# SIMCLOUD: RUNNING OPERATIONAL SIMULATORS IN THE CLOUD

**Annabell Langs[1], Claudia Mehlig[1], Stefano Ferreri[2], Mehran Sarkarati[3]**

***[1]Telespazio VEGA Deutschland GmbH***
***Europaplatz 5, 64293 Darmstadt, Germany***
***Email: annabell.langs@telespazio-vega.de, claudia.mehlig@telespazio-vega.de***

***[2]Terma GmbH***
***Europahaus, Europaplatz 5, 64293 Darmstadt, Germany***
***Email: stefano.ferreri@esa.int***

***[3]European Space Operations Centre***
***Robert-Bosch-Str. 5, 64293 Darmstadt, Germany***
***Email: mehran.sarkarati@esa.int***

## ABSTRACT

At ESOC, operational simulators are typically deployed as separate systems using dedicated hardware and a dedicated installation of the simulation infrastructure (SIMSAT) per simulator. To enable a more flexible use of resources and to ease deployment and validation of operational simulators, the SIMCLOUD activity targets a new Cloud based provisioning model, where SIMSAT acts as a Platform-as-a-Service (PaaS). The PaaS provides a dynamically scalable, shared runtime environment which can host multiple versions of multiple operational simulators simultaneously. Taking this concept one step further, multiple versions of the re-usable Generic Models and Ground Models are provided as Software-as-a-Service (SaaS) on top of the SIMSAT PaaS such that even more of the common SIMULUS infrastructure can be shared.
The solution presented here includes a new SIMSAT monitoring agent that manages SIMSAT daemons which run on shared physical machines as well as specific SIMSAT cloud daemons that are provisioned dynamically on virtual machines (VM) in the ESA cloud (based on VMware vCloud). The process of creating additional VMs is driven by mission resource requirements like CPU and RAM utilisation and the availability of existing resources. Apart from the prompting to confirm the automatic creation of a new VM in the case that not enough resources are available, the user is not confronted with any additional management overhead. This management including the connection, configuration and release of such VMs (done via secured connection and the VMware vCloud API) is instead handled by the monitoring agent. Moreover, for the support of multiple users and multiple versions of missions a new simulator deployment structure has been developed.
As part of the research activity the performance impact of running in a virtualised environment has been evaluated. This analysis includes a comparison of disk I/O rate, CPU usage and speed factor values of an operational simulator under different loads and for different hardware configurations using both virtual and physical servers.
We conclude this paper with an outlook on possible future developments.

## INTRODUCTION

Nowadays operational simulators are typically delivered to and deployed at ESOC as a package containing all software required to run the simulator. This includes the simulation infrastructure (SIMSAT), Ground, Reference Architecture and Generic Models and any other third party libraries and packages. This infrastructure and the mission specific simulator models are installed on dedicated hardware to ensure isolated execution. However this approach requires both a lot of resources that are statically occupied by a mission and a thorough management of the various machines and simulators by the mission teams.
The SIMCLOUD solution presented in this paper targets a new approach to simulator deployment and runtime management by following a cloud based provisioning model that allows a more flexible use of resources and an easier deployment and validation of operational simulators. For example, the new concepts enable deploying multiple versions of an operational simulator based on the same simulation infrastructure. Moreover, hardware resources such as server machines together with software resources such as a common infrastructure installation may be shared between missions, thus increasing the efficient utilisation of machines and thereby reducing costs.

### Platform-as-Service

In the SIMCLOUD approach, SIMSAT acts as a *Platform-as-a-Service* (PaaS) that allows users to run and manage simulators without the complexity previously imposed on them. The PaaS provides a shared runtime environment where multiple versions of multiple operational simulators can run simultaneously. The platform consists of the installation of a cloud-enabled version of SIMSAT, a web interface to deploy the simulators and the underlying hardware. It dynamically provisions additional virtual machines which act as hosts to the simulators in case a lack of

