(51) International Patent Classification:
G06F 9/455 (2006.01) G06F 9/445 (2006.01)

(21) International Application Number:
PCT/EP2011/068695

(22) International Filing Date:
26 October 2011 (26.10.2011)

(25) Filing Language:
English

(26) Publication Language:
English

(30) Priority Data:
10191899.3 19 November 2010 (19.11.2010) EP


(72) Inventors: and

(74) Agent: KUISMA, Sirpa; IBM Deutschland Management & Business Support GmbH, IBM-Allee 1, 71139 Ehningen (DE).


Published: with international search report (Art. 21(3))

(54) Title: METHOD AND SYSTEM FOR GENERATING A VIRTUAL MACHINE BASED ON比赛模板

(57) Abstract: The invention relates to a method and a system for providing a virtual machine according to a specification of a user for use on a hypervisor, which allows an automatic selection of a suitable virtual machine template for providing a virtual machine in a short time. A virtual machine configurator provides a user interface to specify the software elements of various software hierarchy levels to be included in the virtual machine. The virtual machine configurator applies a scoring algorithm based on scores assigned to software elements according to their software hierarchy levels to calculate a score value for all virtual machine templates provided in a library, selects from the library the virtual machine template having the highest score value, and delivers the selected virtual machine template as a virtual machine to the hypervisor.
DESCRIPTION

METHOD AND SYSTEM FOR GENERATION A VIRTUAL MACHINE BASED ON TEMPLATES

FIELD OF INVENTION

The present invention relates to a method for providing a virtual machine according to a specification of a user for use on a hypervisor. The present invention also relates to a system for providing a virtual machine which is adopted to perform the method according to the present invention. Furthermore, the present invention relates to a computer readable medium containing a set of instruction that causes a computer to perform the above method and a computer program product adopted to execute the above method.

BACKGROUND

Virtualization is becoming more and more important to allow the use of different software installations on different hardware platforms. For example high performance server computers can be provided, which can be easily accessed from different client computers. The user does not even have to be aware of the underlying hardware of the server for running the desired software. This is enabled by a so-called hypervisor or virtual machine manager, which is installed on the hardware platform and provides an environment for running virtual machines, also called guests, which allow the execution of an operation system with desired applications. The hypervisor itself can be running directly on the hardware platform, which means without a separate operating system, or as an application within a standard operating system like Linux, Windows, or others. Also
hypervisors running on an intermediate abstraction layer are known in the art.

Provisioning a virtual machine with a configuration specified by the user is a key functionality for virtualization technologies. To provide a suitable virtual machine, the user has not only the option to specify an operating system to be used as a basis for further software elements, but also some kind of combination of middleware as well as specific applications and utilities to be available to the operating system in the virtual machine. To allow an efficient working, the virtual machines have to be provided in a timely fashion upon request by the user and with limited resource utilization.

To fulfil the requests of the users, the virtual machines can be created from the scratch each time. Accordingly, the user will be served a virtual machine exactly as specified, but he has to wait a maximum time each time when requesting a new virtual machine. Even if identical virtual machines are requested, the virtual machine has to be generated from the scratch each time, which consumes lots of resources.

Another typical approach to provide a virtual machine is to provide a certain set of virtual machine templates which are used as basis to start from when setting up the virtual machine as requested. Depending on the chosen virtual machine template, the degree of customization required to fulfil the user's request can be rather big and result in a provisioning time, which is not productive for the user. The user typically picks the virtual machine template most suitable from a list of available virtual machine templates. This reduces the possibilities to configure a virtual machine according to the needs of the user.
SUMMARY

It is therefore an object of the invention to provide a method and a system for providing a virtual machine according to a specification of a user for use on a hypervisor, which allow an automatic selection of a suitable virtual machine template for providing a virtual machine in a short time.

This object is achieved by the independent claims. Advantageous embodiments are detailed in the dependent claims.

Accordingly, this object is achieved by a method for providing a virtual machine according to a specification of a user for use on a hypervisor, whereby the specification is based on software elements of different software hierarchy levels and virtual machine templates are provided in a library, whereby a virtual machine configurator providing a user interface for specifying software elements according to the software hierarchy levels, the virtual machine configurator applying a scoring algorithm based on scores assigned to software elements according to their software hierarchy levels to calculate a score value for all virtual machine templates from the library, the virtual machine configurator selecting from the library the virtual machine template having the highest score value, and the virtual machine configurator delivering the selected virtual machine template as a virtual machine to the hypervisor.

