Whitepaper - Draft preliminary

UnifiedSessionsManager

A Service Management Approach for CloudComputing

November 24, 2010

Contents

List of Tables

List of Figures

.

1 Abstract

The UnifiedSessionsManager is from beginning on designed for the management of embedded services within virtual and physical machines, including the login sessions for both. This results in a slightly different concept than of the products which are actually designed around the machine in their main focus.

Thus the concept of the UnifiedSessionsManager avoids - whenever possible - the application of machine oriented attributes, but presents a common generic task oriented interface. Where the user could be either an ordinary user with restricted permissions, or an administrator with advanced permissions. Therefore the interface is designed as an task oriented User-Centric interface, encapsulating technical details whenever possible.

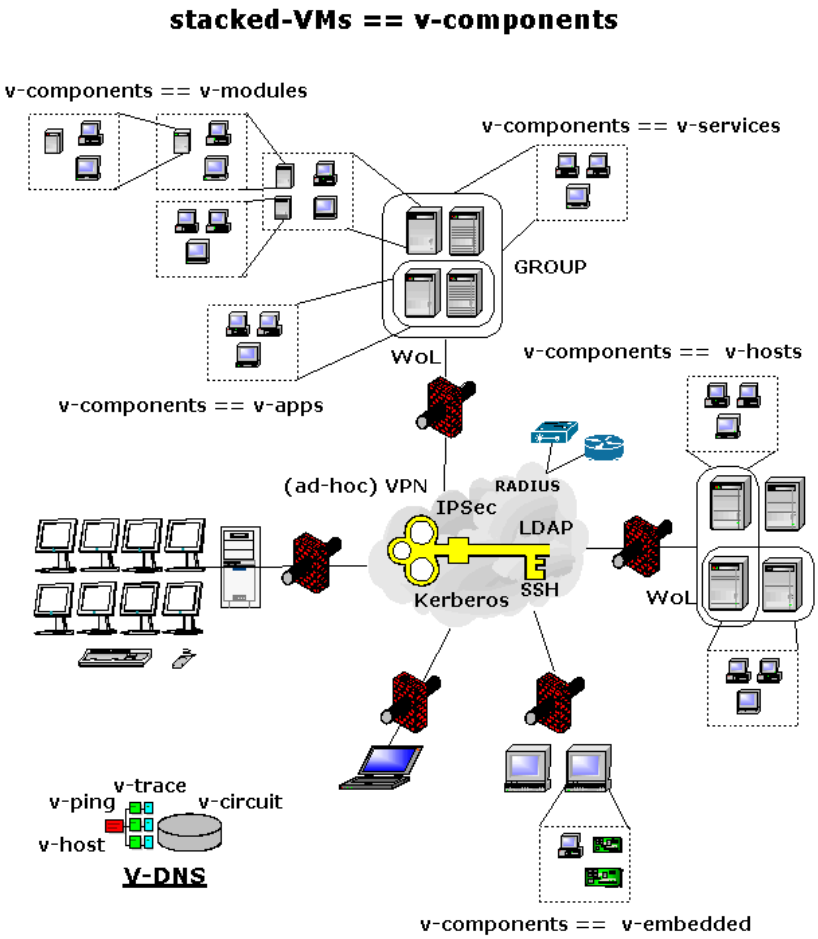

Figure 1: UnifiedSessionsManager as a Service-Composer

The overall resulting functionality is positioned as a personal Service-Manager, and offers specific features for the ease of the handling of combined services. The presented interface for this is kept unique by the definition of a superset of partly optional attributes. Thus the neatless view onto a pool of services with various available features - v-components - is provided.

The current version features particularly the desktop based assembly and automation of arbitrary services, following versions will emphasize the VM based assembly. Either by reference, or by containment of nested VMs - stacked-VMs - VSTACKs.

2 A Basic View to Services in the Cloud

The first step to define a target structure for the requirements of a 'personal Cloud-Management' application is the definition of an overall concept for the major day-by-day usage embedded into a heterogeneous IT landscape.

