Virtualisation and CloudComputing

The impact of virtualization on software architectures and lifecycles

ACCU-2012

Arno-Can Üstünsöz

Ingenieubüro Arno-Can Üstünsöz

2012.04.27

Agenda

- **Evolution of Virtualisation and CloudComputing**
  - A short definition of the terms Virtualization and CloudComputing.
- **Virtualisation and CloudComputing - A Marketing Mix**
  - A focusing direction for virtualization and CloudComputing application.
- **An OpenSource Example as a practical approach**
  - A future design paradigm and its infrastructure requirements.
- **Design impact on Target Applications**
  - Some examples of application patterns for target applications.
- **Management of test and Development Environments**
  - An application covering the presented requirements.

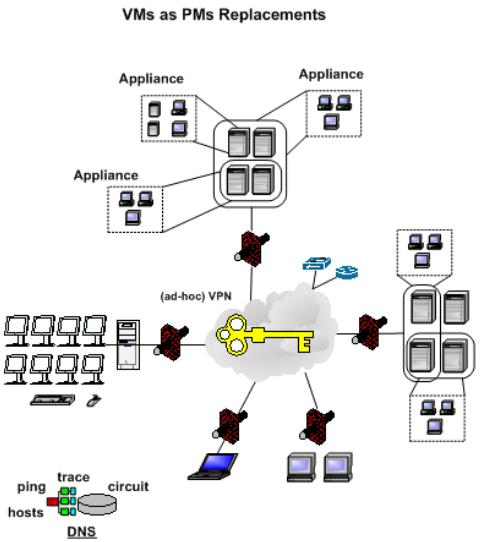
Evolution of Virtualisation and CloudComputing

- **A conceptual blueprint of Virtualisation and CloudComputing**
  - The underlying basic concept of **CloudComputing** is as stated by Sun-Microsystems, Inc. in the 80s:

    'The network is the computer.'
  - The mayor distinction results from the introduction of virtual machines:

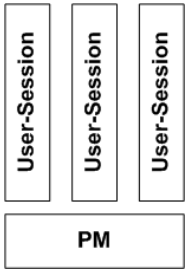
    + ' VMs.'
  - This enables the mobility of - even 'living' - machines, resulting in the flexibility of the execution location.

A conceptual blueprint



A conceptual blueprint

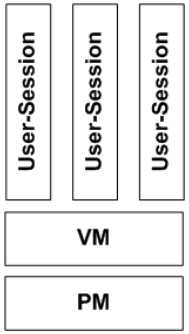
- The application of advanced virtualisation requires the definition of some basic design concepts for the categorisation and description of structures.



Standard user access to PMs.

A conceptual blueprint

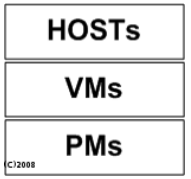
- Introduction of VMs



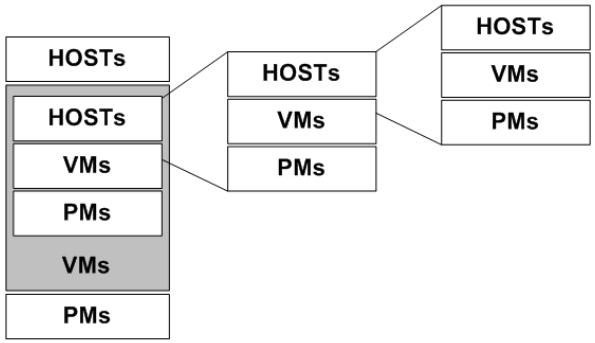
Standard user access to VMs.

Evolution of Virtualisation and CloudComputing

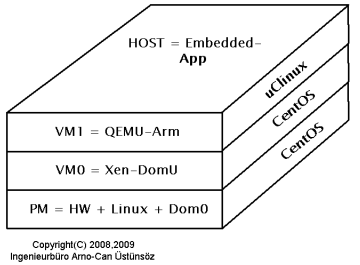
- **VMs - vStacks**
  - The introduction of VMs sets up a layerd architecture similar to the ITU-T ISO-Layering - vStacks. The definition of the 3-layer model introduces a conceptual building block for the description of virtual containers - vStacks.



