MIKELANGELO

D2.2
Intermediate Use Cases Implementation Strategy

Workpackage	2	Use Case & Architecture Analysis	
Author(s)	Uwe Schilling		USTUTT
	Matej Andrejašič		PIPISTREL
	Gregor Berginc		XLAB
	Benoît Canet		SCYLLA
	Peter Chronz		GWDG
Reviewer	John Kennedy		INTEL
Reviewer	Bastian Koller		USTUTT
Dissemination Level	PU		

Date	Author	Comments	Version	Status
19 April 2016	Uwe Schilling	Initial draft / Initial structure	V0.0	Draft
30 May 2016	Matej Andrejašič, Gregor Berginc, Uwe Schilling, Benoît Canet	Added use case content	V0.1	Draft
2 June 2016	Uwe Schilling	Added executive summary / Introduction	V0.2	Draft
20 June 2016	Gregor Berginc, Uwe Schilling	Extending executive summary and introduction; overview of use cases	V1.0	Review

© Copyright Beneficiaries of the MIKELANGELO Project

		and their plans. Reordering of the use cases.		
29 June 2016	Gregor Berginc, Uwe Schilling	Document ready for submission	V2.0	Final

© Copyright Beneficiaries of the MIKELANGELO Project

Executive Summary

This is the second iteration of a deliverable describing the four business use cases of the MIKELANGELO project. Contrary to the four separate reports released in August 2015, this encompases status, new findings, requirements and implementation strategies for all of them. The purpose of these use cases is to allow efficient validation and evaluation of the whole MIKELANGELO technology stack, from as many perspectives as possible.

The cancellous bones use case is a typical HPC application providing requirements for performance improvements of various I/O subsystems used in virtual environments. Contrary to this, the cloud bursting use case drives the work towards highly optimised cloud application stacks, in particular focusing on improvements in the guest operating system: the novel application programming APIs fully exploit modern processor and memory architectures and are the cloud operation system of MIKELANGELO. The big data use case extends this and opts for an integration of high performance virtualised environments with the flexibility offered by the cloud. Finally, the aerodynamic maps use case supplies two variants suitable for a comprehensive evaluation of a fully integrated stack for HPC and cloud applications.

All these use cases are based on real-world applications. This allows us to constantly monitor and update the list of high-level requirements. It is essential that requirements are refined throughout the entire lifecycle of the project, because advancements in underlying components open new ideas, as well as issues. Furthermore, these use cases ensure usability, integration and other important practicalities are considered by the partners responsible for the various implementations. Finally, the nature of the use cases carries tremendous potential to support our dissemination and exploitation efforts.

The use cases rely on different improvements of the components developed in technical work packages, some of which are still pending for the realisation. To this end they are currently in different development and integration stages.

This document encompases thorough descriptions of use cases and the datasets they rely upon. Current implementation and integration status is elaborated, as well as new requirements. The requirements address individual components and the HPC/Cloud stack integrations. Finally, future implementation plans are given where applicable.

© Copyright Beneficiaries of the MIKELANGELO Project

Acknowledgement

The work described in this document has been conducted within the Research & Innovation action MIKELANGELO (project no. 645402), started in January 2015, and co-funded by the European Commission under the Information and Communication Technologies (ICT) theme of the H2020 framework programme (H2020-ICT-07-2014: Advanced Cloud Infrastructures and Services)

© Copyright Beneficiaries of the MIKELANGELO Project

Table of contents

1 Introduction 9
1.1 MIKELANGELO Workflow 10
1.2 Evaluation of the Key Performance Indicators 11
1.3 Roadmap 15
2 Cancellous Bone Simulation Use Case 18
2.1 Dataset 18
2.2 Implementation 19
2.3 Evaluation and Validation 21
2.4 Use Case Requirements 23
2.5 Roadmap 24
3 Cloud Bursting Use Case 25
3.1 Dataset: Cassandra-Stress Database Schema 25
3.2 Implementation 26
3.3 Evaluation and Validation 28
3.4 Use Case Requirements 29
4 Virtualized Big Data Use Case 30
4.1 Datasets 30
4.2 Implementation 31
4.3 Evaluation and Validation 33
4.4 Use Case Requirements 34
4.5 Roadmap 34
5 Aerodynamic Maps Use Case 36
5.1 Introduction 36
5.2 Dataset 37
5.3 Implementation 39
5.3.1 OpenFOAM Cloud 40
5.3.2 OpenFOAM HPC 42
5.4 Use Case Requirements 43
5.4.1 End-user Requirements 44

© Copyright Beneficiaries of the MIKELANGELO Project

5.4.2 OSv Requirements 45
5.4.3 Package Management Requirements 45
5.4.4 The Snap Requirements 45
5.4.5 HPC Environment Requirements 46
5.5 Roadmap 46
6 Concluding Remarks 49
7 References and Applicable Documents 50