The basic idea of the present invention is therefore to apply different scores to software elements, which can be selected by the user for the use in a virtual machine, according to the different software hierarchy levels so that the scoring algorithm can automatically calculate the score value for all available virtual machine templates from the library. Based on
the calculated score values, the virtual machine template with the highest score value is automatically selected and provided to the hypervisor. The separation of software elements according to different software hierarchy levels enables to consider provisioning requirements, which are usually different for the distinct software hierarchy levels. A system used for applying this method comprises the virtual machine configurator and the library containing available virtual machine templates. The virtual machine configurator can be run on a dedicated hardware within an IT-network and connected to a hardware, on which the hypervisor is running, or can be located physically on the same hardware platform. The hypervisor itself can be a single hypervisor instance running on its hardware platform, or can be one out of multiple hypervisors running on respective multiple hardware platforms, which does not imply any changes for the method according to the present invention. The library can be a kind of database, which can also be allocated at any place within the IT-network, or can be allocated physically on the same hardware platform as the virtual machine configurator.

Also, the library can be provided on different locations within the IT-network. The user interface provided by the virtual machine configurator can be accessed from any point within the IT-network for specifying the desired configuration of the software elements.

According to a modified embodiment of the present invention applying a scoring algorithm based on scores assigned to software elements comprises applying different scores for an operating system level, a middleware level, an application level and a utility level, the operating system level being the highest and the utility level being the lowest software hierarchy level. The different hierarchy levels specified here consider that e.g. the entire installation of an operating system implies a bigger provisioning time for the virtual
machine then just adding a utility, which is usually a small piece of software that can easily be installed. The use of four hierarchy levels is usually sufficient to allow the user an appropriate choice of software elements and represents a typical software installation. Nevertheless, a different number of software hierarchy levels can also be implemented.

In a preferred embodiment of the present invention applying a scoring algorithm based on scores assigned to software elements comprises the scores being assigned to each software element of a software hierarchy level and the sum of the scores of all software elements of a certain software hierarchy level being below the score of each software element of the above software hierarchy level. Accordingly, the selection of an appropriate virtual machine template first depends on the upper hierarchy levels and only afterwards on lower hierarchy levels, since the scores of software elements of a certain hierarchy level can never reach the score of a software element from the above hierarchy level. Nevertheless, assigning the scores this way still allows high flexibility and also the assignment of different scores to software elements of the same hierarchy level.

According to a preferred embodiment of the present invention, applying a scoring algorithm based on scores assigned to software elements comprises a software element being considered as a match if a software element name and a software element version match. Since different versions of a software element can comprise specific features, important bugfixes, fulfill certain performance requirements or show other advantages/disadvantages, it can be important to exactly match the software element including the element version. Such a matching can also comprise the specification of a certain software element version as a threshold value, so that all
software element versions above or below this threshold can be considered as a match.

According to a modified embodiment of the present invention calculating the score value comprises a software element provided in a virtual machine template adding its score to the scoring value, if the software element is requested by the user, and subtracting its score from the scoring value, if the software element is not requested by the user. This allows the consideration of software elements which are contained in a certain virtual machine template as well of those, which are not contained in this virtual machine template. Accordingly, differences regarding the provisioning of software elements of the same hierarchy level can be considered for the selection of the virtual machine template. For example, software elements that are more complicated to install or require a longer time for installation can have a higher score.

In a preferred embodiment of the present invention the virtual machine configurator connecting to a software element database containing installable files of software elements and automatically installing software elements, which are specified by the user but not provided in the virtual machine template, from the software element database to the virtual machine of the virtual machine template having the highest score before delivery. This enables the automatic installation of software elements as specified by the user, which are not already contained in the virtual machine template. Accordingly, with the best matching virtual machine template as basis, a high variety of configurations of virtual machines can be provided by adding different software elements with high efficiency. The respective system for performing this method comprises a software element database, which contains installables for software elements. The software element database can contain all offered software
elements, or just most frequently used software elements, so that less frequently used software elements have to be requested from a different database when required. The software element database can be provided on any kind of a server in the IT-network or even outside in a third party database.