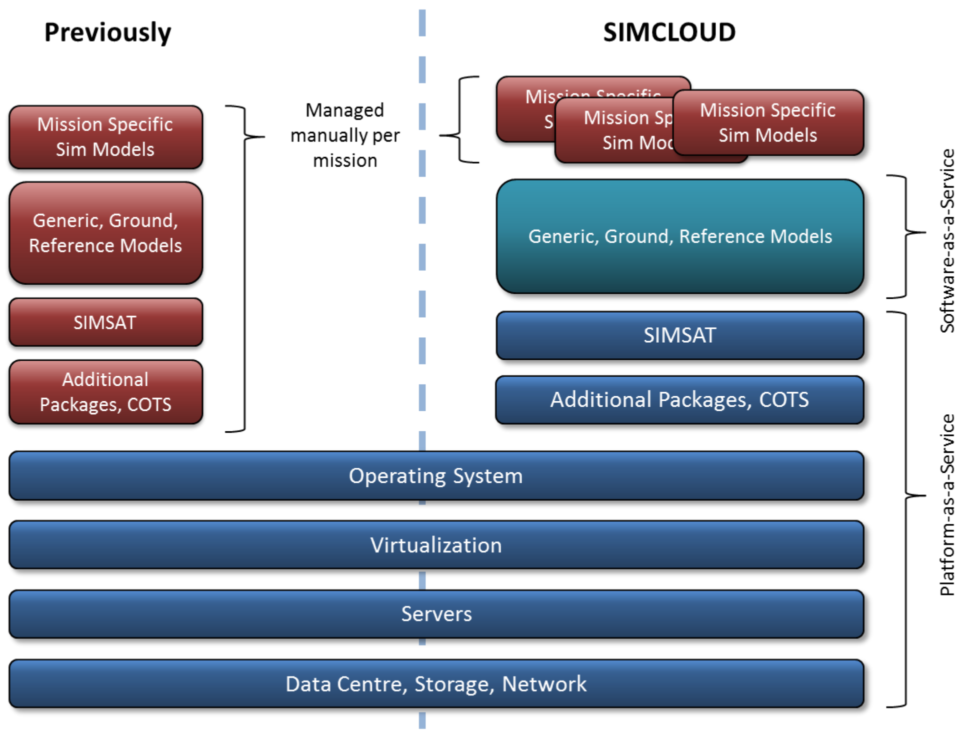

Previously
SIMCLOUD
Mission Specific Sim Models
Managed manually per mission
Mission Specific Sim Models
Mission Specific Sim Models
Mission Specific Sim Models
Generic, Ground, Reference Models
Generic, Ground, Reference Models
Software-as-a-Service
SIMSAT
SIMSAT
Additional Packages, COTS
Additional Packages, COTS
Platform-as-a-Service
Operating System
Virtualization
Servers
Data Centre, Storage, Network

**Figure 1 SIMCLOUD Overview**

available resources is detected. The solution scales with the amount of operational simulators running at any time and frees up resources once they are not used by a mission anymore.

### Software-as-a-Service

While the PaaS allows sharing hardware resources and the basic installation of SIMSAT amongst the missions, further elements of the underlying SIMULUS infrastructure such as the Generic Models, the Ground Models and the Reference Architecture are often identical amongst missions and can equally be shared. In the SIMCLOUD solution these elements are adapted to act as a *Software-as-a-Service* (SaaS). This requires a new concept of the deployment of simulators for use with SIMCLOUD. Additionally, the SaaS approach allows to easily integrate new versions of infrastructure models.

## DEPLOYMENT CONCEPT

Currently, the installation of a simulator into SIMSAT mostly consists of populating the single *share* directory of SIMSAT with simulator components such as configuration files and libraries as well as installation of required dependencies like Ground Models. To support parallel execution of different simulators with different configurations within a single shared installation a new deployment concept has been introduced in order to separate data from different missions or versions. This concept allows for changes or even deletion of files without a direct impact on other missions in the shared environment.

Instead of using just one share folder multiple such share folders are needed for a SIMSAT PaaS. Each folder represents the installation of one operational simulator with a specific version. This means that a simulator is still installed using the same directory structure thus minimizing the impact on simulator development. The individual *deployment* areas (share folders) are registered with the SIMSAT PaaS via a top-level *deployments* folder.

Each individual simulator *deployment* consists of binaries and configuration data. Binaries represent the static artifacts that are provided by the simulator developers, for example the library files, while the configuration data consists of all the files that configure the simulator, namely: Ground files, kernel architecture file, scripts & script engine configuration files, display files including ANDs and graphs, breakpoints, thermal configuration and OBSW images.
The configuration files are initially provided by the simulator developers; however they are continuously changed by the simulator users, for example breakpoints are generated, scripts are created or modified.

In the case a simulator needs to provide its own versions of one of the SaaS components, it may do so. To support this, a fall-back mechanism has been implemented allowing to load the mission specific versions if provided - otherwise using the ones of the shared SIMULUS infrastructure.