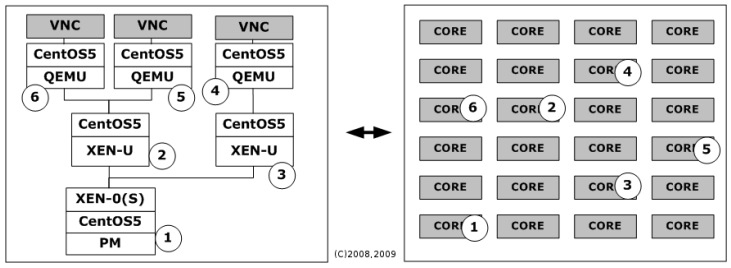
- **nested vStacks - Multi-Layer vStacks**
  - The vStacks could be executed as host applications theirself. The repetitive nesting defines now nested vStacks as Multi-Layer vStacks.



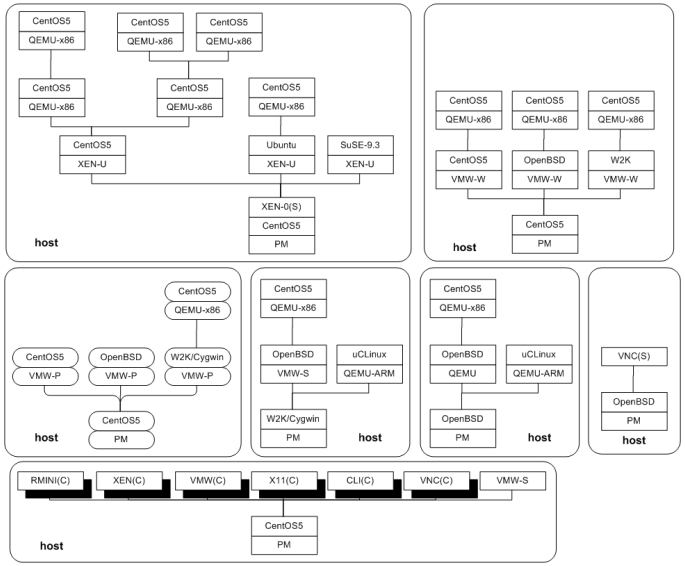
- **nested vStacks' - Multi-Layer vStacks**
  - The vStacks could be depicted in accordance to B-ISDN Management pane representation.

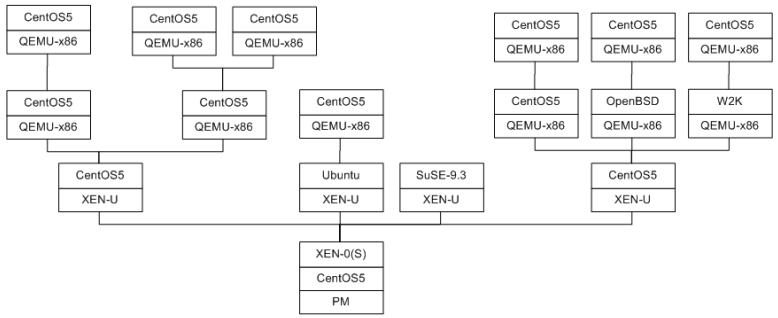


- A Real-World application:
  Nested VMs on Many-Core Systems - From single to many-core
  - The introduction of **nested VMs** leads to the concept of the virtual containment as a vertical logical tree structure.



  - The logical tree is mapped to a flat array of CPUs/Cores, resulting in neglible performance loss.