According to a preferred embodiment of the present invention the virtual machine configurator storing requests for software elements of a virtual machine in a request database and adding a virtual machine template to the library according to the stored requests for software elements of a virtual machine. This allows having the library for a virtual machine templates always prepared with a set of most suitable virtual machine templates that match a high number of requests. The kind of information stored in the request database can be different and comprise for example names and versions of requested software element as well as specific sets of software elements to be considered together. Also the frequency or total number of uses of an existing virtual machine template can be stored in the request database.

Furthermore, the evaluation of requests for software elements depending on a software hierarchy level can be individually stored. When a virtual machine template is added, in the case of limited resources of the library, an existing virtual machine template can at the same time be removed or deleted from the library. A selection process for the virtual library template to be removed can be applied similar to the described method for selecting a virtual machine template to be added. Virtual machine templates to be added to the library can be provided from different databases, actually used virtual machines according to user requests or manually created virtual machine templates, e.g. created by a system administrator. The respective system contains the request database, which can either be provided on the same hardware platform like the virtual machine configurator or at a different location within
the IT-network. The request database can also be provided in an external network.

In an even preferred embodiment of the present invention the virtual machine configurator automatically creating a virtual machine template to be added to the library according to the requests for software elements of a virtual machine. The automatic generation of the virtual machine template can be realized without human interference by means similar to those used for installing software elements to existing virtual machine templates.

According to a modified embodiment of the present invention adding a virtual machine template to the library according to the stored requests for software elements of a virtual machine comprises verifying if software elements, which are not contained in any virtual machine template provided in the library, have been requested at least a certain number of times. A threshold value can be defined to provide a limit, so that a virtual machine template will be added to the library when the threshold value is reached according to the request database. The threshold value can be dynamically modified, e.g. to take into account the number of already available virtual machine templates and a capacity of the library. Such a threshold value can be defined for entire sets of software elements forming a virtual machine or can be based on software elements of a hierarchy level. Also individual software elements of a software hierarchy level can be taken into account, if they are frequently requested. If the virtual machine template to be added will replace an existing virtual machine template, the virtual machine template to be removed can be chosen in a way similar to the one described for adding the virtual machine template. For example, the least frequently chosen virtual machine template can be removed from the library. A replacement
also can be applied when a certain configuration a software
elements of a virtual machine, that is already represented in
the library as a virtual machine template, is less frequently
requested than a different configuration, which has not yet been
added to the library.

In a modified embodiment of the present invention adding a
virtual machine template to the library according to the stored
requests for software elements of a virtual machine comprises
verifying if software elements, which are not contained in any
virtual machine template provided in the library, are more
frequently requested than at least one virtual machine template
provided in the library. Frequency of use can indicate in a
short time increasing importance of a software element.

The object of the present invention can also be solved by a
computer readable medium such as a storage device, a floppy
disk, CD, DVD, Blue Ray disk, or a random access memory (RAM),
containing a set of instruction that causes a computer to
perform a method according to one of the preceding method
claims. Furthermore, the object can be solved by a computer
program product comprising a computer usable medium including
computer usable program code, wherein the computer usable
program code is adopted to execute the method of any of the
preceding claims.

As will be appreciated by one skilled in the art, aspects of the
present invention may be embodied as a system, method or
computer program product. Accordingly, aspects of the present
invention may take the form of an entirely hardware embodiment,
an entirely software embodiment (including firmware, resident
software, micro-code, etc.) or an embodiment combining software
and hardware aspects that may all generally be referred to
herein as a "circuit," "module" or "system." Furthermore,
aspects of the present invention may take the form of a computer program product embodied in one or more computer-readable medium(s) having computer readable program code embodied thereon.

Any combination of one or more computer readable medium(s) may be utilized. The computer readable medium may be a computer readable signal medium or a computer readable storage medium. A computer readable storage medium may be, for example, but not limited to, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, or device, or any suitable combination of the foregoing. More specific examples (a non-exhaustive list) of the computer readable storage medium would include the following: an electrical connection having one or more wires, a portable computer diskette, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or Flash memory), an optical fiber, a portable compact disc read-only memory (CD-ROM), an optical storage device, a magnetic storage device, or any suitable combination of the foregoing. In the context of this document, a computer readable storage medium may be any tangible medium that can contain, or store a program for use by or in connection with an instruction execution system, apparatus, or device.