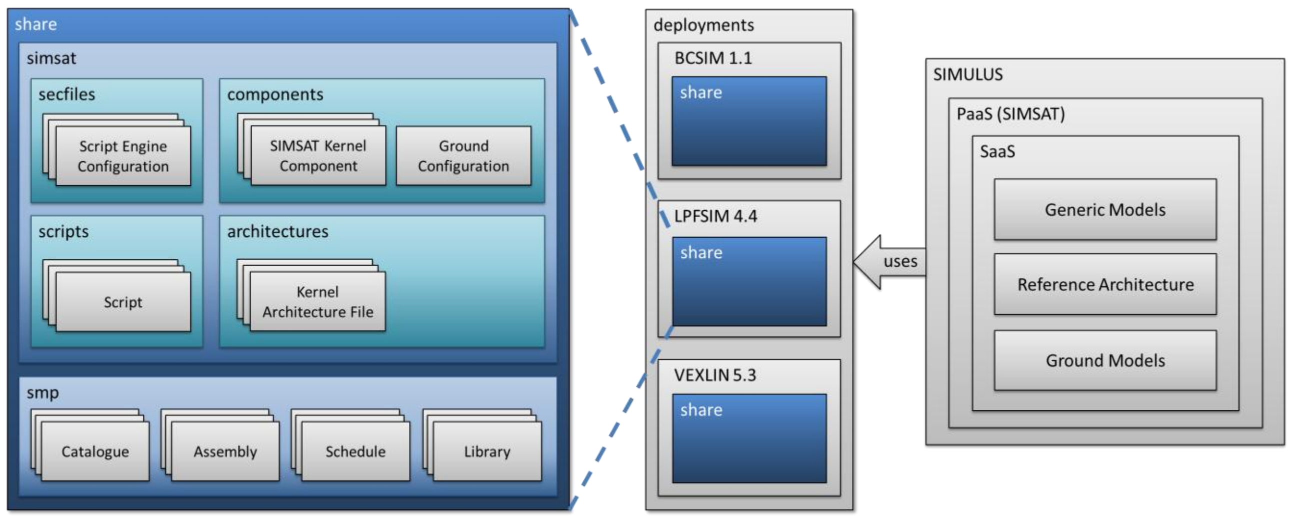


**Figure 2: Deployment Concept**

The SIMULUS infrastructure models have been adapted to work with the new deployment concept in a shared environment. Minor changes were necessary in the build and logging systems of the models.

While in a traditional single user and mission environment the SIMSAT share directory only needs to be accessible by the user that starts the SIMSAT daemon that accesses this directory, the deployment concept is based on the idea of a shared storage. The SIMCLOUD concept uses an NFS server and client architecture where the deployments folder together with the SIMSAT PaaS is installed on a shared drive. This shared drive needs to be accessible for all SIMSAT machines that are part of the SIMSAT cloud.

## CLOUD BASED PROVISIONING

Besides changing the deployment concept it has also been necessary to adapt the process of starting simulators in SIMSAT. The following steps are involved:

| **Step** | **Original SIMSAT** | **SIMCLOUD** |
|---|---|---|
| Daemons | Start of the SIMSAT daemon by the simulator user on a machine later hosting the simulator. | Start of the SIMSAT Cloud Agent and Daemon by the SIMSAT cloud operator once on a central PaaS server. |
| MMI | Start of a SIMSAT MMI to open a simulation on a specific daemon. | Start of a SIMSAT MMI to open a simulation by simulation name and version. |
| Simulation Selection | The daemon reports the available simulators of the single architectures folder as part of the share directory back to the user. | The agent reports the available simulators of the deployments folder back to the user. |
| Simulation Launch | Launch of the specified simulation. This involves the SimHost running the Kernel, the Ground Model Manager to read configuration files and the Smp2Adapter to load a number of libraries, catalogues, assemblies, schedules and further mission specific data. | Launch of the specified simulation involving an automatic cloud based provisioning of hardware resources in addition to the previous processes adjusted to the new deployment concept. |

### The SIMSAT Cloud Agent and Daemon

The process of manually starting a SIMSAT daemon on a machine by the user is replaced by the concept of a so-called SIMSAT *Cloud Agent* automatically starting SIMSAT *Cloud Daemons*. The Cloud Agent is accessed by the MMI and thus the single point of contact for the end user.