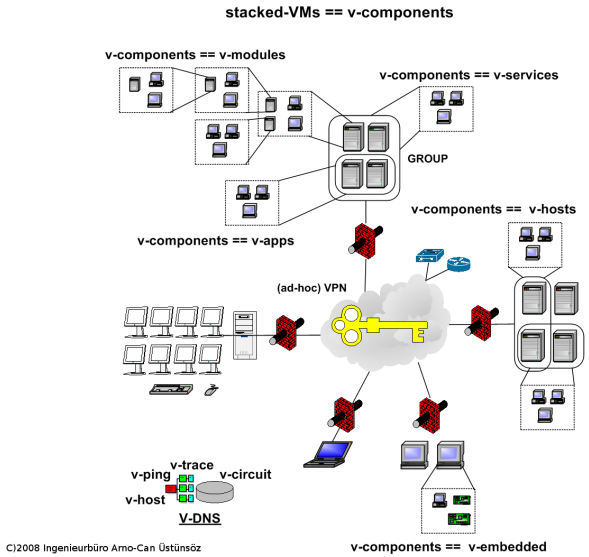
- **...a conceptual blueprint of Virtualisation and CloudComputing**
  - The introduction of multi-core systems, and particularly the upcoming many-core systems enables the extension of the concept of software components to complete VMs.
  - The introduction of virtual CPUs by hypervisors enables the scalability and dynamic reconfiguration.
  - The application of CPU emulators by 'hypervisors' enables the cross-execution of multiple achitectures - either for Mixed-Operation or security enhancement.
  - The enhanced processing capability of embedded SOC devices including multicore-processors brings the former data-center 'into your pocket'.

- **...a conceptual blueprint of Virtualisation and CloudComputing**
  - These extended conceptual elements provide for the introduction of a new abstraction element - the

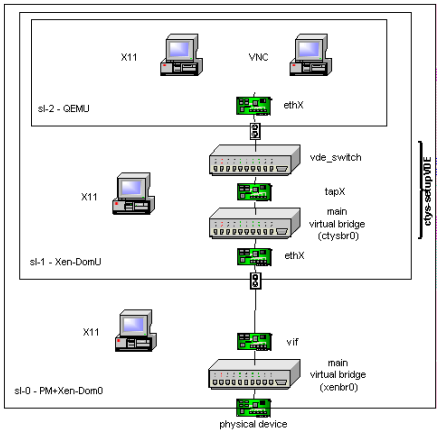
v-components

==

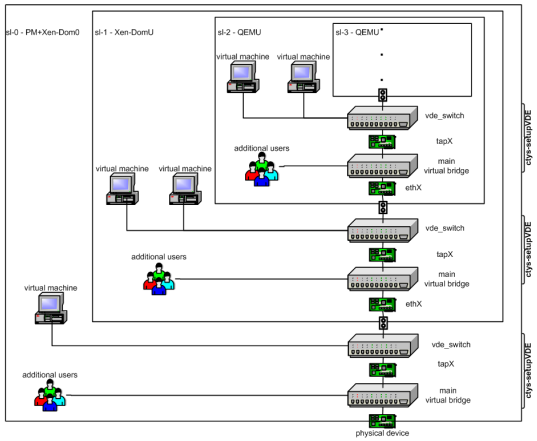
v-modules
v-apps
v-services
v-hosts
v-embedded



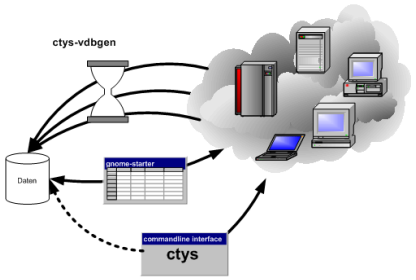
- **nested vStacks' - Network Layer**
  - The networking required for vStacks involves multiple layers of data exchange comprising chains of multiple virtual networking devices.



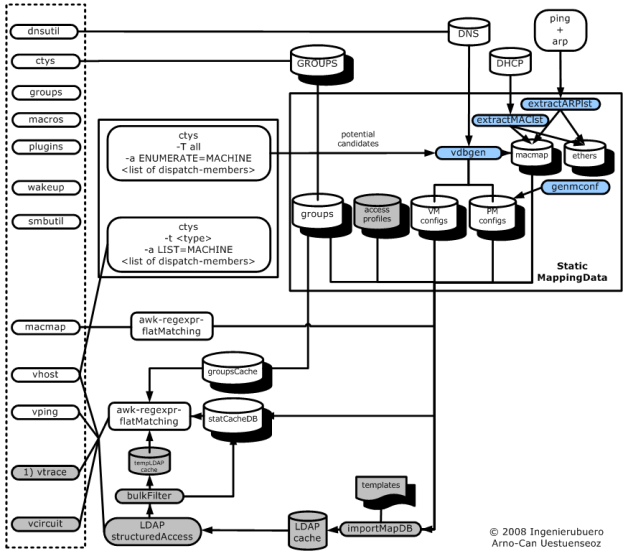
- **nested vStacks - Network Layer**
  - The required amount of networking entities for vStacks could expand quickly.