A computer readable signal medium may include a propagated data signal with computer readable program code embodied therein, for example, in baseband or as part of a carrier wave. Such a propagated signal may take any of a variety of forms, including, but not limited to, electro-magnetic, optical, or any suitable combination thereof. A computer readable signal medium may be any computer readable medium that is not a computer readable storage medium and that can communicate, propagate, or transport
a program for use by or in connection with an instruction
execution system, apparatus, or device.

5 Program code embodied on a computer readable medium may be
transmitted using any appropriate medium, including but not
limited to wireless, wireline, optical fiber cable, RF, etc., or
any suitable combination of the foregoing.

10 Computer program code for carrying out operations for aspects of
the present invention may be written in any combination of one
or more programming languages, including an object oriented
programming language such as Java, Smalltalk, C++ or the like
and conventional procedural programming languages, such as the
"C" programming language or similar programming languages. The
15 program code may execute entirely on the user's computer, partly
on the user's computer, as a stand-alone software package,
partly on the user's computer and partly on a remote computer or
entirely on the remote computer or server. In the latter
scenario, the remote computer may be connected to the user's
20 computer through any type of network, including a local area
network (LAN) or a wide area network (WAN), or the connection
may be made to an external computer (for example, through the
Internet using an Internet Service Provider).

25 Aspects of the present invention are described below with
reference to flowchart illustrations and/or block diagrams of
methods, apparatus (systems) and computer program products
according to embodiments of the invention. It will be understood
that each block of the flowchart illustrations and/or block
30 diagrams, and combinations of blocks in the flowchart
illustrations and/or block diagrams, can be implemented by
computer program instructions. These computer program
instructions may be provided to a processor of a general purpose
computer, special purpose computer, or other programmable data
processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

These computer program instructions may also be stored in a computer readable medium that can direct a computer, other programmable data processing apparatus, or other devices to function in a particular manner, such that the instructions stored in the computer readable medium produce an article of manufacture including instructions which implement the function/act specified in the flowchart and/or block diagram block or blocks.

The computer program instructions may also be loaded onto a computer, other programmable data processing apparatus, or other devices to cause a series of operational steps to be performed on the computer, other programmable apparatus or other devices to produce a computer implemented process such that the instructions which execute on the computer or other programmable apparatus provide processes for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

Preferred embodiments of the invention are illustrated in the accompanied figures. These embodiments are merely exemplary, i.e. they are not intended to limit the content and scope of the appended claims.
Fig. 1 shows a schematic diagram of the system for performing the method according to the present invention in connection to a hypervisor,

Fig. 2 shows a flowchart of the method according to the present invention,

Fig. 3 shows a flowchart of method steps for provisioning a virtual machine based on an existing virtual machine template, and

Fig. 4 shows a flowchart of method steps for provisioning a virtual machine from the scratch.

DETAILED DESCRIPTION

Referring now to Fig. 1, a system 1 for providing a virtual machine according to a specification of a user can be seen. The system 1 comprises a virtual machine configurator 2, which is a computer program product with a computer usable program code. The virtual machine configurator 2 is executed on a server hardware, which is not explicitly shown in Fig. 1. The virtual machine configurator 2 is located within a not explicitly shown IT-network and connected thereover with a virtual machine template library 3, which contains 1 to n virtual machine templates 4. The virtual machine templates 4 in this embodiment contain a software image and a configuration information, which enable the delivery and execution of the image.

The system 1 further comprises a software element database 5, which contains different software elements for building virtual machine templates 4 or for installation to a virtual machine template 4.
Furthermore, the system 1 comprises a user request database 6, which is also connected to the virtual machine configurator 2 via the IT-network. The user request database 6 is adopted to store user requests, which are entered via a user interface 7 provided by the virtual machine configurator 2. The user interface 7 is provided on a client system connected to the virtual machine configurator 2 via the IT-network. The system 1 of the present invention is according to Fig. 1 connected via the IT-network to a hypervisor 8, which is in this embodiment a single high performance computer hardware for hosting multiple virtual machines 9 thereon. Accordingly, the virtual machines 9 can be executed on the hypervisor 8 without considering the underlying hardware.