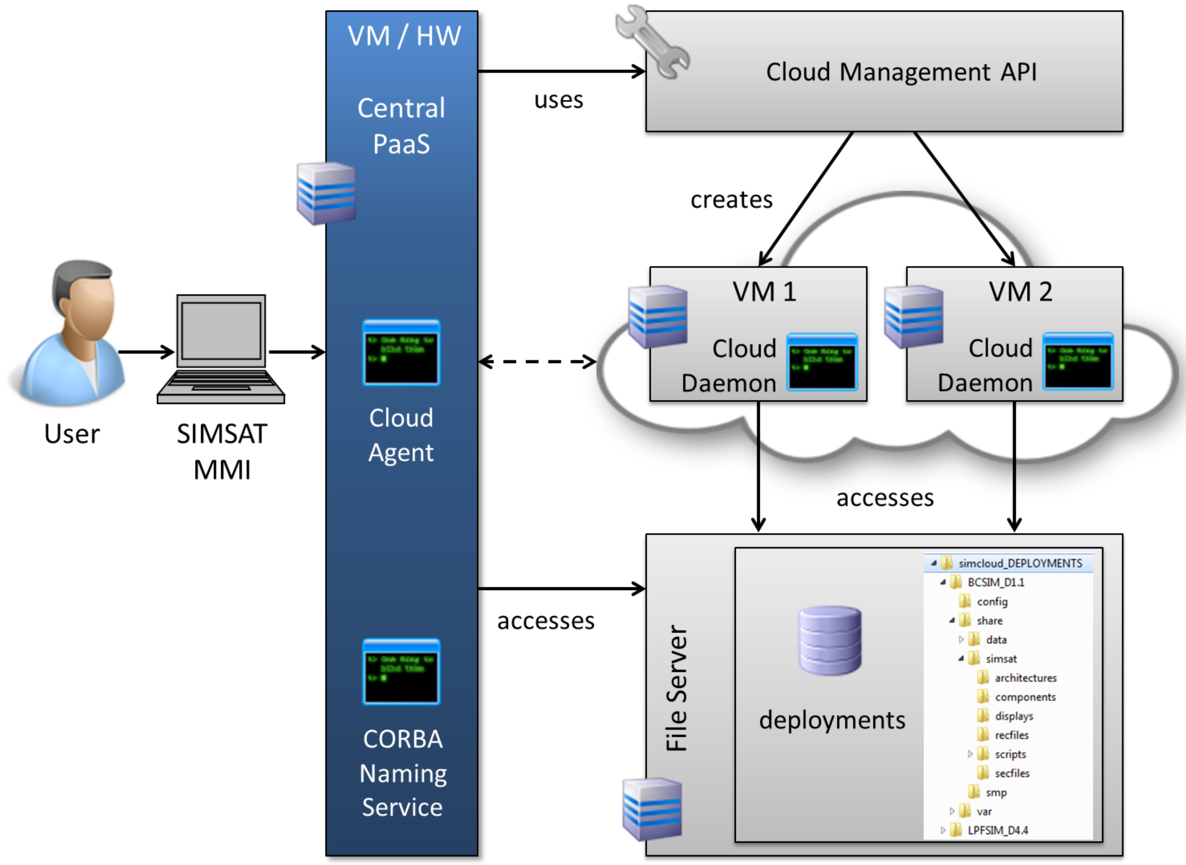


**Figure 3: Cloud Based Provisioning Concept**

It is responsible for managing a number of Cloud Daemons and acts as a master daemon while a Cloud Daemon is an advanced SIMSAT Daemon. There are two subsets of Cloud Daemons: shared and elastic daemons. Differentiation between those two types of daemons allows for a more reliable and predictable environment: *Elastic daemons* are started (and stopped) on demand for a particular simulator and provide the simulator expert a dedicated environment without interference of other simulators. On the other hand the *shared daemons* are never stopped and can be used by multiple simulators at the same time.

A minimal cloud based setup consists of a SIMSAT Cloud Agent started manually by the cloud operator on a central PaaS server and a shared Cloud Daemon that is started on an arbitrary machine which is connected to the central PaaS server. This daemon is required and will never be stopped by the Agent as it provides for example the functionality to retrieve the architecture files in the deployments folder and the required resources for the architectures. The required resources (number of CPUs and RAM) are denoted in a separate file deployed together with the architecture file and need to be configured by the simulator developers.

The Cloud Agent uses a *Cloud Management API* to possibly create virtual machines each running a Cloud Daemon. The Agent keeps track of all the Cloud Daemons that it started and requests additional information from them. This includes the number of (virtual) CPUs and RAM still available on the machine that the daemon is running on. Thereby when a simulation is about to be started the Agent can decide according to the resource availability information whether a new virtual machine is required. This is the case if no shared daemon reports enough free resources – elastic daemons are not considered since they are bound to a specific simulator mission. Since the Agent is the only instance controlling the Cloud Daemons the simulator user does not need to manually start any daemons anymore. Similarly the user is not confronted with the selection of a SIMSAT Daemon but rather only needs to select which simulator to start.

## The Cloud Management API

The SIMSAT Cloud Agent manages Cloud Daemons on multiple virtual machines (VMs). These VMs are provisioned via the *VMware vCloud API* [1] that allows controlling the virtual datacentre (vDC) that hosts the virtual machines in the *ESA Cloud*. A *VMware SDK for Java* [2] hides the details of REST and HTTP which are used by the API to command the virtual datacentre. Since SIMSAT is mainly written in C++ a separate Java program is used to manage the virtual machines. This API represents an additional layer separate from SIMSAT which has the advantage that the API