Related to the CloudServices the evolution of the basically quite similar progress of the internet/intranet could be used as a pattern. This was developed by using the term 'internet' for the 'geek-period', the term 'intranet' for the migration into the high-priced enterprise customers. Finally ther term 'internet' is used for the conversion into the public volume market. Each of the overlapping periods had it's specific product policies and design requirements.

The analogy which already could be seen and used for the next steps is the eveolution of the iconising terminology Cloud-Computing, Private-Clouds, and Public-Clouds with Cloud-Computing as a synonym. The implied content is quite similar as for the internet for a public access to variuos distributed resources. The association with **intranet** as a closed-group access pattern is almost the same as associated with **Private-Cloud**.

Therefore the expectation is the evolution to a quite similar destination with some adapted design requirements. The major difference for the targeted **Public-Cloud** specification is the personal on-demand application of the provided services without the requirement of an provisioning human administrator. This dialogue oriented public purchase of on-demand processing and application services is now applicable due to widely available technology.

Thus the design implication for the UnifiedSessionsManager was the early introduction of a **personal service management application - A Personal CloudComposer**.

The UnifiedSessionsManager is focusing on the personal management and application of embedded services within the machines. The contained services are represented by specialized applications, either as a single application, or a set of grouped applications.

The mapping of the design pattern within the UnifiedSessionsManager is represented by the functional architecture and the provided interfaces. The architecture is mapped here to a layered plugin structure, where these are categorised to task specific sets of entities. These are either representing hardware associated sessions features such as physical machines/**PMs** and virtual machines/**VMs**, or defining logical sessions objects, dynamically created and tighly coupled to the demanded services. The latter extends the definition of a service even to the actual sessions as the frontend, interconnection entity, and the lifetime of the optinally executed application. The on-demand sessions are handeled by the **HOSTs** plugins comprising logins and the execution of applications.

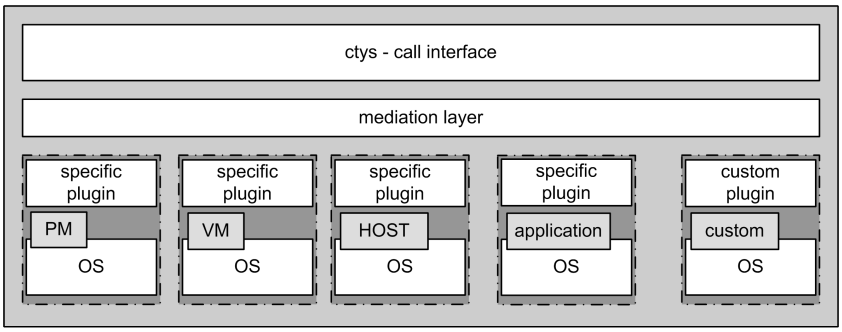

Figure 2: UnifiedSessionsManager as a Service-Composer

This is visibly expressed by a common syntax which is the superposition of all attributes. Thus the UnifiedSessionsManager is targeted as a **Personal Service-Management Application** - a **ServiceComposer**.

The first emphasis is the composition of assembled desktops by various network services, which are actually ordinary logins to machines with specific software sets. Therefore the script interface with GROUPS and MACROs is provided first. Thus any user could easily design his personal complex service compositions including the frontend. Optionally by dialogue, or for batch-repetition by the creation of a few simple bash scripts based on

remote-logins and remote-execution.

The evolutionary steps to the application level service composition leak for now some OS and probably HW support for nested multi-level stacked VMs. This becomes by the application of emulators quickly a performance challange.

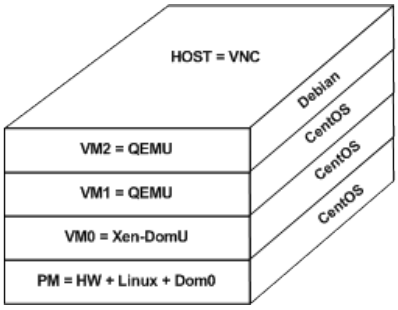

Figure 3: Stacked VMs - VSTACK

Thus this is for now still a draft for proof-of-concept from 07/2008. The actual implementation was based on a VSTACK of nested QEMU based emulators within Xen and VMware Hypervisors. The following test demonstrated the instanciation of multiple entities within each layer.