- **nested vStacks - Network Layer**
  - The application of networks requires flexible and automated network facilities on user-level. This is particularly required for cloning of machines.

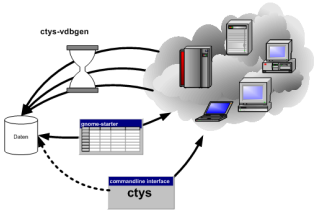


- **Address Resolution Infrastructure - Network Layer**



Evolution of Virtualisation and CloudComputing

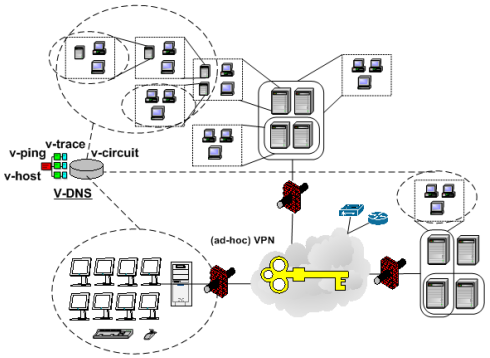
- **Inventory Database** and **Network Layer**
  - The integration of a Cloud-Inventory database and the network information is required for automation and ease of application for flexible Cloud-Services.



  - This is particularly mandatory for the individual creation and configuration of v-components. A typical case for individual test environments.

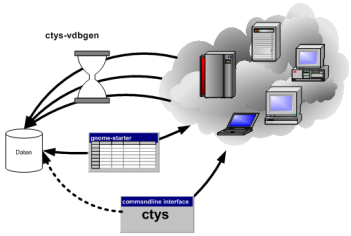
Evolution of Virtualisation and CloudComputing

- **Inventory Database** - **Views**
  - The individual application of huge amounts of VMs requires databases with individual views.



Evolution of Virtualisation and CloudComputing

- **CloudComputing** = **Inventory Database** + **Views**
  - CloudComputing could be 'draftly' defined as sets of machines managing and executing VMs, controlled by a load-balancer.



  - The comprising set of concepts and facilitities for the operations of cloud services is defined by the previous slides now.

Virtualisation and CloudComputing - A Marketing Mix

- **Market Development and Business Cases**
  - Understanding the Marketing Mix of CloudComputing may help to focus on the own actual requirements.
  - The Marketing Mix of Cloud computing is quite similar to the Internet-Hype era.
    1. Focus on huge enterprise customers - early birds.
    2. Market virtualisation and proclaim 'the Cloud'.
    3. Market Private-Clouds - medium level customers.
    4. Market Public-Clouds - enter the mass market.
  - The available services may be for the software development of various interests. Either as a consumer, or as a product supplier. To provides services could be of interest for support and maintenance, but also for individual services.

Virtualisation and CloudComputing - A Marketing Mix

- **Market Development and Business Cases**
  - The firts trial of categorisation for products and subproducts with main focus on commercial business cases.
    - Server-Virtualisation
    - Desktop-Virtualisation
    - Client-Virtualisation
  - The actual cases or better single case is:
    - Virtualisation with connected desktop.
    - Virtualisation in headless mode - no active desktop connection.
  - The remaining feature configuration is a matter of the contained OS and applications.

Virtualisation and CloudComputing - A Marketing Mix

- **Nested VMs on Many-Core Systems**
  - The basic concepts are now spreading datacenters and will be expanded continuously.
  - The next step of the overall Marketing Mix now brings the datacenter to small companies and end-users, some examples for the mass market are
    - 'managed virtual CPE' - Customer Premisses Equipment - as 7x24h local proxy for private users: e.g. ®AVM Fritz-Box
    - 'private client'/'private cloud' as a 'Secure-Personal-Datacenter' and local cache: ©UnifiedSessionsManager
    - a peer-to-peer-cloud or managed private cloud as a 'Secure-Distributed-Datacenter' for private groups and commercial teams: ©UnifiedSessionsManager

Virtualisation and CloudComputing - A Marketing Mix

- **Nested VMs on Many-Core Systems**
  - These concepts are now also putting their shadow on mobile devices and embedded systems.