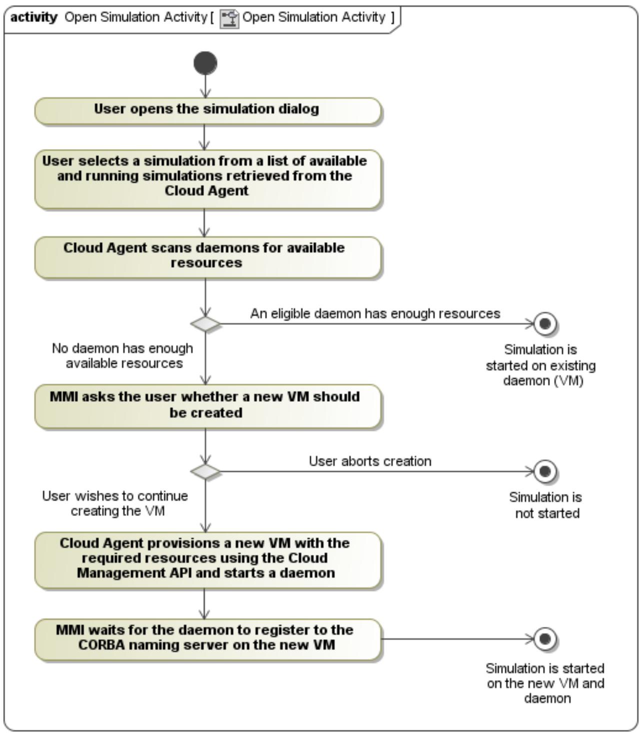
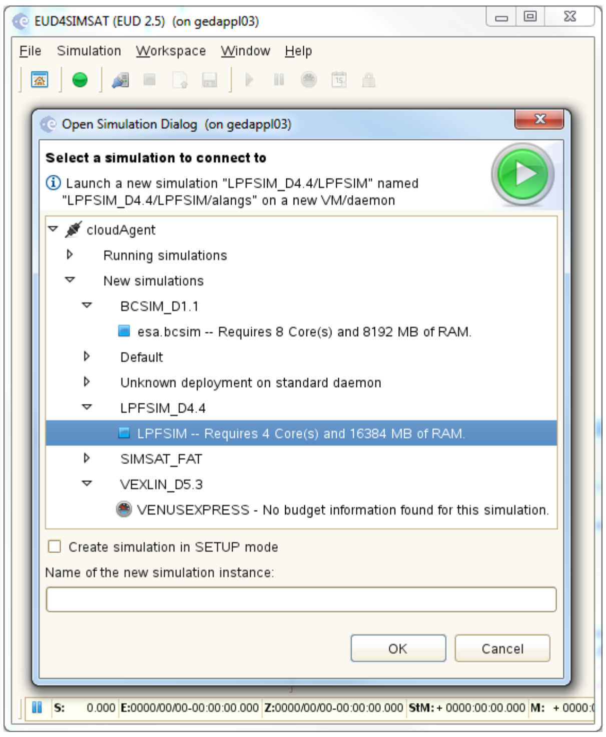

activity Open Simulation Activity [ Open Simulation Activity ]
User opens the simulation dialog
User selects a simulation from a list of available and running simulations retrieved from the Cloud Agent
Cloud Agent scans daemons for available resources
An eligible daemon has enough resources
Simulation is started on existing daemon (VM)
No daemon has enough available resources
MMI asks the user whether a new VM should be created
User aborts creation
Simulation is not started
User wishes to continue creating the VM
Cloud Agent provisions a new VM with the required resources using the Cloud Management API and starts a daemon
MMI waits for the daemon to register to the CORBA naming server on the new VM
Simulation is started on the new VM and daemon

**Figure 4: Flow Diagram When Starting a Simulation**


EUD4SIMSAT (EUD 2.5) (on gedappl03)
File Simulation Workspace Window Help
Open Simulation Dialog (on gedappl03)
OK
Cancel

Select a simulation to connect to

Launch a new simulation "LPFSIM_D4.4/LPFSIM" named "LPFSIM_D4.4/LPFSIM/alangs" on a new VM/daemon

cloudAgent
Running simulations
New simulations
BCSIM_D1.1
esa.bcsim -- Requires 8 Core(s) and 8192 MB of RAM.
Default
Unknown deployment on standard daemon
LPFSIM_D4.4
LPFSIM -- Requires 4 Core(s) and 16384 MB of RAM.
SIMSAT_FAT
VEXLIN_D5.3
VENUSEXPRESS - No budget information found for this simulation.

Create simulation in SETUP mode

Name of the new simulation instance:

S: 0.000 E:0000/00/00-00:00:00.000 Z:0000/00/00-00:00:00.000 StM: + 0000:00:00.000 M: + 0000:

**Figure 5: New "Open Simulation" Dialog**

implementation may easily be replaced later on without impacting SIMSAT, for example in case the virtual datacentre provider is exchanged by a different one.