Referring now to Fig. 2, the general method for providing a virtual machine 9 according to a specification of a user of the present invention is shown. In step 100 a user, which is not shown in Fig. 1, connects via the user interface 7 to the virtual machine configurator 2. The virtual machine configurator 2 provides in the user interface 7 available software elements according to software hierarchy levels to the user. The software hierarchy levels in this embodiment are an operating system level, a middleware level, an application level and a utility level. The user selects a choice of at least one software element and this selection is transmitted as request via the IT-network to the virtual machine configurator 2.

In step 110, the request of the user is transmitted from the virtual machine configurator 2 to the user request database 6 and stored for further evaluation. Nevertheless, the user request can be stored in the user request database 6 basically at any time as long as the user request is being processed by the virtual machine configurator 2.
Based on the user request from step 100, in step 120 the virtual machine configurator 2 calculates a score value for all virtual machine templates 4 from the virtual machine template library 3. The calculation of the score is based on the software hierarchy levels and is explained by example with a set of three virtual machine templates 4. The different virtual machine templates 4 have the following configuration:

Virtual machine template #1:
- Windows 2003 Server SP2

Virtual machine template #2:
- Linux Red Hat ES 4.0
- IBM DB2 9.1
- WinZip 9.0

Virtual machine template #3:
- Linux Suse 9.0
- IBM Websphere Server 6.1
- IBM DB2 8.5
- IBM ITDS 6.0
- TPM 7.1

Each of the used software elements in any of the virtual machine templates has assigned a score according to the following list. The list also identifies the software hierarchy level of each software element:

<table>
<thead>
<tr>
<th>Software element name and version</th>
<th>Software hierarchy level</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows 2003 Server SP2</td>
<td>Operating System</td>
<td>100000</td>
</tr>
<tr>
<td>Linux Red Hat ES 4.0</td>
<td>Operating System</td>
<td>100000</td>
</tr>
</tbody>
</table>
Important with the scores assigned to the different software elements is, that the sum of the scores for all software elements from each software hierarchy level is below the lowest score of a software element from the respective upper software hierarchy level. In this example, the sum of the scores of the software elements from the middleware level is 43,000, whereas all operating systems defining the upper software hierarchy level have a score of 100,000.

Considering that the user requests a virtual machine comprising Windows 2003 Server SP2 with IBM DB2 9.1 and WinZip 9.0.

Accordingly, the score values for the three virtual machine templates 4 are calculated like the following, with the applied score of each software element given in brackets:

25 Virtual machine template #1: score value +100000
Windows 2003 Server SP2 (+100000)

Virtual machine template #2: score value -89950
Linux Red Hat ES 4.0 (-100000)
IBM DB2 9.1 (+10000)
WinZip 9.0 (+50)

Virtual machine template #3: score value -134000
Linux Suse 9.0 (-100000)
IBM Websphere Server 6.1 (-15000)
IBM DB2 8.5 (-10000)
IBM ITDS 6.0 (-8000)
TPM 7.1 (-1000)

It is to be noted that a match is only considered when the name as well as the version number of a software element match. In case of a match, the positive score is applied, i.e. the score is added to the score value of a virtual machine template 4. Otherwise, the negative score is applied, i.e. the score is subtracted from the respective score value. Overall, only the first virtual machine template 4 has a positive score value, whereas the score value of the other virtual machine templates 4 is negative.

In step 130, the score values of the three virtual machine templates 4 are compared and the virtual machine template 4 with the highest score is selected, in this case virtual machine template #1.

Subsequently, according to step 140 a verification is performed, if the virtual machine template 4 is suitable compared to the user request from step 100. Therefore, in this embodiment the score value is analyzed to verify, whether it is above a certain threshold value. Other means of decision if the virtual machine template 4 is suitable can also be applied. If the virtual machine template 4 is considered suitable, the method continues with step 150, otherwise with step 160.

In step 150, the virtual machine configurator 2 prepares the virtual machine template 4 for release as shown in Fig. 3.
Accordingly, in step 200, a usage counter for the virtual machine template 4 is increased and stored as a historical data within the user request database 6.

According to step 210, the image of the virtual machine 9 is copied from the virtual machine template library 3 to the hypervisor 8.