    Two current examples are:
    - ©Smartphones e.g. by ®Apple, ®LG, ®Nokia, ®Samsung, ...
    - ©'Raspberry IP' for the scale of the next generation of home stations.
  - The home networks and smart grids are going to be designed as integrated private clouds for remote management purposes too.

So it is going to be the paradigm of 'nested VMs' in your pocket as well.

Virtualisation and CloudComputing - A Marketing Mix

- **What is actually required for R&D?**
  - The application for SW development may mainly require the setup and operations of local VMs, thus seems to be a private cloud for test and development.
  - The toolset for test and development requires support for heteregeneous hypervisors with local inventory databases providing custom views.
  - The development of cloud based products may require arbitrarily additional facilities.

An OpenSource Example as a practical approach

- **UnifiedSessionsManager V.01.x - Masada**
  - The UnifiedSessionsManager - V.01.x Masada - is a 'grown' development for the application of Virtualisation and CloudComputing in Software and Hardware development since 06/2007, available since 02/2008.
    - The UnifiedSessionsManager is a 'grown' product mainly specialized for the operations of test and development environments.
    - Therefore the actual implementation represents almost any required aspect as a running prototype.
    - Due to the 'grown' history requires some skills for operations, lacks performance and maintainability.

An OpenSource Example as a practical approach

- **UnifiedSessionsManager V.02.x - Mjoelnir**
  - The new version - V.02.x Mjoelnir - is extended to a product covering enterprise degree features combined with private end-user market requirements.
    - The UnifiedSessionsManager V.02.x is complete recoded and enhanced by redesign.
    - The new version is mainly a pure Python implementation with GUI and enterprise degree backend server.
    - The new version is targeting the complete range of requirements for commercial and non-commercial application.
    - Due to the new design with competitive performance, cost-efficient maintainability and easy-to-use operations are provided.

An OpenSource Example as a practical approach

- **Required Design principles**
  - The first and basic requirement for future and current application of hypervisors is the

    vendor independency.

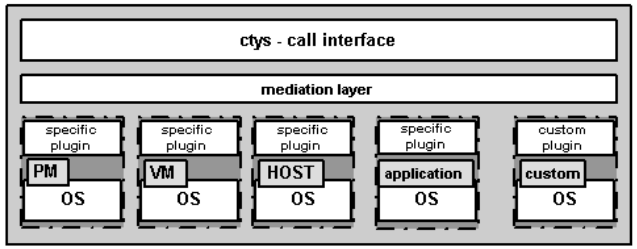
    Sounds simple, but is crucial.

    This is particularly challanging for the vendors of hypervisors, requiring the adaption of their business models.

An OpenSource Example as a practical approach

- **Required Design principles - vendor independency**

  Vendor independency by dynamic configuration.



An OpenSource Example as a practical approach

- **Required Design principles - vendor independency**

  Vendor independency as part of the interface.

  Hypervisor and Emulators:
  - KVM: ctys [-t kvm] -a create=label:mySession myuser@myhost
  - Qemu: ctys [-t qemu] -a create=label:mySession myuser@myhost
  - VirtualBox: ctys [-t vbox] -a create=label:mySession myuser@myhost
  - VMware: ctys [-t vmw] -a create=label:mySession myuser@myhost
  - Xen: ctys [-t xen] -a create=label:mySession myuser@myhost

  Graphical Userinterfaces:
  - CLI: ctys -t cli -a create=label:mySession myuser@myhost
  - RDP: ctys -t rdp -a create=label:mySession myuser@myhost
  - VNC: ctys -t vnc -a create=label:mySession myuser@myhost
  - X11: ctys -t x11 -a create=label:mySession myuser@myhost

