or memory usage; hook mode comprises using a hook to obtain a start time and an end time for loading the plugin, a start time and an end time for running the plugin, and information on the plug's CPU usage and/or memory usage while the plugin is loading and running; reception mode comprises a client directly notifying starting and ending of loading and running of the plugin; end user review comprises collecting reviews of end users on the performance of the plugin; web crawling comprises crawling search engine data and/or website review data to evaluate the performance of the plugin; and the empirical data is stored in a remote cloud database or a local database.

Preferably, determining if the plugin meets the performance criterion based on the performance attributes and empirical data comprises: matching the performance attributes with empirical data stored on a cloud database; obtaining a preset determination result after a successful match, and determining whether the plugin meets the performance criterion based on the preset determination result.

In accordance with embodiments of the present invention, an apparatus for intercepting or cleaning-up a plugin is provided, the apparatus comprising: a performance acquisition module for

obtaining performance attributes of a plugin; a first performance determination module for determining whether the plugin meets a performance criterion based on the performance attributes and empirical data; and a plugin processing module for intercepting or cleaning-up the plugin if the plugin does not meet the performance criterion.

Preferably, the apparatus further comprises a plugin monitoring module for monitoring activity of the plugin in a user terminal, wherein the activity is selected from a group comprising installing, loading, and running of the plugin; identifying the plugin based on the monitored activity; and obtaining performance attributes of the plugin when the plugin is identified.

Preferably, the apparatus further comprises a performance evaluation module for, after identifying the plugin based on the monitored activity, evaluating performance attributes of the plugin when the plugin is not identified; wherein the first performance determination module is configured for determining whether the plugin meets a performance criterion based on the performance attributes and empirical data; and the plugin processing module is configured for intercepting or cleaning-up the plugin if the plugin does not meet the performance criterion.

Preferably, the apparatus further comprises an empirical data acquisition module for obtaining empirical data of the plugin, wherein the empirical data comprises plugin importance data, plugin performance data, and/or end user review data; and a second performance determination module for determining whether the plugin meets a performance criterion based on the empirical data.

Preferably, the empirical data obtained through plugin testing, backend collection, and/or web crawling, wherein backend collection comprises frontend collection and end user review, frontend collection comprises hook mode and notification mode, plug testing comprises testing the plugin in an automatic module to obtain information on the plugin's CPU usage, load time and/or memory usage; hook mode comprises using a hook to obtain a start time and an end time for loading the plugin, a start time and an end time for running the plugin, and information on the plug's CPU usage and/or memory usage while the plugin is loading and running; reception mode comprises a client directly notifying starting and ending of loading and running of the plugin; end user review comprises collecting reviews of end users on the performance of the plugin; web crawling comprises crawling search engine data and/or website review data to evaluate the performance of the plugin; and the empirical data is stored in a remote cloud database or a local database.

Preferably, the first performance determination module is further configured for: matching the performance attributes with empirical data stored on a cloud database; obtaining a preset determination result after a successful match, and determining whether the plugin meets the performance criterion based on the preset determination result.

In accordance with embodiments of the present invention, the performance attributes of a plugin can be evaluated to determine whether the plugin meets a preset performance criterion, and the plugin can then be processed according to the result of the determination. The embodiments of the present invention are primarily directed to evaluate and process plugins that are being installed, loaded, or running: if the plugin does meet the performance criterion, it will be intercepted if being installed, and cleaned-up if being loaded or running.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is an exemplary flowchart for a method for intercepting or cleaning-up plugins in accordance with an embodiment of the present invention.

Figure 2 is an exemplary flowchart for a method for intercepting or cleaning-up plugins in accordance with another embodiment of the present invention.

Figure 3 is an exemplary flowchart for a method for intercepting or cleaning-up plugins in accordance with yet another embodiment of the present invention.

Figure 4 is an exemplary flowchart for acquiring empirical data in accordance with an embodiment of the present invention.

Figure 5 is an exemplary schematic diagram for relevant components for acquiring empirical data in accordance with an embodiment of the present invention.

Figure 6 is an exemplary flowchart for processing a plugin being installed in accordance with an embodiment of the present invention.

Figure 7 is an exemplary schematic diagram for an interface for intercepting a plugin being installed in accordance with an embodiment of the present invention.

Figure 8 is an exemplary flowchart for processing a plugin being loaded in accordance with an embodiment of the present invention.

Figure 9 is an exemplary schematic diagram for relevant components for processing a plugin being loaded in accordance with an embodiment of the present invention.

Figure 10 is an exemplary flowchart for processing a running plugin in accordance with an embodiment of the present invention.

Figure 11 is an exemplary schematic diagram for relevant components for processing a running plugin in accordance with an embodiment of the present invention.

Figure 12 is an exemplary schematic diagram for an apparatus for intercepting or cleaning-up plugins in accordance with an embodiment of the present invention.

Figure 13 is an exemplary schematic diagram for an apparatus for intercepting or cleaning-up plugins in accordance with another embodiment of the present invention.

Figure 14 is an exemplary schematic diagram for an apparatus for intercepting or cleaning-up plugins in accordance with yet another embodiment of the present invention.

Figure 15 is another exemplary schematic diagram for an apparatus for intercepting or cleaning-up plugins in accordance with an embodiment of the present invention.

To better illustrate the technical features of the embodiments of the present invention, various embodiments of the present invention will be briefly described in conjunction with the accompanying drawings.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

It should be noted that the various embodiments are merely provided to illustrate the present invention, and are not intended to limit the scope of the present invention.

Plugins are common in computer software, and are supported by many types of software. For example, text editor software Ultra Edit and software development tool Visual Studio both support plugins. There are also plugins for browsers, which comes in different forms, such as browser helper object (BHO), ActiveX, Url SearchHook, and ToolBar.

BHO: Browser Helper Objects (BHOs) are applets that automatically run whenever an Internet browser (such as IE) starts. Usually, a BHO file is installed into the user terminal systems by some other software. For example, some Adware with downloading capability may install a BHO file to track the various advertisements while the user terminal surfs the web. BHO usually makes it easier for user terminals to browse the Internet or use various accessory functions, but some BHOs are Adware or Spyware as which monitor the online behavior of user terminals and transmit recorded data to the creator of the BHO. BHO may also create conflicts with other programs that are running, and causes various display and running-time errors that may prevents the normal browsing.

ActiveX: ActiveX plugins, also known as OLE controls or OCX controls, are software components or objects that can be inserted into a webpage or other applications. Typical software needs to be separately downloaded by a user terminal before being installed, while ActiveX plugins will be automatically downloaded by the browser when the user navigates to a particular webpage, and the user will be prompted to install them. ActiveX plug-ins must be downloaded first, authenticated, and eventually confirmed by the user terminal before they can be installed. A

webpage embedded with ActiveX scripts may become very slow, or may even freeze the browser.

Url SearchHook: when a user terminal enters a non-standard Web address in the address bar, such as English characters or Chinese characters, that cannot be successfully interpreted by the address bar, the browser will automatically open a page containing the search result where the string entered by the user is used as the search term, which may help the user to find the needed contents. Url SearchHook object is the plugin that completes such search, usually developed by a third-party company or individual, and installed on the browser as a plugin to help the user to better use the Internet. For example, if the user terminal enters “mobile phone” in the address bar, the search results for “mobile phone” will directly appear. Some companies or individuals may modify a browser's Url SearchHook without the knowledge of the user terminal in order to direct traffic to a particular website or for other commercial purposes.

Toolbar usually refers to accessory tools loaded in a browser, located below the standard browser toolbar. In Internet Explorer, one can right click the blank space in the toolbar to check the installed toolbars, and to display or hide the installed toolbars by making the corresponding selections.

The plugins in accordance with embodiments of the present invention are programs written accordance to certain standard application programming interfaces. For example, after being installed, browser plugins can be called directly by the browser. The plugin can also be a component for certain software features, which can facilitate or impede the user’s use of the software. Performance refers to the specification parameters of a computer, such as speed, which is an important indicator of computer performance. The performance of a computer can typically be evaluated by the CPU usage and the time it takes to complete a task. Inception refers to stopping the installation or loading of a plugin. Clean-up refers to the removal of the plugin from the computer.

