Infrastructure Cloud Software Layers

Infrastructure-as-a-Service (IaaS) clouds comprise a large number of software layers: Each physical machine in the provider's datacenter runs a virtualization solution with virtual machines that can be accessed by the cloud users. Within these virtual machines (VMs), users typically install commodity operating systems like Linux or Windows and an application server that hosts another virtual machine (e.g. Java/JVM) for the actual program the user wants to run. This huge stack of software layers is a source of many security issues, which can be exploited sometimes even if the application does not use that vulnerable feature in particular.

Our approach tries to minimize the trusted code base for an individual service by configuring the software stack within a VM to include only the absolutely necessary features. Figure 2 shows the layout of such a tailored stack: A thin software layer specific to the virtualization interface of the cloud provider forms the basis for implementing a configurable set of network protocols and application-specific algorithms. Depending on which environment the service will operate in, the basic application can also be modularized into features that can be configured with regard to external requirements.

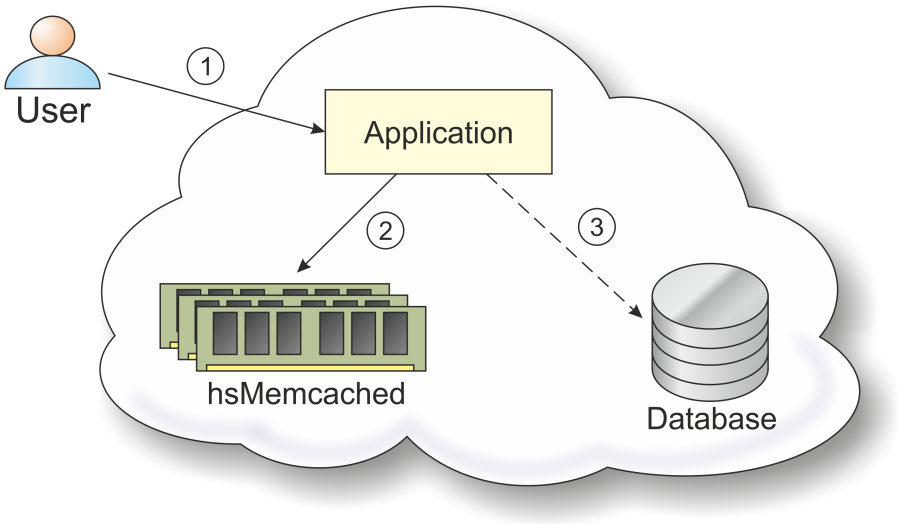

Figure 1: HsMemcached use case

HsMemcached: A Key/Value Cache in Haskell

In order to demonstrate the viability of our approach, we provide a reconfigurable implementation of a in-memory Key/Value cache service called hsMemcached. It is fully compatible with the well-known Memcached protocol. The typical usage scenario is shown in Figure 1 above: An user requests data from an application, e.g. a website with a summary of his power consumption in the previous month (1). Gathering this information requires a rather slow database query, so the application will consult hsMemcached (2) for the summary first. Only if the information is not present in the cache yet, the complex query has to be performed (3).

Apart from basic storage operations, the Memcached protocol also provides commands for atomic updates and manipulation of text strings and numbers. As these features are not required in every use case, this gives us plenty of opportunity for application-level tailoring.