Once the Cloud Agent detects the necessity to create a new VM it interfaces with the vCloud API via a Java layer. First it needs to login to the vDC, then it fetches a predefined template to create the VM from. The template contains a VM which is already configured to use for example an appropriate network and storage type. The number of virtual CPUs and RAM however are passed from the Cloud Agent to the vCloud API and vary according to the simulators to be started. The cloud management API finally powers on the VM once its deployment in the cloud is completed and the operating system on the VM has finished booting up. This is checked by a ping mechanism that is also robust enough to detect the network stability by reporting a successful connection only once the ping is stable for a predefined amount of time. It then sends the IP address of the newly created VM back to the Cloud Agent. Before the Cloud Agent can start the Cloud Daemon on the VM, the central PaaS server automatically gets configured to accept the SSH key of the new VM in order to facilitate further communication via SSH and the VM is commanded to mount a shared drive hosting the SIMSAT PaaS. Once these post-deployment activities have completed successfully the Cloud Agent finally starts the SIMSAT Cloud Daemon on the VM via SSH.

Once the Agent detects that no simulator is running on a VM anymore and a certain timeout is reached to prevent premature termination, the Cloud Daemon is gracefully stopped on the VM and the VM is shut down and deleted from the virtual datacentre via the vCloud API. Thus the available resources of the vDC are treated with care and freed up for alternate usage.

## USER INTERFACE

The EUD based user interface to start, stop and command simulations in SIMSAT can be run on any machine that has access to the same CORBA Naming Service that is used by the Cloud Agent and Daemons – this can be configured in the EUD MMI. The modified EUD MMI for SIMCLOUD can work with both “normal” SIMSAT daemons and the Cloud Agent. In case it detects a Cloud Agent in the CORBA Naming Service it changes the behaviour of the “Open simulation” dialog to reflect the cloud based environment.

First the EUD MMI requests a list of all available and running simulators from the Cloud Agent. The list is filtered according to permissions that can be configured per simulator to allow or disallow usage of a simulator by user name. In addition the required resources (number of CPUs and RAM) for simulations are also retrieved and displayed to the user. Instead of presenting the user a tree of SIMSAT daemons that are available to run a simulation the user only sees the relevant simulations with no reference to a particular daemon.
Once the user selects a simulation the Cloud Agent scans all Cloud Daemons until it finds one with sufficient free resources and reserves the required resources to avoid race conditions in reservation calls from multiple MMIs. The

daemon reservation mechanism favours those daemons that have been created previously by the same user (so-called elastic daemons) over other shared daemons. If none of the available daemons has enough resources left the user is informed that a new VM will be created. The user may still abort this process in order for example to close another simulation before trying to start a simulation again. In both cases – whether a new VM is created or not – the simulation is started with the only notable difference being the duration until the simulation is ready as VM provisioning takes additional time. If the user aborts the load dialog, any resources that were previously reserved for the user are freed up for other users.

Commanding of the simulator remains unchanged though all file operations need to be done in relation to the remote SimHost. The EUD MMI already provides a means to browse the remote file system, therefore the simulator user can mostly use the MMI as usual.

Once the simulator user closes the simulation the corresponding daemon and VM are scheduled for deletion unless the user starts another simulation again during a predefined grace period. In this case the VM is reused automatically allowing for rapid turn-around time.

## PERFORMANCE ANALYSIS

In addition to the cloud concepts described above, a performance assessment of simulators running in a virtualized environment has been executed by S. Ferreri at ESOC [3]. The analyses involve the operational simulators of four missions (namely Bepi Colombo, Gaia, Lisa Pathfinder and SeoSat) and several hardware and software configurations.
The test cases are designed by the simulator developers, the mission Flight Control Team members and the simulator technical officers. This approach provides the highest coverage possible in terms of tested scenarios.

### Hardware Specifications, Software and Configuration Details

All the tested operational simulators are installed on machines equipped with the ESOC baseline: SUSE Linux Enterprise Server (x86_64), version 11, service pack 1; Kernel version 2.6.32.59-0.7-default.
The results of the performed tests are very similar among the simulators of the different missions. Therefore, in this paper, we only present the results of the Bepi Colombo simulator which was chosen as the most representative system.
Table 1 summarizes the main system specifications of the machines involved in the tests.

**Table 1: Hardware specifications of tested systems**

| # | Machine name | Machine type | CPUs | Memory |
|---|---|---|---|---|
| 1 | bsimp | Physical G8 | 2 CPU Sockets<br>Intel Xeon E5-2643 3.30GHz<br>4 cores per socket, 2 threads per core | 64 GB |
| 2 | bsimv | Virtualized G8 | 2 CPU Sockets<br>Intel Xeon E5-2637 v2 3.50GHz<br>8 cores per socket, 1 thread per core | 64 GB |
| 3 | bsimvCloud | Virtual machine on the ESA Cloud | 2 vCPUs on a blade chassis based on Intel Xeon CPU E5-2680 2.70GHz<br>16 cores per vCPU, 1 thread per core | 64 GB |

The tested machines have different CPU models. In order to make the comparison meaningful, similar clock speeds and the same number of computing units is used.
The number of processing units available (i.e. 16) is higher than the number needed (i.e. 7 for the heaviest test).
The machine used as basis for performance comparison is bsimp (see #1 in Table 1).
The operational simulator version used for the tests is BCSIM D2.1.0 based on the infrastructure suite SIMULUS 5.2 Final Acceptance.