Windows-NT-family Host Intrusion Prevention System(WinHIPS)is a kernel driver which uses techniques such as filtering and system call hooking to implement intelligence interception and safeguard computer systems.

As shown in Figure 1, in accordance with an embodiment of the present invention, a method for intercepting or cleaning-up a plugin is provided, the method comprising:

Step 11: obtaining performance attributes of a plugin;

Step 12: determining whether the plugin meets a performance criterion based on the performance attributes and empirical data. If the plugin does not meet the performance criterion, proceed to Step 13; if the plugin meets the performance criterion, proceed to Step 14.

Step 13: intercepting or cleaning-up the plugin;

Step 14: no action taken.

In accordance with the method for intercepting or cleaning-up a plugin, the performance attributes of a plugin can be evaluated to determine whether the plugin meets a preset performance criterion, and the plugin can then be processed according to the result of the determination. This embodiment of the present invention is primarily directed to evaluating and processing plugins that are being installed, loaded, or running: if the plugin does meet the performance criterion, it will be intercepted if being installed, and cleaned-up if being loaded or running; if the plugin meets the performance criterion, no action is taken.

As shown in Figure 2, in accordance with another embodiment of the present invention, the method further comprises, prior to the above method,

Step 10: monitoring activity of the plugin in a user terminal, wherein the activity is selected from a group comprising installing, loading, and running of the plugin; identifying the plugin based on the monitored activity; if the plugin is identified, proceeds to step 11; if the plug is not identified, proceeds to step 15.

Step 15: evaluating performance attributes of the plugin when the plugin is not identified;

Step 16: determining whether the plugin meets a performance criterion based on the performance attributes and empirical data. If the plugin does not meet the performance criterion, proceed to Step 13; if the plugin meets the performance criterion, proceed to Step 14.

As shown in Figure 3, in accordance with another embodiment of the present invention, the method further comprises, prior to Step 10:

Step 8: obtaining empirical data of the plugin, wherein the empirical data comprises plugin importance data, plugin performance data, and/or end user review data; and

Step 9: determining whether the plugin meets a performance criterion based on the empirical data.

As shown in Figure 4, the empirical data can be obtained through plugin testing, backend collection, and/or web crawling, backend collection comprises frontend collection and end user review, frontend collection comprises hook mode and notification mode, the empirical data is stored in a remote cloud database or a local database. The cloud database is located at the server and can be used to store empirical data, and can communicate with the client over the network. The local database is located locally at the client and can be used to store empirical data.

Further, plug testing comprises testing the plugin in an automatic module to obtain information on the plugin's CPU usage, load time and/or memory usage.

Frontend collection can be implemented through a client, and includes hook mode and notification mode.

1. Hook mode: using a hook to obtain a start time and an end time for loading the plugin, a start time and an end time for running the plugin, and information on the plug's CPU usage and/or memory usage while the plugin is loading and running;

2. Reception mode: a client directly notifying starting and ending of loading and running of the plugin.

Frontend collection includes collection hardware and software environment to further evaluate performance data.

End user review includes collecting reviews of end users on the performance of the plugin, which can be placed in the plugin cleanup module of security software, or plugin monitoring modules, or any other places.

Web crawling includes crawling search engine data and/or website review data to evaluate the performance of the plugin.

In constructing the cloud database for empirical data, one factor that should be considered is the necessity of the plugin. For example, plugins like the Flash are highly necessary, and should be loaded even if its performance is poorer than other plugins, and the performance attributes of such plugins should be directly set as meeting the criterion. The factors to be considered regarding whether or not a plugin meets the performance criterion include but are not limited to plugins necessity or importance data, end user plugin performance review data; plugin performance testing data, frontend collected performance data, and relevant performance data on the Internet. For example, a plugin with long loading time, poor end user review and not particularly important can be set as not meeting the performance criterion.

The method for intercepting or cleaning-up a plugin can be implemented in a client and a server, and the client can include a performance acquisition module 201, an evalution/determination module 202 and a plugin processing module 203. The method can also be implemented in a plugin signature database 206 and a local empirical database 205. The server can include backend module 207. The method can also be implemented in cloud empirical data database 204 as shown in Figure 5. The performance acquisition module 201, evaluation/determination module 202 and plugin processing module 203 generally can be one or several independent installation files based

on the needs, and whose functionalities can be implemented as either Dynamic Link Library (DLL) or Label Information Base (LIB).

The evaluating and processing of a plugin being installed at the user terminal will be described in reference to Figure 6. The process includes:

Step 101: monitoring the installation of a plugin at a user terminal;

Step 102: obtaining the performance attributes of the plugin;

Step 103: matching the performance attributes with empirical data stored on a cloud database; obtaining a preset determination result after a successful match, and determining whether the plugin meets the performance criterion based on the preset determination result. If the plugin does not meet the performance criterion, proceed to Step 104; if the plugin meets the performance criterion, proceed to Step 105.

Step 104: intercepting or cleaning-up the plug-in.

Step 105: no action taken.

The plugin monitoring module 201 monitors various activities at the user terminal, such as addition, deletion, or modification of entries in the registry, files, and services. The plugin monitoring module 201 generally runs in the driver layer. The plugin signature database 206 can record identifying characteristics of plugins, such as those written to specific locations of the registry, which can be used to identify the plugin. When the plugin is being installed at the user terminal, the plugin monitoring module 201 can catch such activity, and compares such activity with plugin signature database 206 to identify the plugin and obtain other information on the plugin. As a result, it can be confirmed that the monitored activity is the installation of the specific plugin.

The plugin monitoring module 201 then sends information regarding the plugin, such as the identification (ID) of the plugin, to the evaluation/determination module 202. The evaluation/ determination module 202 is mainly used to obtain performance attributes of the plugin. The evaluation/determination module 202 can visit cloud empirical database 204 or local empirical database 205 to obtain performance attributes of the plugin. The evaluation/determination module 202 visits cloud empirical database 204 first to obtain performance attributes of the plugin from the server, then determines whether the plugin meets the performance criterion or directly obtains a preset result on whether the plugin meets the performance criterion. The result can be pre-obtained by backend module 207. The local empirical database 205 is a backup when there is no network connection, or when the network connection is poor and the cloud empirical database 204 cannot be connected. In accordance with this embodiment, the present result on whether the plugin meets the performance criterion can be located on the server, and be updated when data on

the server data is being updated. The present result can also be located at the client, and be updated through replacement of the relevant files.

Whether the plugin meets the performance criterion is determined not only based on the loading time of the plugin, or any other individual data, but can also be based on a variety of factors, such as combining the loading time with end user review.

Plugin can be installed as regular software, such as through a plugin installation package, or can be installed while other software is being installed. When a plugin is being installed, if it is determined that the plugin does not meet the performance criterion, the plugin can be intercepted by the plugin processing module 203 as shown in Figure 7. The interception user interface can be a popup window shown in Figure 7. If the plugin meets the performance criterion, no action will be taken.

The evaluation and processing of plugin being loaded at the user terminal will be described in reference to Figures 8 and 9. The process includes:

Step 401: monitoring the loading of a plugin at a user terminal;

Step 402: obtaining the performance attributes of the plugin;

Step 403: matching the performance attributes with empirical data stored on a cloud database; obtaining a preset determination result after a successful match, and determining whether the plugin meets the performance criterion based on the preset determination result. If the plugin does not meet the performance criterion, proceed to Step 404; if the plugin meets the performance criterion, proceed to Step 405.

Step 404: cleaning-up the plug-in.

Step 405: no action taken.

For plugin that has already been installed, the plugin monitoring module 501 can catch the plugin when it is being loaded at the user terminal, and sends to evaluation/determination module 502. The evaluation/determination module 502 can visit cloud empirical database 504 or local empirical database 505 to obtain performance attributes of the plugin. The data in the cloud empirical database 504 is acquired through the backend module 507. The performance attributes will be matched with empirical data. If a match is successfully found, a preset determination result is obtained, and whether the plugin meets the performance criterion is determined based on the preset determination result. If the plugin does not meet the performance criterion, prohibit the loading of the plugin if permitted by the user terminal, and process the plugin using plugin processing module 503 if permitted by the user terminal. If the plugin meets the performance

criterion, no action will be taken. During the removal of the plugin, identifying characteristics of the plugin should be obtained from the plugin signature database 505.