An OpenSource Example as a practical approach

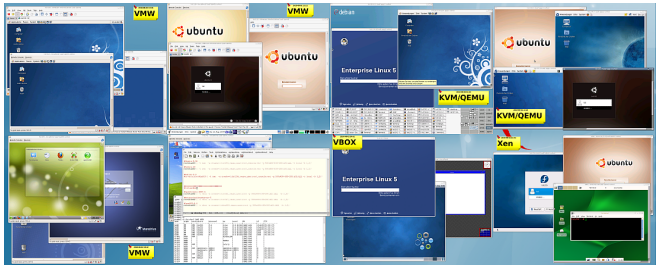
- **Required Design principles - vendor independency** The customer 'backend' platforms may spread even more than until now:
  - Hypervisors and Emulators:
    - KVM, QEMU, VirtualBox, VMware, Xen, ...
  - Operations Systems:
    - Linux(CentOS, debian, Fedora, Mandriva, OpenSUSE, Ubuntu,...),
    - 'Adapted' Linux(Android, MeeGo,...),
    - FreeBSD, OpenBSD, OpenSolaris, Solaris
    - MS-Windows(Windows-7/8, Windows-2003/2008, ...)

An OpenSource Example as a practical approach

- **Required Design principles - vendor independency** The customer 'frontend' platforms may spread even more than until now:
  - Remote Desktops:
    - 'X11'
    - VNC(TightVN, TigerVNC, RealVNC,...)
    - RDP(rdesktop, tsclient,...)
    - Consoles
  - Graphical Userinterfaces:
    - X11 based: FVWM, Gnome, KDE, Xfce
    - MS-Windows
    - Apple-Mac-OS

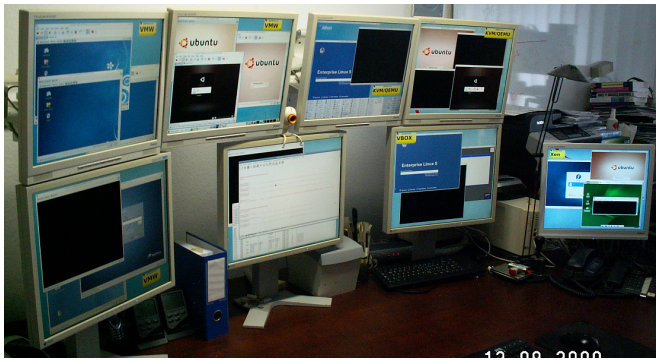
An OpenSource Example as a practical approach

- **Required Design principles - vendor independency**



An OpenSource Example as a practical approach

- **Required Design principles - vendor independency**



An OpenSource Example as a practical approach

- **Required Design principles - vendor independency**

The last screen-photo shows, desktop virtualisation is almost just a graphical frontend, either a forwarded display, or a local client.

So virtualisation technically comprises client, server, and desktop virtualisation. The term just describes the major application type.

Therefore the design of software and IT systems may handle virtualisation as a container for a v-component only. This may or may not have a graphical frontend including a desktop representation.

An OpenSource Example as a practical approach

- **Required Design principles - vendor independency**

One special case to be handled when multiple hypervisors are involved is the coordination of similar but independenly managed IDs. For example the Famebuffer addressing of VNC. The Desktop-ID or connection port is used by the server components of:

  - KVM
  - Qemu
  - Xen
  - VMware-WS(optional)
  - VNC

Each has it's own algorithm for automatic or manual assignment. In case of multi-user environment also preferences and ranges have to be coordinated. Each scope is local only on the targeted server.

An OpenSource Example as a practical approach

- **Required Design principles - vendor independency**

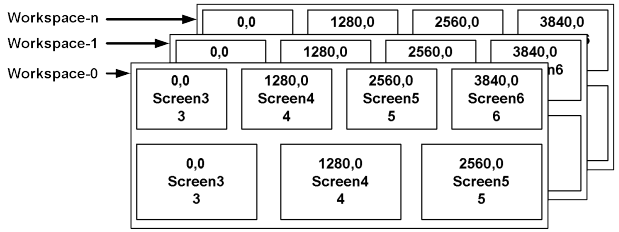
This also shows the importance of extended user frontends when huge amounts of information has to be worked out. This could be basically any type of multi-aspect data, e.g. multiple stock diagrams, or network and systems management data.

But particularly for software development and test the usage of multiple monitors provides a number of advantages. This is comprises the coding as well as the test and debugging of processes.