All performed tests focus on assessing the simulator speed in different scenarios. This can be achieved by letting the simulator run in Free Running mode (i.e. as fast as possible) without the occurrence of Real Time Slips (i.e. delays in the execution of the scheduled events).
The tested scenarios are listed in Table 2 and the detailed description of the test cases can be found in [3].

**Table 2: Test case scenarios**

| Test Case | Test Description |
|---|---|
| TC1 | Maximum simulator speed factor when in nominal operational configuration |
| TC2 | Elapsed time for generation of set of operational configurations |
| TC3 | Simulator performance when multiple (up to 6) emulators are running in parallel |
| TC4 | Simulator performance with high logging rate (i.e. 1MB/s) |

| | |
|---|---|
| TC5 | Simulator performance under heavy load: 2 Ground Station models, max. X-band and Ka-Band downlink rate, AOCS SHM mode, payload models in Science mode, high TC injection rate |
| TC6 | As TC5, but without payload models |

**Test Results**

The test results are relative with respect to the reference machine (bsimp), and are summarized in Table 3.

**Table 3: Test results overview**

| Test Case | System#2 (bsimv) | System#3 (bsimvCloud) |
|---|---|---|
| TC1 | 95.3% slower than system#1 (bsimp) | 3.5% faster than system#1 (bsimp) |
| TC2 | 96.4% slower than system#1 (bsimp) | 4.0% faster than system#1 (bsimp) |
| TC3 | 105.0% slower than system#1 (bsimp) | 10.5% faster than system#1 (bsimp) |
| TC4 | 95.4% slower than system#1 (bsimp) | 7.0% faster than system#1 (bsimp) |
| TC5 | 99.8% slower than system#1 (bsimp) | 4.8% faster than system#1 (bsimp) |
| TC6 | 107.5% slower than system#1 (bsimp) | 4.2% faster than system#1 (bsimp) |

The percentage indicates how much time the test case took to run relative to the reference system and normalized to its execution time, e.g. (exec_time_bsimv-exec_time_bsimp)/(exec_time_bsimp/100) therefore, if TC6 took 100 seconds to execute on bsimp, it took 207.5 seconds on bsimv.
The performance results can be summarized as follows:

1. The virtual machine running on vCloud (system #3) provides the best performance, around 5% faster than the physical machine, on average
2. The virtualized G8 machine (system #2) doesn't provide operationally acceptable performance since, under certain loads, it is not able to run the simulator in real time. The reasons of this low performance are explained in the next section.

**Machine Performance Results**

Together with the simulator performance, the machine CPU utilization and I/O performance are analysed, with a particular focus in understanding the reasons of the low performance of the virtualized G8 machine (system #2).
A direct comparison between system #1 and #2 shows that the CPU utilization profile is very similar between the physical (Figure 6) and the virtualized (Figure 7) environment.

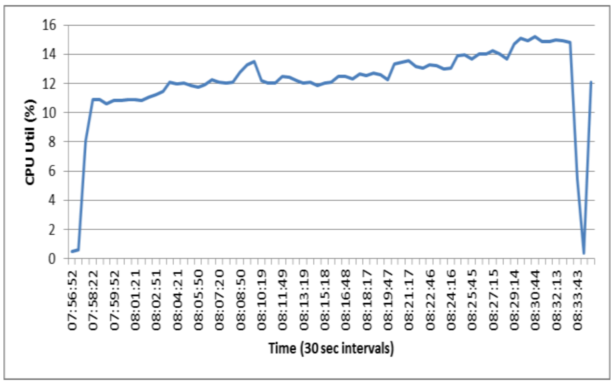


**Figure 6: CPU utilization, TC3, bsimp**

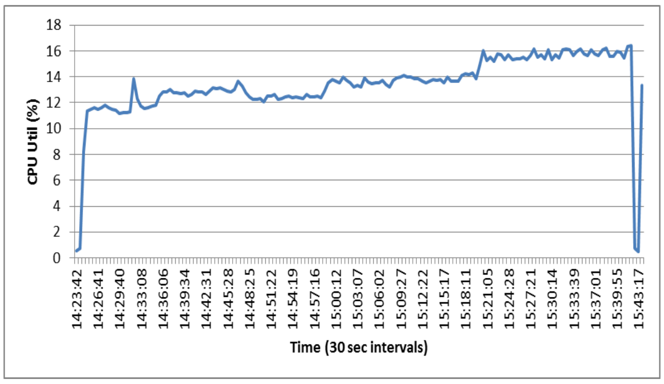


**Figure 7: CPU utilization, TC3, bsimv**

There are major differences in the disk I/O performances: in particular the I/O rate of the virtualized machine (system#2) is between 3 to 7 times lower than the one of the physical machine (system #1), see Figure 8 and Figure 9.