The evaluation and processing of plugin running at the user terminal will be described in reference to Figures 10 and 11. The process includes:

Step 601: monitoring the running of a plugin at a user terminal, obtaining performance attributes of the plugin;

Step 602: evaluate the performance attributes of the plugin;

Step 603: determining whether the plugin meets the performance criterion based on the performance attributes. If the plugin does not meet the performance criterion, proceed to Step 604; if the plugin meets the performance criterion, proceed to Step 605.

Step 604: cleaning-up the plug-in.

Step 605: no action taken.

A running plugin may have passed the testing during its loading, or its performance attributes might not been collected by the cloud empirical database 704 o local empirical database 705 through the backend module 707. Thus, the plugin monitor module 701 can be used to catch the plugin, and send it to the evaluation/determination module 702 to obtain the performance attribute of the plugin. It can also test the performance of the plugin in real time if needed, and derive a determination result. If the plugin does not meet the performance criterion, clean-up the plugin using plugin processing module 703 if permitted by the user terminal. If the plugin meets the performance criterion, no action will be taken. During the cleanup of the plugin, identifying characteristics of the plugin should be obtained from the plugin signature database 706. If the performance attributes of the plugin has already be collected, the plugin can be processed in similar fashion discussed above when it is being installed or loaded.

Figure 12 is an exemplary schematic diagram for an apparatus for intercepting or cleaning-up plugins in accordance with an embodiment of the present invention. As shown in Figure 12, the apparatus 20 includes a performance acquisition module 21, a first performance determination module 22, and a plugin processing module 23. The performance acquisition module 21 is used for obtaining performance attributes of a plugin; the first performance determination module 22 is sued for determining whether the plugin meets a performance criterion based on the performance attributes and empirical data; and the plugin processing module 23 is used for intercepting or cleaning-up the plugin if the plugin does not meet the performance criterion.

In accordance with the apparatus 20 for intercepting or cleaning-up a plugin, the performance attributes of a plugin can be evaluated to determine whether the plugin meets a preset performance criterion, and the plugin can then be processed according to the result of the determination. This embodiment of the present invention is primarily directed to evaluating and processing plugins that are being installed, loaded, or running: if the plugin does meet the performance criterion, it will be intercepted if being installed, and cleaned-up if being loaded or running; if the plugin meets the performance criterion, no action is taken.

Figure 13 is an exemplary schematic diagram for an apparatus for intercepting or cleaning-up plugins in accordance with another embodiment of the present invention. As shown in Figure 13, the apparatus 20 can further include a plugin monitoring module 24 for monitoring activity of the plugin in a user terminal, wherein the activity is selected from a group comprising installing, loading, and running of the plugin; identifying the plugin based on the monitored activity; and obtaining performance attributes of the plugin when the plugin is identified.

Figure 14 is an exemplary schematic diagram for an apparatus for intercepting or cleaning-up plugins in accordance with yet another embodiment of the present invention. As shown in Figure 14, the apparatus 20 can further include a performance evaluation module 25 for, after the plugin is not identified, evaluating performance attributes of the plugin; wherein the first performance determination module 22 is configured for determining whether the plugin meets a performance criterion based on the performance attributes and empirical data; and the plugin processing module 23 is configured for intercepting or cleaning-up the plugin if the plugin does not meet the performance criterion.

As shown in Figure 4, the empirical data can be obtained through plugin testing, backend collection, and/or web crawling, backend collection comprises frontend collection and end user review, frontend collection comprises hook mode and notification mode, the empirical data is stored in a remote cloud database or a local database.

Figure 15 is another exemplary schematic diagram for an apparatus for intercepting or cleaning-up plugins in accordance with an embodiment of the present invention. As shown in Figure 15, the apparatus 20 can further include an empirical data acquisition module 26 and a second performance determination module 28; the empirical data acquisition module 26 is used for obtaining empirical data of the plugin, wherein the empirical data comprises plugin importance data, plugin performance data, and/or end user review data. The second performance determination module 28 is used for determining whether the plugin meets a performance criterion based on the empirical data.

As shown in Figure 5, the apparatus 20 for intercepting or cleaning-up a plugin can be implemented in a client and a server, and the client can include a performance acquisition module 201, a evaluation/determination module 202 and a plugin processing module 203. The apparatus can also be implemented in a plugin signature database 206 and a local empirical database 205. The server can include backend module 207. The method can also be implemented in cloud empirical data database 204. The performance acquisition module 201, evaluation/determination module 202 and plugin processing module 203 generally can be one or several independent installation files based on the needs, and whose functionalities can be implemented as either Dynamic Link Library (DLL) or Label Information Base (LIB).

In this embodiment, the performance acquisition module 24 is equivalent to the performance acquisition module 201 in Figure 5, the performance acquisition module 21, the first performance determination module 22 and performance evaluation module 25 are equivalent to evaluation/determination module 202 in Figure 5; the plugin processing module 23 is equivalent to plugin processing module 203; the empirical data acquisition module 26 and the second performance determination module 28 are equivalent to backend module 207 in Figure 5.

The various embodiments of the present invention are merely preferred embodiments, and are not intended to limit the scope of the present invention, which includes any modification, equivalent, or improvement that does not depart from the spirit and principles of the present invention.

Claims

1. A method for intercepting or cleaning-up a plugin, the method comprising:

obtaining performance attributes of a plugin;

determining whether the plugin meets a performance criterion based on the performance attributes and empirical data; and

intercepting or cleaning-up the plugin if the plugin does not meet the performance criterion.

2. The method of claim 1, further comprising

monitoring activity of the plugin in a user terminal, wherein the activity is selected from a group comprising installing, loading, and running of the plugin;

identifying the plugin based on the monitored activity; and

obtaining performance attributes of the plugin when the plugin is identified.

3. The method of claim 2, further comprising, after identifying the plugin based on the monitored activity,

evaluating performance attributes of the plugin when the plugin is not identified;

determining whether the plugin meets a performance criterion based on the performance attributes and empirical data; and

intercepting or cleaning-up the plugin if the plugin does not meet the performance criterion.

4. The method of claim 1, further comprising, prior to obtaining performance attributes of the plugin,

obtaining empirical data of the plugin, wherein the empirical data comprises plugin importance data, plugin performance data, and/or end user review data; and

determining whether the plugin meets the performance criterion based on the empirical data.

5. The method of claim 1, wherein the empirical data is obtained through plugin testing, backend collection, and/or web crawling, backend collection comprises frontend collection and end user review, frontend collection comprises hook mode and notification mode,

plug testing comprises testing the plugin in an automatic module to obtain information on the plugin's CPU usage, load time and/or memory usage;

hook mode comprises using a hook to obtain a start time and an end time for loading the plugin, a start time and an end time for running the plugin, and information on the plug's CPU usage and/or memory usage while the plugin is loading and running;

reception mode comprises a client directly notifying starting and ending of loading and running of the plugin;

end user review comprises collecting reviews of end users on the performance of the plugin;

web crawling comprises crawling search engine data and/or website review data to evaluate the performance of the plugin; and

the empirical data is stored in a remote cloud database or a local database.

6. The method of claim 1, wherein determining if the plugin meets a performance criterion based on the performance attributes and empirical data comprises:

matching the performance attributes with empirical data stored on a cloud database;

obtaining a preset determination result after a successful match, and

determining whether the plugin meets the performance criterion based on the preset determination result.

7. An apparatus for intercepting or cleaning-up a plugin, the apparatus comprising:

a performance acquisition module for obtaining performance attributes of a plugin;

a first performance determination module for determining whether the plugin meets a performance criterion based on the performance attributes and empirical data; and

a plugin processing module for intercepting or cleaning-up the plugin if the plugin does not meet the performance criterion.

8. The apparatus of claim 7, further comprising

a plugin monitoring module for monitoring activity of the plugin in a user terminal, wherein the activity is selected from a group comprising installing, loading, and running of the plugin;

identifying the plugin based on the monitored activity; and obtaining performance attributes of the plugin when the plugin is identified.