In step 220, an XML data file is created containing required instance settings before running the virtual machine 9 on the hypervisor 8. These settings include a network host name, an IP address, and so on.

In a step 230, the XML data file is copied to a virtual floppy disc, also called v-floppy, and attached to the virtual machine 9. When the virtual machine 9 is booted on the hypervisor 8, the virtual floppy disc becomes automatically visible to an activation engine of the hypervisor 8.

The virtual machine 9 is powered on in step 240. Accordingly, the execution of activation scripts using the information of the v-floppy parameter file is automatically triggered.

According to a step 250, the virtual machine configurator 2 creates a list of software elements, which have been specified by the user in step 100, but which are not contained in the selected virtual machine template 4.

The installation of the missing software elements is done in step 260. The missing software elements are retrieved from the software element database 5 and copied to the virtual machine 9.

According to step 270, the installation of the software elements is launched on the virtual machine 9. In this embodiment, the
required software elements are installed subsequently based on the software hierarchy levels, starting with the software elements from the highest software hierarchy level. The method then continues with step 170.

If the virtual machine template 4 selected in step 130 is not considered suitable in step 140, according to step 160 a new virtual machine 9 is created from the scratch. The details of step 160 can be seen in Fig. 4.

Accordingly, in step 300, the virtual machine configurator 2 creates a new virtual machine 9 and performs the configuration of its hardware resources.

In step 310, the virtual machine 9 is started on the hypervisor 8 and the image file of the operating system is retrieved from the software elements library 5, so that the installation of the operating system can be started.

In step 320, all required software elements according to the configuration requested by the user in step 100 are retrieved from the software element library 5 and installed on the virtual machine. The method then continues with step 170.

In step 170, the virtual machine configurator 2 releases the virtual machine 9 to the user to be executed on the hypervisor 8, independently whether the virtual machine 9 was based on an existing virtual machine template 4 from the virtual machine template library 3 according to step 150 or was created from the scratch according to step 160.

At any point of the method described above including before the described method or afterwards, historical data of the request can be analyzed to evaluate whether the virtual machine template
library 3 has to be updated by adding a virtual machine template 4 or by replacing an existing virtual machine template 4 by a different virtual machine template 4. Virtual machine templates 4 to be added can be built from the scratch according to steps 300 to 320, as shown in Fig. 4, or can be based on existing virtual machine templates 4, which are updated with additional software elements according to steps 200 to 270 according to Fig. 3.

According to the present embodiment, evaluate whether the virtual machine template library 3 has to be updated is based on the starting software hierarchy levels. The number of prior provisioning requests that match the configuration starting from the highest software hierarchy level is compared to a certain threshold value. To be more detailed, first the user requests are considered only in respect of the operating system level, next by considering the operating system level and the middleware level, furthermore by additionally considering the application level and finally by considering the operation system level, the middleware level, the application level and the utility level. In case the threshold value is exceeded, the virtual machine 9 containing the respective configuration is designated as candidate to become a template. The candidate will be added to the virtual machine template library 3 after being created as already described before.

When the capacity of the virtual machine template library 3 reaches its limit, the less frequently requested virtual machine template 4 from the virtual machine template library 3 is removed. In this way, the virtual machine template library 3 will always contain the most suitable virtual machine templates 4, so that the provisioning of a virtual machine 9 to a user can also be automatically performed in an efficient manner.
The flowchart and block diagrams in the Figures illustrate the architecture, functionality, and operation of possible implementations of systems, methods and computer program products according to various embodiments of the present invention. In this regard, each block in the flowchart or block diagrams may represent a module, segment, or portion of code, which comprises one or more executable instructions for implementing the specified logical function(s). It should also be noted that, in some alternative implementations, the functions noted in the block may occur out of the order noted in the figures. For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved. It will also be noted that each block of the block diagrams and/or flowchart illustration, and combinations of blocks in the block diagrams and/or flowchart illustration, can be implemented by special purpose hardware-based systems that perform the specified functions or acts, or combinations of special purpose hardware and computer instructions.
CLAIMS

1. Method for providing a virtual machine (9) according to a specification for use on a hypervisor (8), the specification being related to software elements of different software hierarchy levels, wherein the method comprises:
   - providing virtual machine templates in a library (3);
   - providing a user interface (7) for specifying software elements according to the software hierarchy levels;
   - assigning scores to said software elements according to their software hierarchy levels;
   - calculating a score value for all virtual machine templates (4) from the library (3), based on the scores assigned to said software elements;
   - selecting from the library (3) the virtual machine template (4) having the highest score value; and,
   - delivering the selected virtual machine template (4) as a virtual machine (9) to the hypervisor (8).