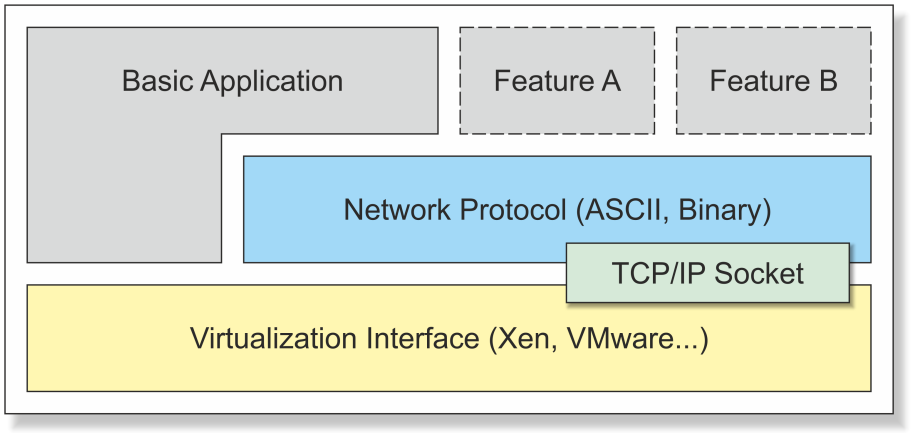

Figure 2: Tailored Service Layers

TAILORED KEY/VALUE STORE

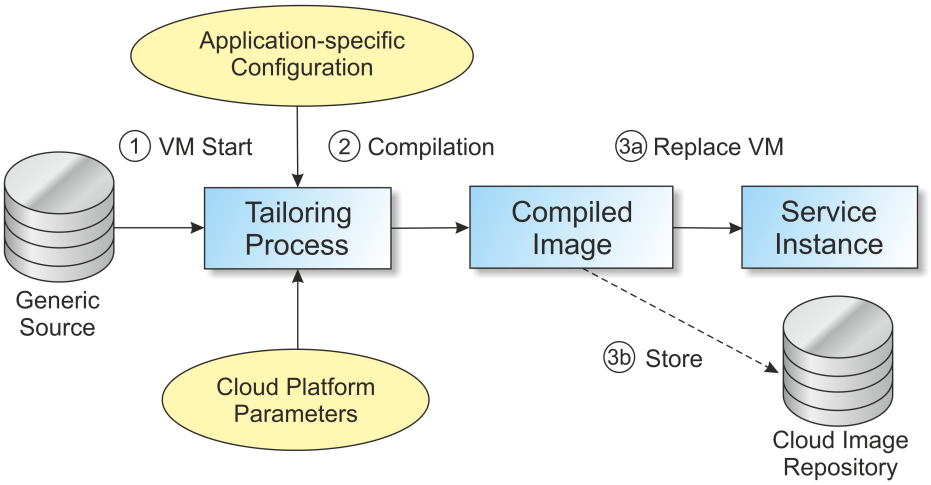

Figure 3: Tailoring process

Tailored Software in Cloud Environments

The process to create a tailored service instance in the cloud is depicted in Figure 3 and is entirely performed on the IaaS platform. Each service instance starts from a VM image containing the full source code repository of the service with code for each configurable feature (1). In the next step, a specific configuration is derived from information about the cloud platform (e.g. Hypervisor vendor) and requirements from the application that will use the service instance (e.g. Protocol support). The tailoring process then compiles the selected code pieces into a bootable VM image (2). In the final step, the created image is used to replace the tailoring VM with the actual service (3a). In case we want to start multiple instances with the same configuration at once, we optionally save a copy of the created image in the repository of the underlying IaaS cloud (3b).

Haskell Programming Language

The individual parts of the service infrastructure and the hsMemcached application are written in the Haskell programming language. Haskell is purely functional with a strong, static type system where computations with side-effects are represented as special types. This allows us to leverage the properties of the language to make components more accessible to formal verification methods and ensure safety properties due to its superior type system.

Our prototype interfaces with the Xen hypervisor using Galois' HaLVM, which is a thin operating system layer to provide the necessary runtime and library support. Large portions if this layer are also written in Haskell, allowing the implementation to take advantage of high-level language features and type-safety even on the system level and makes tailoring across all layers possible.

Further Information

Further information about Tailored Key/Value Store can be found under Deliverable „D2.1.2—Preliminary Description of Mechanisms and Components for Single Trusted Clouds".

Disclaimer

The TClouds project has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement number ICT-257243.

TClouds at a glance

Project number:
257243

TClouds mission:

- Develop an advanced cloud infrastructure that delivers computing and storage with a new level of security, privacy, and resilience.
- Change the perceptions of cloud computing by demonstrating the prototype infrastructure in socially significant application areas.

Project start:
01.10.2010

Project duration:
3 years

Total costs:
EUR 10.536.129

EC contribution:
EUR 7.500.000

Consortium:
14 partners from 7 different countries.

Project Coordinator:
Dr. Klaus-Michael Koch
coordination@tclouds-project.eu

Technical Leader:
Dr. Christian Cachin
cca@zurich.ibm.com

Project website:
www.tclouds-project.eu