9. The apparatus of claim 8, further comprising a performance evaluation module for, after identifying the plugin based on the monitored activity, evaluating performance attributes of the plugin when the plugin is not identified;

wherein the first performance determination module is configured for determining whether the plugin meets the performance criterion based on the performance attributes and empirical data; and

the plugin processing module is configured for intercepting or cleaning-up the plugin if the plugin does not meet the performance criterion.

10. The apparatus of claim 7, further comprising an empirical data acquisition module for obtaining empirical data of the plugin, wherein the empirical data comprises plugin importance data, plugin performance data, and/or end user review data; and

a second performance determination module for determining whether the plugin meets the performance criterion based on the empirical data.

11. The apparatus of claim 7, wherein the empirical data is obtained through plugin testing, backend collection, and/or web crawling, wherein backend collection comprises frontend collection and end user review, frontend collection comprises hook mode and notification mode,

plug testing comprises testing the plugin in an automatic module to obtain information on the plugin's CPU usage, load time and/or memory usage;

hook mode comprises using a hook to obtain a start time and an end time for loading the plugin, a start time and an end time for running the plugin, and information on the plug's CPU usage and/or memory usage while the plugin is loading and running;

reception mode comprises a client directly notifying starting and ending of loading and running of the plugin;

end user review comprises collecting reviews of end users on the performance of the plugin;

web crawling comprises crawling search engine data and/or website review data to evaluate the performance of the plugin; and

the empirical data is stored in a remote cloud database or a local database.

12. The apparatus of claim 7, wherein the first performance determination module is further configured for:

matching the performance attributes with empirical data stored on a cloud database;

obtaining a preset determination result after a successful match, and

determining whether the plugin meets the performance criterion based on the preset determination result.

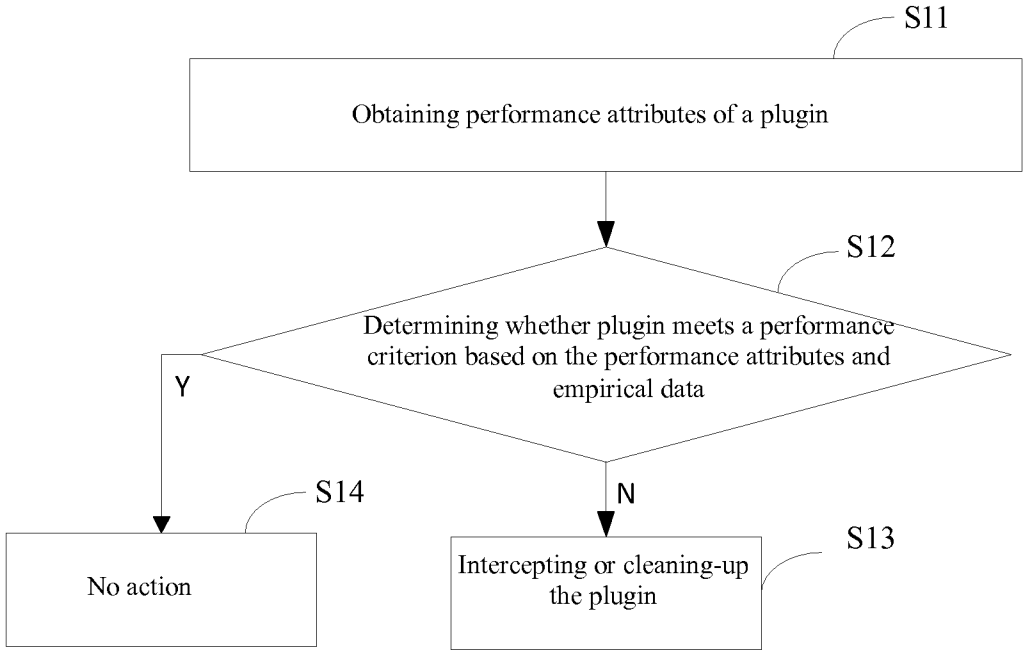
S11
Obtaining performance attributes of a plugin
S12
Determining whether plugin meets a performance criterion based on the performance attributes and empirical data
Y
N
S14
No action
S13
Intercepting or cleaning-up the plugin

Figure 1

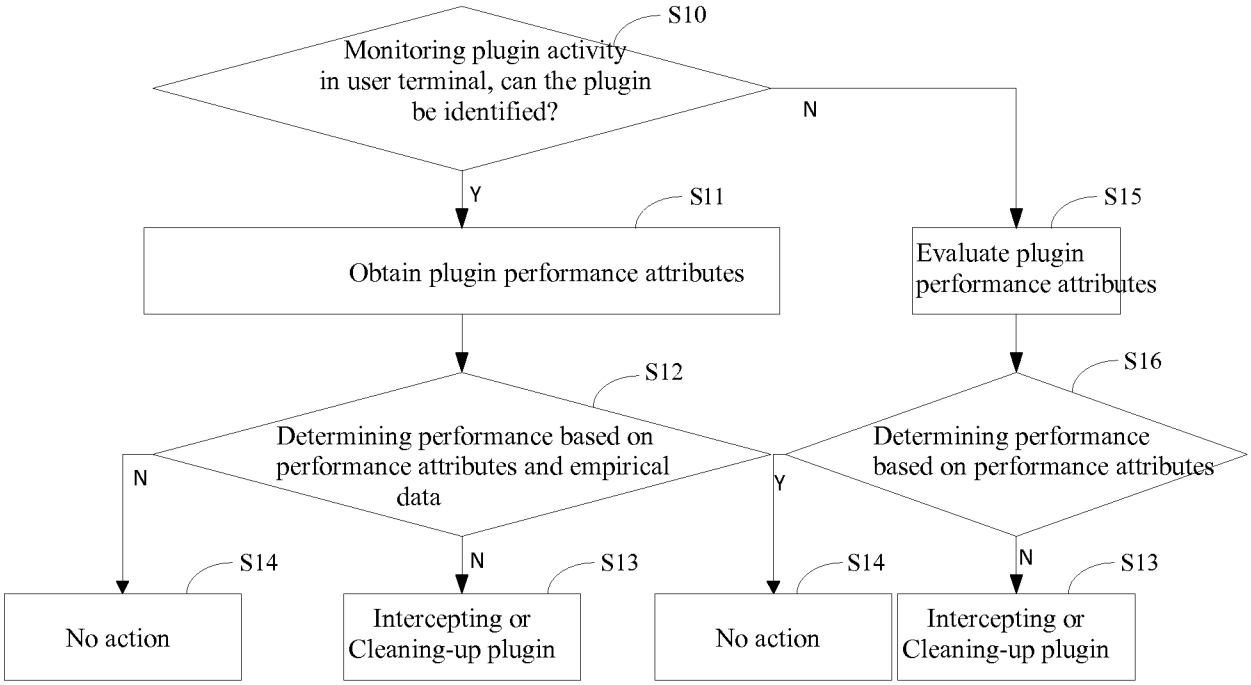
S10
Monitoring plugin activity in user terminal, can the plugin be identified?
N
Y
S11
Obtain plugin performance attributes
S15
Evaluate plugin performance attributes
S12
Determining performance based on performance attributes and empirical data
S16
Determining performance based on performance attributes
N
Y
N
N
S14
No action
S13
Intercepting or Cleaning-up plugin
S14
No action
S13
Intercepting or Cleaning-up plugin