2. Method according to claim 1, wherein said step of assigning scores to said software elements comprises assigning different scores for an operating system level, a middleware level, an application level and a utility level, the operating system level being the highest and the utility level being the lowest software hierarchy level.

3. Method according to any of claims 1 or 2, wherein said step of assigning scores comprises assigning the scores to each software element of a software hierarchy level so that the sum of the scores of all software elements of a certain software hierarchy level are below the score of each software element of the above software hierarchy level.
4. Method according to any preceding claim, wherein said step of assigning scores comprises considering a software element as a match if a software element name and a software element version match.

5. Method according to any preceding claim, wherein said step of calculating a score value for all the virtual machine templates comprises for each software element provided in each virtual machine template (4) adding its score to the score value, if the software element is requested by the user, and subtracting its score from the score value, if the software element is not requested by the user.

6. Method according to any preceding claim, wherein it comprises connecting to a software element database (5) containing installable files of software elements and automatically installing software elements which are specified by the user but not provided in the virtual machine template (4), from said software element database (5) to said virtual machine (9).

7. Method according to any preceding claim, wherein it comprises storing requests for software elements of a virtual machine (9) in a request database (6), a virtual machine template (4) being added to said library (3) depending on said stored requests.

8. Method according to claim 7 wherein said step of adding a virtual machine template (4) to the library (3) according to the stored requests for software elements of a virtual machine (9) comprises verifying a condition related to the number of times said software elements have been requested, if the software elements are not contained in any virtual machine template (4).

9. Method according to claim 8, wherein said condition comprises verifying if said software elements are more frequently
requested than at least one virtual machine template (4) provided in the library (3).

10. A computer-readable medium such as a storage device, a floppy disk, CD, DVD, Blue Ray disk, or a random access memory (RAM), containing a set of instruction that causes a computer to perform a method according to one of the preceding method claims.

11. A computer program product comprising a computer usable medium including computer usable program code, wherein the computer usable program code is adopted to execute the method of any of the preceding method claims.

12. System (1) comprising means adapted for carrying out the method according to any one of claims 1 to 9.
### A. CLASSIFICATION OF SUBJECT MATTER

INV. G06F9/455  G06F9/445

ADD.

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)

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 practical, search terms used)

EPO-Internal, WPI Data

### C. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
</table>

**X** Further documents are listed in the continuation of Box C. **X** 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 document 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.

**A** document member of the same patent family

15 December 2011

Date of the actual completion of the international search

23/12/2011

Date of mailing of the international search report

Name and mailing address of the ISA/
European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040,
Fax: (+31-70) 340-3016

Glaser, Norbert

Authorized officer
<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>EP 2 037 362 A1 (NEC CORP [JP])</td>
<td>1-12</td>
</tr>
<tr>
<td></td>
<td>18 March 2009 (2009-03-18)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>paragraph [0036] - paragraph [0037]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>paragraph [0053] - paragraph [0054]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>paragraph [0066] - paragraph [0074]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>paragraph [0109] - paragraph [0112]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>paragraph [0132]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>paragraph [0027] - paragraph [0032]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>paragraph [0036]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>paragraph [0056] - paragraph [0059]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>paragraph [0060] - paragraph [0062]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>paragraph [0068] - paragraph [0069]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>paragraph [0072]</td>
<td></td>
</tr>
<tr>
<td>Patent document cited in search report</td>
<td>Publication date</td>
<td>Patent family member(s)</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>-----------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WO 2008080827 A1</td>
</tr>
<tr>
<td>US 2009300057 A1</td>
<td>03-12-2009</td>
<td>US 2009300057 A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 2009300076 A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 2009300151 A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 2009300604 A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 2009300641 A1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 2009210527 A1</td>
</tr>
<tr>
<td>US 2007157185 A1</td>
<td>05-07-2007</td>
<td>NONE</td>
</tr>
</tbody>
</table>

Form PCT/ISA/210 (patent family annex) (April 2005)