When multiple platfroms and architectures has to be supported, the could be provided for test purposes cost efficient by virtualisation. Composed into a combined test environment by screen integration.

An OpenSource Example as a practical approach

- **Required Design principles - vendor independency**



An OpenSource Example as a practical approach

- **Required Design principles - comprising network support**

  The main aspect even of pure virtualisation is the quality of the provided networking. This comprises the NIC driver of course, which e.g. in case of KVM and libvirtio have a performance boost of 10. But also the management of network configuration and reconfiguration for shifted/moved devices.

  The first identifier to recognise is the MAC address. This has to be unique within the whole runtime scope, particularly in cases of dynamic reallocation into different management domains. Any new solution should have at least a roadmap for IPv6 support.

An OpenSource Example as a practical approach

- **Required Design principles - comprising network support**

  The opensource example contains the basic networking tools:
  - ctys-dnsutil
  - ctys-extractMACIst, ctys-extractARPIst
  - ctys-macmap
  - ctys-smbutil
  - ctys-vhost
  - ctys-vnetctl/ctys-setupVDE

  These tools provide the complete scope of MAC and IP address management including an integrated database. The database provides for address mapping and address based searchs of VMs.

An OpenSource Example as a practical approach

- **Required Design principles - inventory management**

  The opensource example contains automation tools for the creation of inventory databases.
  - These database is particularly critical for stacked and distributed VMs, because the balanced allocation of a VM for execution requires offline information, which is not available online.
  - The database provides for VM allocation in multipath execution providers, and external clouds too.
  - It gives an overview of available VM sets, including information for contained OSs.
  - Provided tools are:
    - ctys-vdbgen
    - ctys-vhost

An OpenSource Example as a practical approach

- **Required Design principles - inventory management**

  The consequently next part to consider is the automation of cloning of VMs. This has also to provide multivendor support, including the integration into several IDs applied by the vendors into the vendor independent inventory database. This particularly also comprises 'pseudo-off-line' instances, which may be executed 'headless' by an cloud-provider. The same is true for stored VMs on a mobile device, either a storage only, or processing device.
  - Provided tools are:
    - ctys-attribute
    - ctys-config
    - ctys-convert
    - ctys-createConfVM
    - ctys-config
    - ctys-cloneVM

An OpenSource Example as a practical approach

- **The integration of public and private clouds**

  This also shows the importance of extended user frontends when huge amounts of information has to be worked out. This could be basically any type of multi-aspect data, e.g. multiple stock diagrams, or network and systems management data.

  But particularly for software development and test the usage of multiple monitors provides a number of advantages. This comprises the coding as well as the test and debugging of processes.

  When multiple platfroms and architectures has to be supported, these could be provided for test purposes cost efficient by virtualisation. Composed into a combined test environment by screen integration.

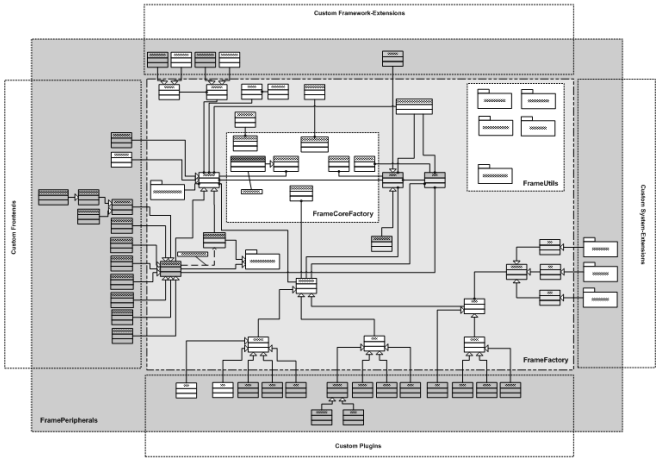
An OpenSource Example as a practical approach

- **Sum up of basic requirements for virtualized development environments**

  A virtualisation and cloud solution for software development has to provide:
  - Multivendor capability.
  - Automated inventory management.
  - Subsets of inventory as views.
  - Multimonitor support.
  - Scripting interface.
  - Extendibility.