Figure 2

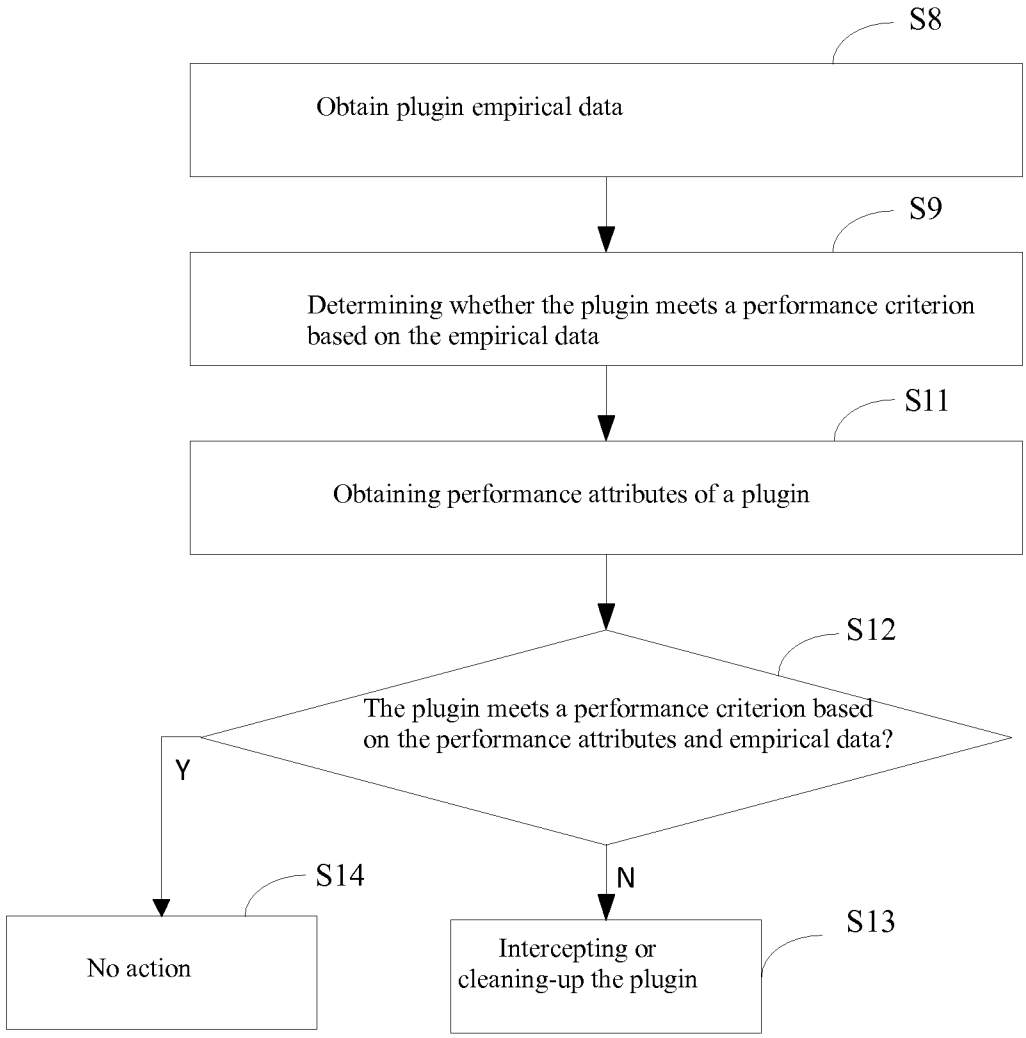
S8
Obtain plugin empirical data
S9
Determining whether the plugin meets a performance criterion based on the empirical data
S11
Obtaining performance attributes of a plugin
S12
The plugin meets a performance criterion based on the performance attributes and empirical data?
Y
N
S14
No action
S13
Intercepting or cleaning-up the plugin

Figure 3

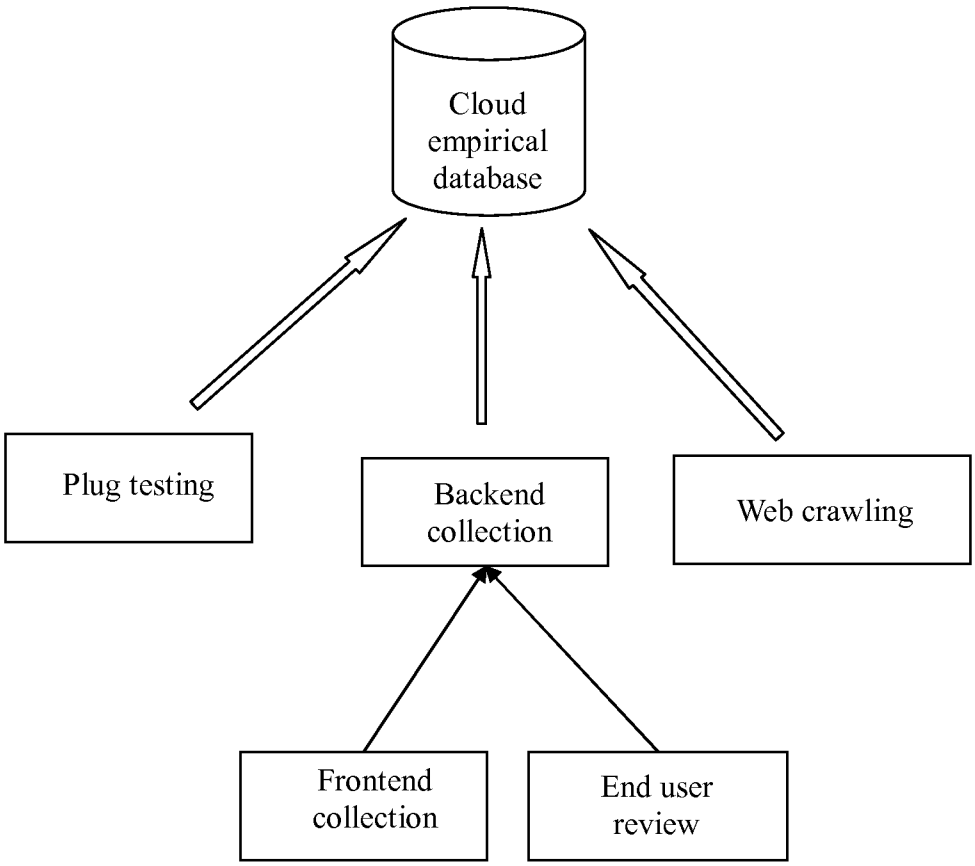
Cloud empirical database
Plug testing
Backend collection
Web crawling
Frontend collection
End user review

Figure 4

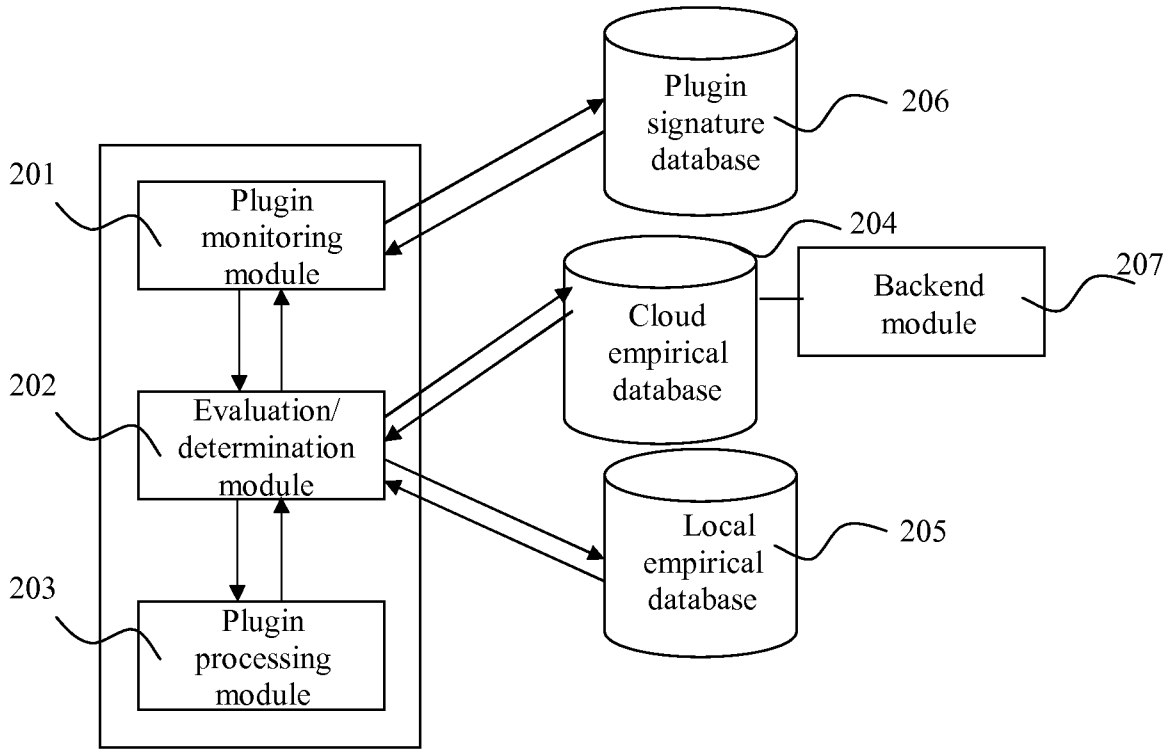
Plugin signature database
206
201
Plugin monitoring module
204
Cloud empirical database
Backend module
207
202
Evaluation/ determination module
Local empirical database
205
203
Plugin processing module