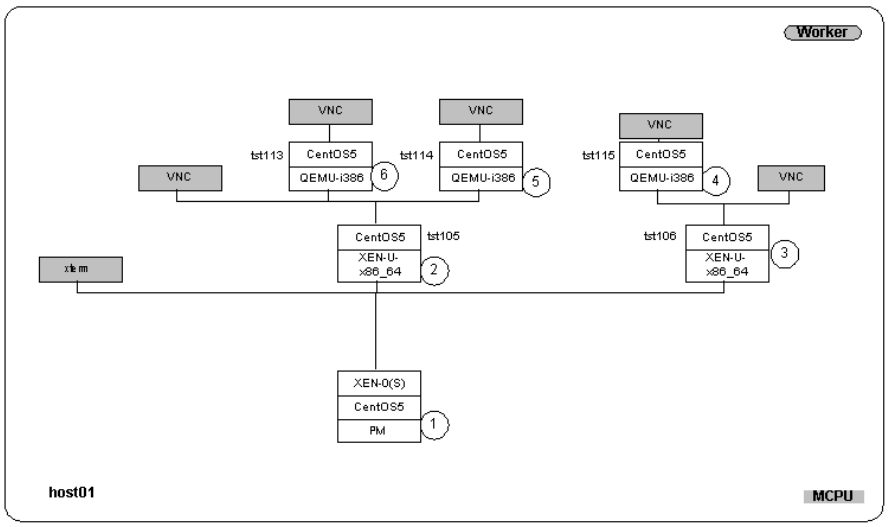

Figure 4: Stacked VMs - Multiple-Instance VSTACK

The scope of applicability comprises the neatless application by ordinary users as well, as by administrators with advanced permissions. The access control is almost solely delegated to the system facilities - here OpenSSH, Kerberos, LDAP, and sudo in combination with file access permissions either by UNIX only or by Posix-Attributes. This could be extended by SELinux and/or AFS when required and running on Linux.

The current application of VMSTACKs is mainly applied for the advanced management of VMs as a flat appliance structure based on their contained guest OS attributes.

The first full feature set and target direction of the UnifiedSessionsManager was described 02/2008 within the version 01.03.003a01 available from sourceforge.net/berlios.org.

The first basically stable applicable VSTACK feature set for the VM-based service composition by the UnifiedSessionsManager was described and implemented within the releases of the version 01.07.001 beginning at

08/2008 available from sourceforge.net/berlios.org.
These comprise the definition and refrence implementation for the automatic boot and shutdown of VSTACKs with multiple instances on each layer forming a tree structure. This particularly requires the controlled bottom-up startup and top-down shutdown of the mashed runtime dependencies.

3 A Detailed View to Service-Composition

There will be no further distinction between products related to the machine-focus, which are focussing actually on the 'technical convenience' of the machine-handling, but less on the contained 'service'. This is partly senseful of course, but it is the convenience of the technical 'administration functionality' only. Either for the systemsadministrator in a more comprising but granular feature scope, or for the user with a more restricted and abstract feature set. Whereas the UnifiedSessionsManager emphasizes the contained applications and shifting the focus to the provided services, which are represented by sets of applications.

One draft example for the difference might be a locator application for Circusses in your area for making a gift to your children. If you want to load a 'Circus-Exploration' service, first a 'GIS-Service', but second a 'Circus-Service', and probably an interconnecting 'Navigator-Service' are required. The machine-focused products support in analogy for the management of three Appliances as individual entities only, which you have to assemble by yourself.