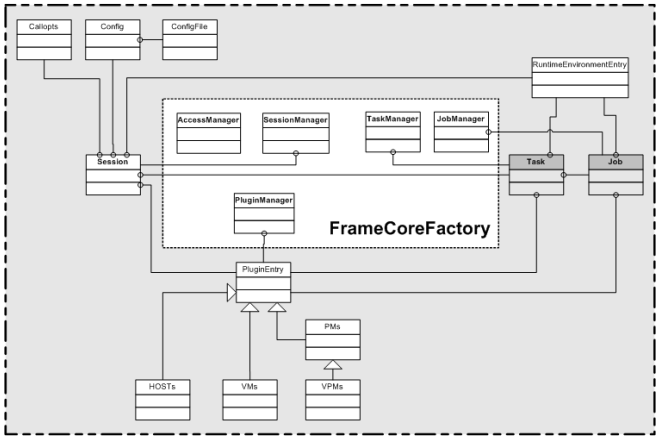
An OpenSource Example as a practical approach

- **A class Design for a Sessions Management Framework**



An OpenSource Example as a practical approach

- **A class Design for a Sessions Management Framework**



Design Impact on Target Applications

- **Application pattern - Datacenter**
  In datacenter applications the packaging of multiple VMs into a 'container-VM' simplifies the combined management. This is particularly supporting the reallocation of the whole set, the partitioning of security issues, and the combined resource allocation including the automated startup of hosting server.

  The security issues may be one specific case, when for example a virtual firewall - probably with a different architecture of the virtual CPU - is utilized for contained VMs. This could be applied for simplified partitioning of security zones within a datacenter, where the complete set could be reallocated without restrictions on the securtiy policies.

Design Impact on Target Applications

- **Application pattern - server systems**
  - Midrange systems are handled the same way as datacenters in general, only the scale of the handled data varies.
  - Small systems, particularly end-user systems are going to provide the same degree of strutural capability as datacenters. The only difference for a system of a private cloud may be the simplicity of application and the lesser requirement of provisioning of policies.
    The main application here is the provisioning of security applications and mobile virtual devices.
  - In the field of embedded systems with SOC at midrange and highend the introductin of VMs may cover various issues related to availability and dynamic reconfiguration.
    This could cover on-board networks of vehicles as well as ships, and airplanes.

Design Impact on Target Applications

- **Application pattern - general**
  - The introduction of 'vComponents' provides means of systems management by standard tools and interfaces, e.g. by standard IP management tools and standard/proprietary hypervisor interfaces.
  - This defines a runtime environment, where the administrator of the system could simply reconfigure a running distributed application or system.
  - The 'vComponents' particularly contain any required dynamic session data, e.g. including kerberos tickets, which enable for quite simple failover concepts.
  - The facilities for the dynamic reconfiguration are immanent, thus just require minor additional development costs.

Management of Test and Development Environments

- **An example OpenSource-Project - UnifiedSessionsManager** The implementation of almost all of the above requirements is provided beginning with 2/2008.
  - The current public version is provided as a 'production-prototype' by shell scripts with minor Python components for Linux/BSD/Solaris Platform. License if - GPL3 and/or CCL - UnifiedSessionsManager.org
  - The current version touched it's limits of perfomance when porting to MS-Windows(C) simply based on Cygwin.
  - A complete refactoring with significant performance and applicability enhancements is currently going to be finished and will be available soon. The first new Version is a mainly Python based with additional graphical user interfaces based on Qt and Web-Browsers.
  - Enhanced customization interfaces and a broad range of scalability including a server variant is provided.
  - Soon available versions are GPL-3, CCL-2, and a proprietary enterprise variant.

Management of modern Test and Development Environments

- **UnifiedSessionsManager** The implementation of almost all of the above requirements is provided beginning with 2/2008.
  - The current public version is provided as a 'production-prototype' by shell scripts with minor Python components for Linux/BSD/Solaris Platform. License if - GPL3 and/or CCL - UnifiedSessionsManager.org
  - The current version touched it's limits of perfomance when porting to MS-Windows(C) simply based on Cygwin.
  - A complete refactoring with significant performance and applicability enhancements is currently going to be finished and will be available soon.

...

That brings me to the end of my presentation,
thank for your attention.

Virtualisation and CloudComputing