Figure 5

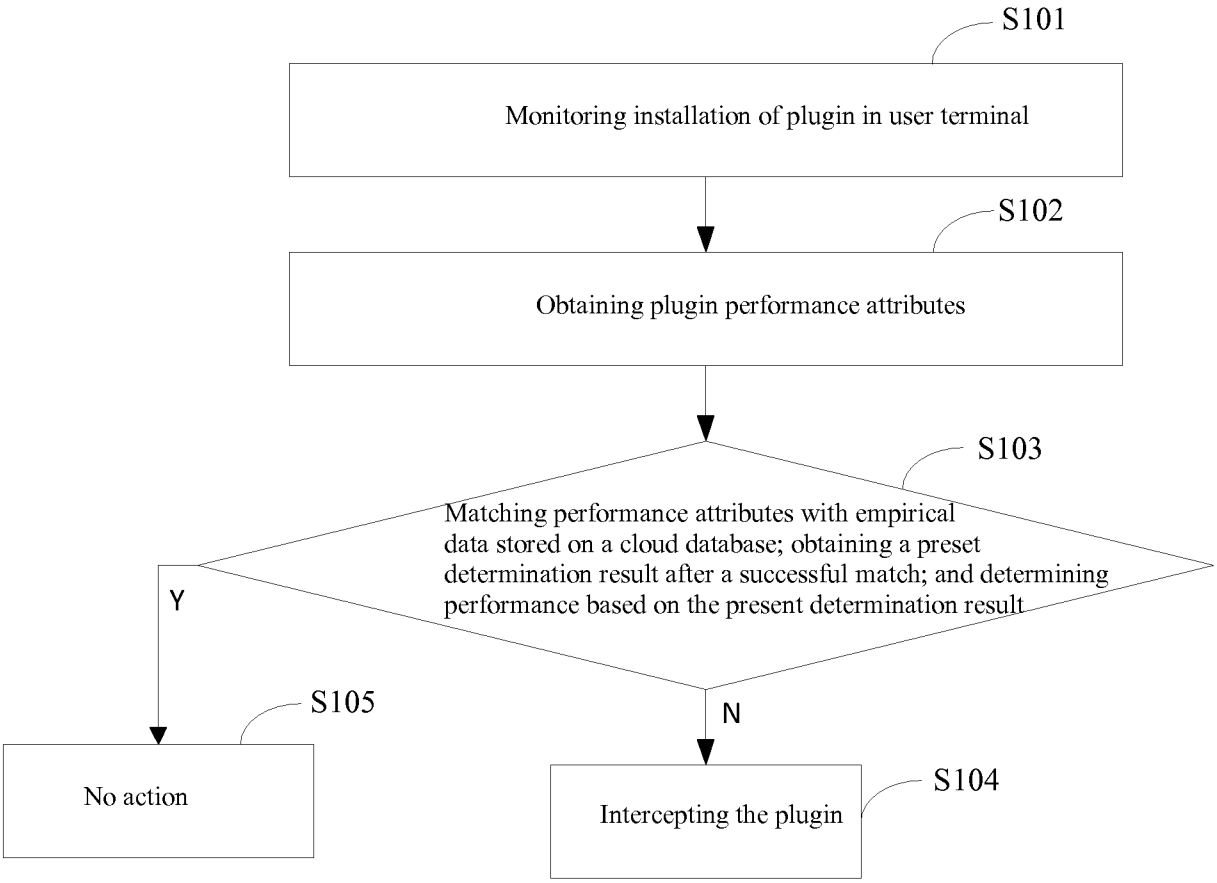

Figure 6

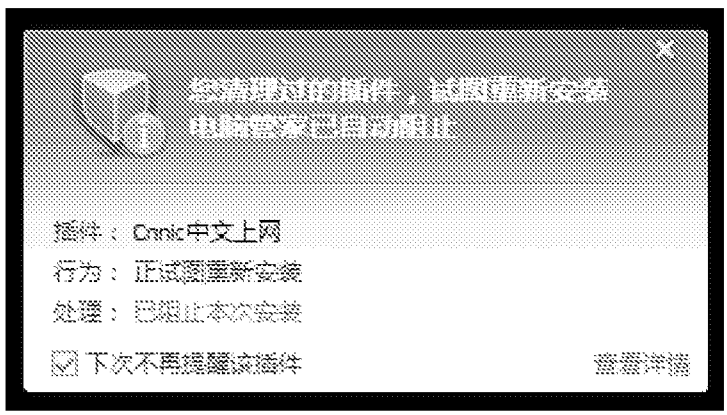

Figure 7

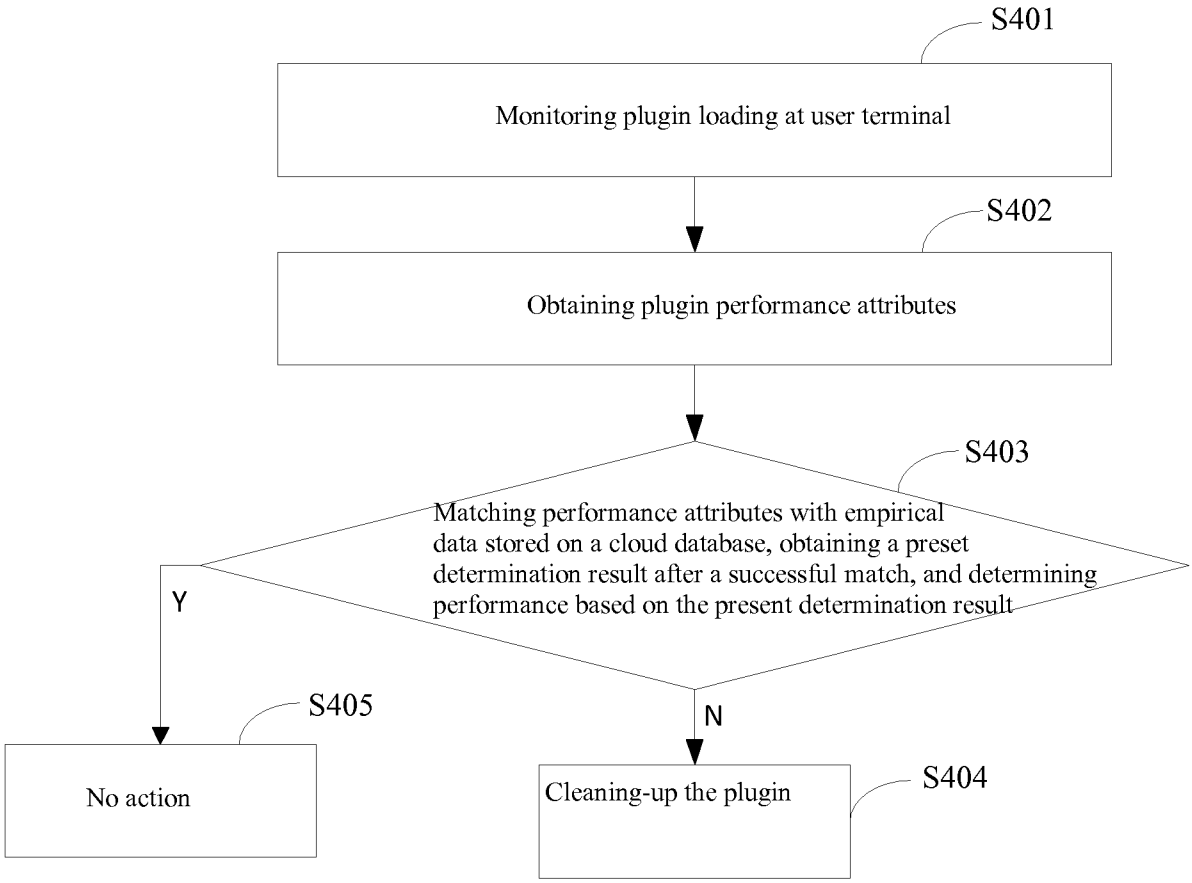
S401
Monitoring plugin loading at user terminal
S402
Obtaining plugin performance attributes
S403
Matching performance attributes with empirical data stored on a cloud database, obtaining a preset determination result after a successful match, and determining performance based on the present determination result
Y
N
S405
No action
Cleaning-up the plugin
S404