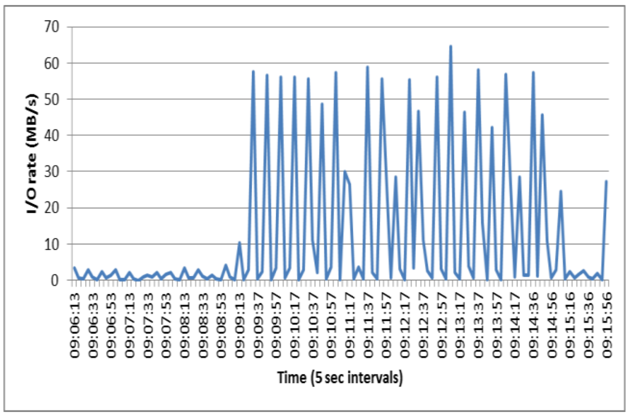


**Figure 8: I/O rate, TC4, bsimp**

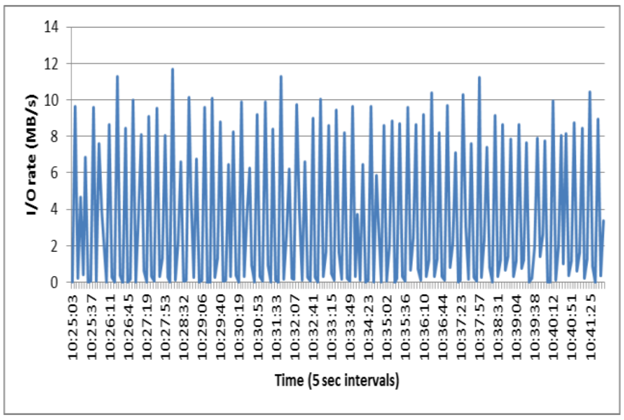


**Figure 9: I/O rate. TC4, bsimv**

The machine performance results show that the biggest problem in using virtualized environments is the I/O rate. This factor is particularly critical when synchronous I/O is used, i.e. typical usage for ESOC operational simulators.
The virtual machine on the vCloud is not affected by this loss thanks to the high performance storage hardware used.

## CONCLUSION AND FUTURE WORK

In this paper we presented the SIMCLOUD approach to run operational simulators in the cloud by implementing a *Platform-as-a-Service* and *Software-as-a-Service*. The SIMSAT PaaS allows to dynamically provision VMs based on required and available hardware resources. The SIMULUS suite of models is adjusted to allow parallel deployment of multiple versions for multiple users and simulator missions. The SIMCLOUD concept eases the runtime environment for simulator users (such as FCT members) by taking over administrative tasks such as the setup of machines with baseline software or management of SIMULUS installations. In addition we have analysed the virtualization with regards to potential performance impacts where the initial results indicate that virtualization can be a valuable alternative to physical machines if using appropriate hardware, where performance is highly depending on the choice and set-up of the virtualization layer.
The solution presented in this paper consists of changes to the architecture, design and code of multiple SIMULUS components which are not yet part of the SIMULUS product. The deployment concept is the first change that is currently under investigation for implementation in the SIMSAT product to allow deployment of multiple missions in one installation.
In addition a web interface to deploy multiple missions has been developed that interacts with the PaaS and eases the management of simulators on the shared drive thus removing the need for the user to directly access the storage drive. This also allows automatically validating and tracking the deployments before they are made available to the simulator users decreasing the possibility of faulty or unintentionally modified simulators.
In conclusion making use of cloud concepts in the domain of operational simulators requires some architectural changes to the underlying software but benefits both the end user by simplifying interaction with the runtime environment and the administrators that can rely on the elasticity of the system and gentle usage of resources.

## REFERENCES

[1] VMware, "VMware vCloud REST API Reference Documentation", https://www.vmware.com/support/vcd/doc/rest-api-doc-1.5-html/, Revision 20110811, Version 1.5, last access February 2015
[2] VMware, "VMware vCloud SDK for Java for vCloud Suite 5.5", https://developercenter.vmware.com/web/sdk/5.5.0/vcloud-java, last access February 2015
[3] S. Ferreri, "BepiColombo Simulator performance comparison (physical vs. virtual machines) tests report", BC-ESC-RP-21006 i1r0, 02/06/2014.