What the UnifiedSessionsManager is designed for is the management of **ONE Combined Service** which is assembled by the three composite services - the **v-components**. The stacked VMs - **VSTACKS** - even offer **a generic assembly feature**, where the contained services could be either **referenced** only, or physically located within the superior VM by **containment**. This actually flexible interface is based on TCP/IP, like CORBA. The distinction to the current SOA pattern is the **completely contained runtime-environment** within each v-component, offering almost independent assembly modules.

Last but not least, when the OSs support for nested VMs is present, these vertically defined logical structure is going to be physically executed in a flat matrix structure of multiple cores. It is expected that in future hundreds, even thousands of cores are going to be available. Making this approach an additional component model for software development and flexible applications management even for execution on one physical machine only.

The personal service composition - **The ServiceComposer** - is implemented as draft for now on application level, but is already present perfectly on the desktop level with extensive configuration features.

The following example is just a gimmick, but the implementation of 'moving and resizing appliances' on your desktop with animated VM and HOSTs Consoles shows the available configuration and customization capabilities. The **-g** option is a superset of the **–geometry** option of X11. The following examples demonstrate the neatless interface, where the type of the session is going to be deleted for unambitiously defined entities based on a pre-created inventory database.

For **QEMU/KVM**:

```
for i in 10 20 30;do ctys -t QEMU -create=l:myApp,reconnect -g ${i}x100+${i}+100;done
```

For **VirtualBox**(TM):

```
for i in 10 20 30;do ctys -t VBOX -create=l:myApp,reconnect -g ${i}x100+${i}+100;done
```

For **VMWare-Server-1/2+Player-1/2/3+WS-6/7**(TM) - next **VMWare-ESX+ESXi**(TM):

```
for i in 10 20 30;do ctys -t VMW -create=l:myApp,reconnect -g ${i}x100+${i}+100;done
```

For **Xen** - next **XenServer**(TM):

```
for i in 10 20 30;do ctys -t XEN -create=l:myApp,reconnect -g ${i}x100+${i}+100;done
```

For **RDP-Console**:

```
for i in 10 20 30;do ctys -t RDP -create=l:myApp -g ${i}x100+${i}+100;done
```

For **VNC-Console**:

```
for i in 10 20 30;do ctys -t VNC -create=l:myApp -g ${i}x100+${i}+100;done
```

And similarly for **X11-Console**:

```
for i in 10 20 30;do ctys -t X11 -create=l:myApp -g ${i}x100+${i}+100;done
```

That's all to be customized.

The presented desktops and workspaces on the home page are just started within seconds by an ordinary Gnome menu entry - even though partly containing dozens of VMs on several hosts and additionaly several guest logins by VNC, RDP, X11, etc..

4 Datacenter and Applications Management

Now to the datacenter and applications management aspect for administrators. The current implemented version has some gaps related to typical production application, particularly for broader client management. The features for the management of servers in the back-end by the administrator may fit quite good, in general the needs for an administrators swiss-army-knife may be matched perfectly. The missing features are going to follow soon, which includes some re-coding, performance enhancements, and the introduction of optional server deamons. Also the final integration of the commercial enterprise degree products VMware-ESX(TM) and Citrix-XenServer(TM) embedded into an additional professional degree graphical user interface.

The sytems administrators tasks of the future are expected to be joined with the application administrators responsibilities. There might be particularly no extended general outsourcing paradigma into Public-Clouds, but a case-by-case approach for additional on-demand options. The so called Private-Clouds may evolve quite similar to nowadays inhouse services, which could be easily extended by on-demand-resources from a Public-Cloud.

The responsibility for the administrator is expected to be the management of the back-end services, including the management of the virtual client-services. These have quite similar algorithms for execution environment constraints. The typical constraint migt be a specific hardware or a specific set of software to be installed in the requested machine founding a service for a specific task. The constraint apply to the physically accessed machine of a user, e.g. requiring a locally attached label-printer.

A typical constraint for the assignment of a service aspects is a personal printer. For employees with a lot of 1-2 page printouts a department printer may not be suitable, also the previous example of a label-printer attached to the physical workstation may require a specific software including drivers. For other tasks the label-printer may not be required.