Figure 8

207

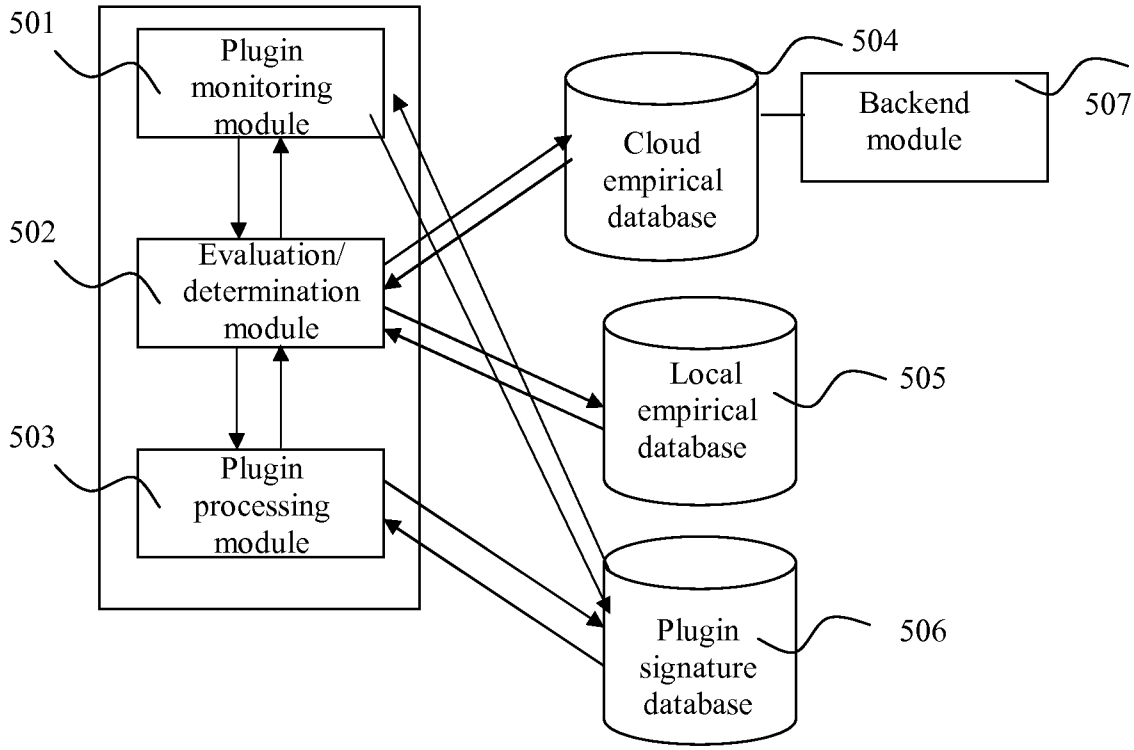
501
Plugin monitoring module
502
Evaluation/ determination module
503
Plugin processing module
504
Cloud empirical database
Backend module
507
Local empirical database
505
Plugin signature database
506

Figure 9

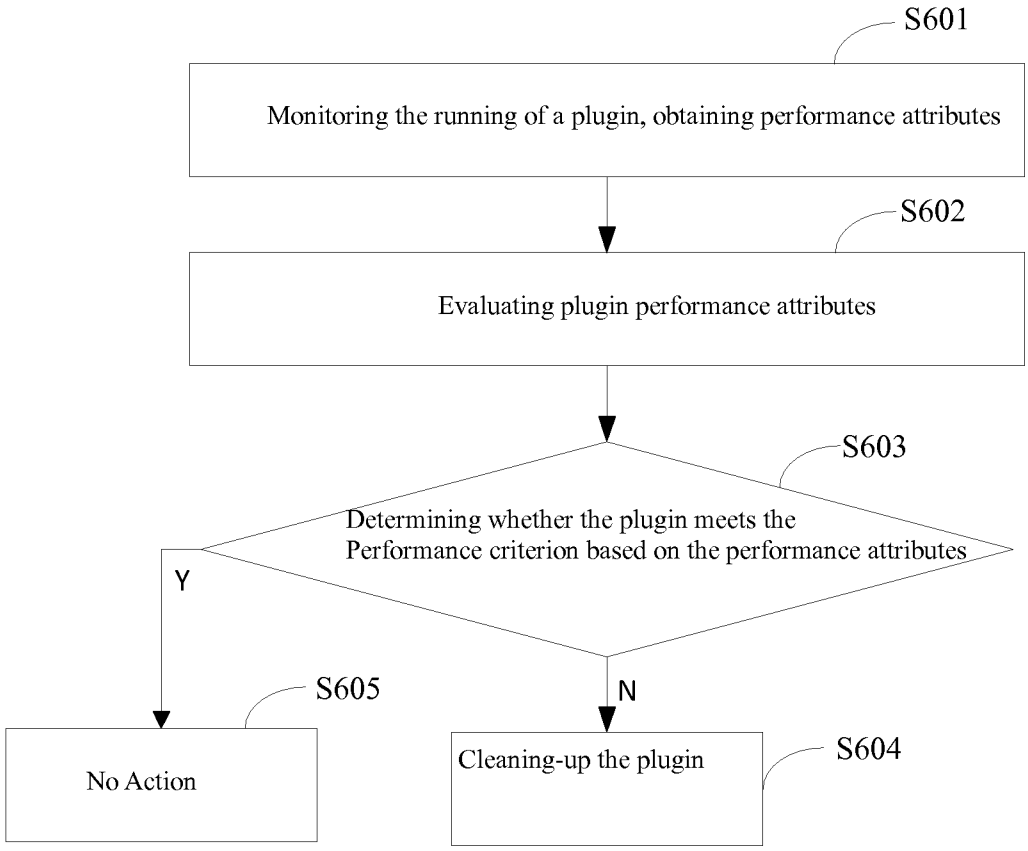
S601
Monitoring the running of a plugin, obtaining performance attributes
S602
Evaluating plugin performance attributes
S603
Determining whether the plugin meets the
Performance criterion based on the performance attributes
Y
N
S605
No Action
Cleaning-up the plugin
S604

Figure 10

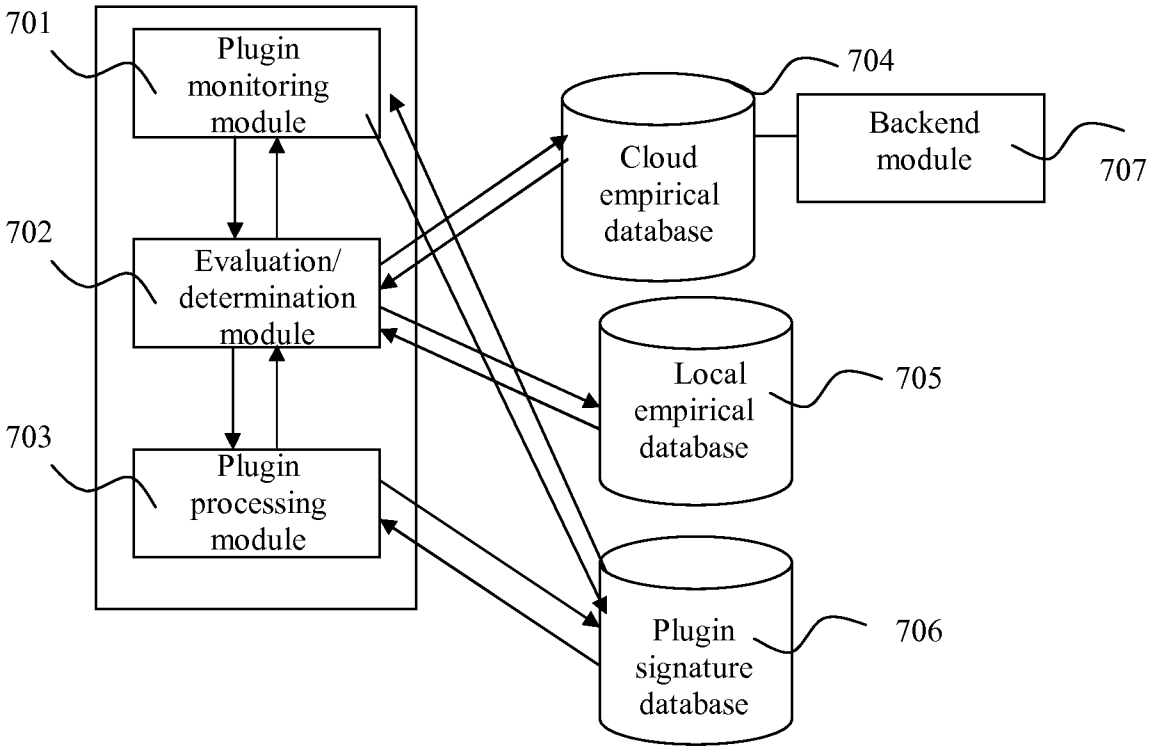
701
Plugin monitoring module
702
Evaluation/ determination module
703
Plugin processing module
704
Cloud empirical database
Backend module
707
Local empirical database
705
Plugin signature database
706

Figure 11

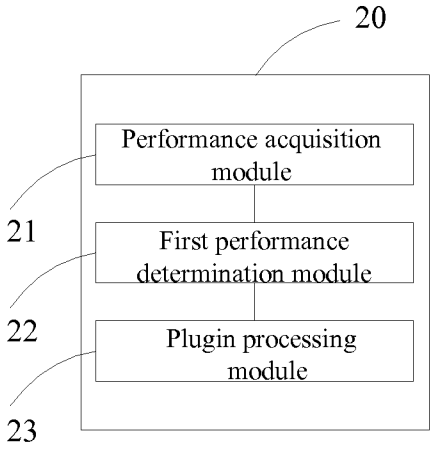

Figure 12

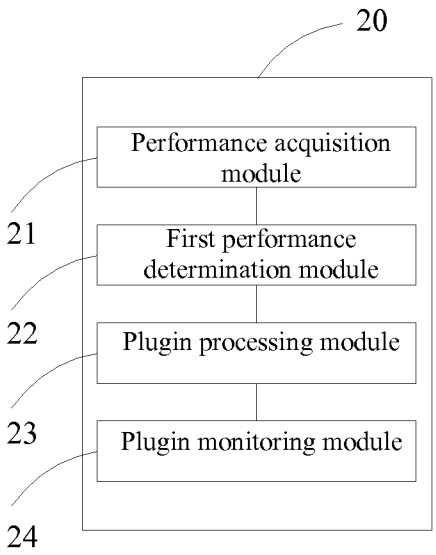

Figure 13

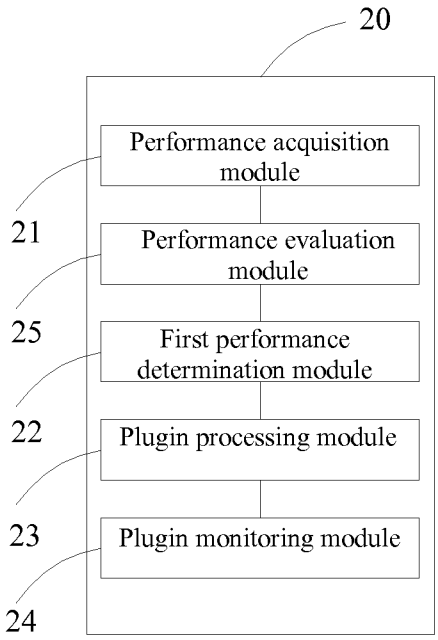

Figure 14

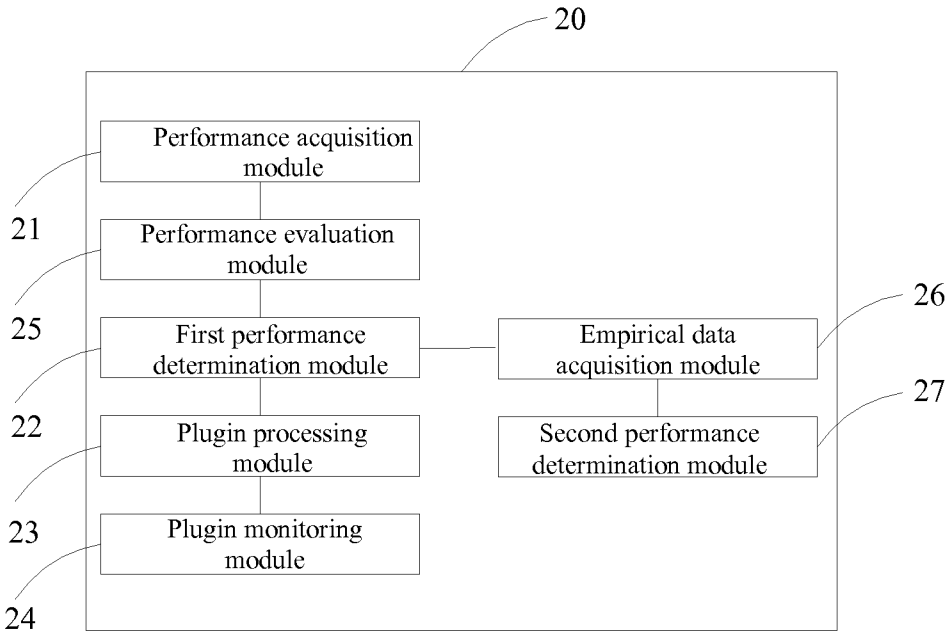

Figure 15

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2013/079482

A. CLASSIFICATION OF SUBJECT MATTER

G06F 11/00 (2006.01) i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC: G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

CNABS, CNTXT, WPI, EPODOC: plugin, plug w in, management+, clean+, intercept+, delet+, obliterat+, cpu, memory, resource, performance, capability+, time

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
P, X	CN 102831021 A (TENCENT TECHNOLOGY SHENZHEN CO LTD) 19 December 2012 (19.12.2012) see claims 1-12	1-12
X	US 20100251034 A1 (ALIBABA GROUP HOLDING CO LTD) 30 September 2012 (30.09.2012) see description, paragraphs 33-36, figure 2	1-4, 6-10, 12
A	CN 101291305 A (TENCENT TECHNOLOGY SHENZHEN CO LTD) 22 October 2008 (22.10.2008) see the whole document	1-12
A	CN 101727380 A (SAMSUNG ELECTRONICS CO LTD) 09 June 2010 (09.06.2010) see the whole document	1-12

☐ Further documents are listed in the continuation of Box C. ☒ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim (S) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
10 October 2013 (10.10.2013)	**24 Oct. 2013 (24.10.2013)**
Name and mailing address of the ISA/CN The State Intellectual Property Office, the P.R.China 6 Xitucheng Rd., Jimen Bridge, Haidian District, Beijing, China 100088 Facsimile No. 86-10-62019451	Authorized officer FENG, Huiping Telephone No. (86-10)62411838

Form PCT/ISA /210 (second sheet) (July 2009)

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2013/079482

Patent Documents referred in the Report	Publication Date	Patent Family	Publication Date
US 2010251034 A1	30.09.2010	EP 2414932 A1	08.02.2012
		US 8145950 B2	27.03.2012
		CN 101510167 A	19.08.2009
		WO 2010114611 A1	07.10.2010
		JP 2012522306 A	20.09.2012
CN 101291305 A	22.10.2008	None	
CN 101727380 A	09.06.2010	None	
CN 102831021 A	19.12.2012	None	