In a future load-balancing scenario you can say now, that the user requires a printer-aware-VM e.g. with an installed label-application when he is working in the store. Afterwords in the office not. But when writing the bill for his travel expenses the application composition may be different than for the construction task of the engineering project he is working on. This could have an impact on the available floating-license pool of contained applications.

So when assigning the VM to a specific PM from a pool of available, the contained application has to be recognize by the balancer. Particularly when a market for appliances as lean applications evolves in a broader range.

This aspect is already designed into the UnifiedSessionsManager by present specific attributes and application specific custom fields. But once again, the UnifiedSessionsManager for now only draftly implements this feature on application level. But the desktop based integration of distributed services is already present and fully operational.

The next aspect is the security related to the access permissions. The administrator is supported for this by system features for user permissions and various features of the UnifiedSessionsManager like the provisioning of individual database sets, which comprise the 'known as accessible' VMs. So due to the basic approach of application of available system services, the user access could be provisioned by standard mechanisms.

5 Current State and Open Issues

The implementation for now is - due to the typical evolution from an applied solution - based on scripting by bash and Python. The emphasis was set on a modular expandable and replaceable software architecture, thus providing the basic structure for the furture migration. Even though, the current available implementation fits the requirements for individual usage and is fully production ready, the application within environments for a huge number of users may cause some performances degradation. Also some feature enhancement is required. Thus based on the current implementation the key challanges will be worked on now for consolidation towards a major enterprise environment.

The main enhancements planned to be implemented and added to the service concept first are:

1. Recoding.
2. Introduction of optional server deamons.
3. Introduction of LDAP based shared data.
4. Introduction of an optional graphical user interface.
5. Introduction of Microsoft-Windows(TM) based features and agents.

Therefore some funding and probably support is required.

6 SEE ALSO

Datasheets:
UnifiedSessionsManager - Virtualisation and Cloud-Computing as a personal Workspace

Manuals:
ctys(1) , *ctys-distribute(1)* , *ctys-createConfVM(1)* , *Command-Reference(*)* , *HowTo(*)* , *User-Manual(*)*

Use-Cases:

- **Desktop Automation - Desktop-Level Service Composition:**
  ctys-configuration-Gnome(*)

- **Plugins:**
  ctys-uc-CLI , *ctys-uc-PM* , *ctys-uc-QEMU(*)* , *ctys-uc-RDP* , *ctys-uc-VBOX(*)* , *ctys-uc-VMW(*)* , *ctys-uc-X11* , *ctys-uc-XEN(*)*

- **GuestOSs:**

  ctys-uc-CentOS(*) , *ctys-uc-Debian(*)* , *ctys-uc-Enterprise-Linux(*)* , *ctys-uc-FreeBSD(*)* , *ctys-uc-Mandriva(*)* , *ctys-uc-OpenBSD(*)* , *ctys-uc-OpenSUSE(*)* , *ctys-uc-RHEL(*)* , *ctys-uc-Ubuntu(*)*

  ctys-uc-Android(*) , *ctys-uc-MeeGo(*)*

  ctys-uc-QNX(*) , *ctys-uc-uCLinux(*)*

  ctys-uc-MS-Windows-NT(*) , *ctys-uc-MS-Windows-2000(*)* , *ctys-uc-MS-Windows-2003(*)* , *ctys-uc-MS-Windows-2008(*)* , *ctys-uc-MS-Windows-7(*)* , *ctys-uc-MS-Windows-XP(*)*

(*) Contained in the DOC-Package only by CCL-3.0.

7 AUTHOR

Maintenance:	<acue_sf1@sourceforge.net>
Homepage:	<http://www.UnifiedSessionsManager.org>
Sourceforge.net:	<http://sourceforge.net/projects/ctys>
Berlios.de:	<http://ctys.berlios.de>
Commercial:	<http://www.i4p.com>

8 COPYRIGHT

The whole document - all sections - is/are defined as invariant.
For additional information refer to enclosed Releasenotes and License files.