© Copyright Beneficiaries of the MIKELANGELO Project

Table of Figures

Figure 1. Development workflow of the MIKELANGELO project. 11
Figure 2. Time plan for the implementation, integration and evaluation of the Cancellous Bone Simulation (CBS) Use Case. 24
Figure 3. An example of a drop in number of operations when scaling ScyllaDB 1.1 cluster. 28
Figure 4. Example of a reduced drop when scaling Scylla 1.3 cluster. 29
Figure 5. Big data roadmap. 35
Figure 6. A section of a log file of a typical OpenFOAM simulation indicating two time iterations of a simulation and an intermediate forces calculation. 38
Figure 7. Beginning of a forces.dat file containing forces and moments calculated on chosen aerodynamic surfaces. 39
Figure 8. Architecture diagram of the OpenFOAM Cloud application. 40
Figure 9. Simulation data of four OpenFOAM cases run simultaneously, collected by snap and visualised in Grafana application. 42
Figure 10. OpenFOAM Cloud as an OpenStack dashboard. 42
Figure 11. Time plan for the implementation, integration and evaluation of the Aerodynamics use case. 48

© Copyright Beneficiaries of the MIKELANGELO Project

Table of Tables

Table 1. Evaluation of Key Performance Indicators from the perspective of use cases.....11

© Copyright Beneficiaries of the MIKELANGELO Project

1 Introduction

To ensure complete coverage of all aspects of the MIKELANGELO technology stack [1], four business use cases are used to test and evaluate the development. The use cases were carefully chosen early in the proposal preparation in order to assess technical progress of the MIKELANGELO progress from as many perspectives as possible. Since the beginning of the project the use cases have been analysed and evolved to show different advancements achieved by MIKELANGELO project for Cloud and HPC, advancements in I/O efficiency of VMs, the workflow in HPC-Clouds and cloud-like infrastructures, and the scalability of those infrastructures. Parts of the use cases have been adapted to be able to run on the new operating system OSv, others have focused on a full stack integration. They will be described in their corresponding sections, in more detail.

The cancellous bones application represents a workload traditionally executed on top of high performing infrastructure offered by an HPC cluster. The application can be parallelised as much as necessary due to the fact that the entire problem may be decomposed into a series of smaller ones. With the increased parallelism, high emphasis is put on communication components (e.g. network cards, InfiniBand cards). Efficient virtualisation of these hardware components and dynamic management of CPU core allocations will decrease the performance penalties of the I/O subsystem introduced by virtualisation. Furthermore, this use case is one of the pillars for the HPC integration work mandatory to allow transition of current workload management into new completely virtualised environments.

The cloud bursting use case on the other hand focuses on a component typically used in modern applications running in the cloud. We have explained in deliverable D2.4 that rapid increase of the number of requests can deteriorate use experience of any application, in particular those that are highly temporal, for example news or weather sites. A particular NoSQL database is chosen (Cassandra) to demonstrate the limitations of modern cloud environments as well as the guest operating systems. The use case is thus primarily focusing the evaluation of the guest operating system improvements along with the demonstration of the effects modern APIs considering advances in hardware development have on real world applications. Cloud integration will also facilitate the evaluation of all other components of the MIKELANGELO project.

The big data application offers a bridge between HPC and cloud. It is estimated that big data applications are going to be the main driving force for the advances in HPC. Besides sheer power, big data crucially requires flexibility of the cloud, allowing composition of applications and complete workloads on the fly. The big data use case consists of several synthetic benchmarks used for low level evaluation of MIKELANGELO. At least one real-world

© Copyright Beneficiaries of the MIKELANGELO Project

application is planned for integration in order to allow for reasonable evaluation from an end-user's perspective.

Finally, the aerodynamics map use case is used for comprehensive evaluation of the entire MIKELANGELO project. The use case has been designed specifically to support cloud and HPC environments separately. While the former targets evaluation of flexibility and manageability, along with the low-level performance improvements, the latter is mostly focusing on the overarching performance improvements. Being the use case supported in both environments, it was the first use case the consortium members have contributed to, first by enabling full support in OSv and workload parallelism, and then continuing towards integration with HPC and cloud.

The purpose of this deliverable is to present the current status of all of these use cases, list and elaborate their requirements, and outline provisional implementation strategy until the end of the project. Requirements are given both for the underlying components of MIKELANGELO project as well as for the implementation of the use cases themselves. The deliverable is structured into four main sections, one for each use case. A brief introduction is given for each of the use cases. This primarily summarises our previous work, however it also includes additional findings made since the previous use case specific report. In order to fully understand the use case, a detailed presentation of used datasets is used. This not only outlines the size of the used data, but also introduces the underlying structure. The current implementation and a brief evaluation is described next. A more detailed evaluation has been the subject of work package 6 and is reported in the corresponding deliverable (D6.1). Finally the use cases elaborate on the current requirements, and outline high-level plans.

Before we begin with the use cases, the following subsections put this report into context of the MIKELANGELO project.

1.1 MIKELANGELO Workflow

Business use cases are serving two primary purposes for the MIKELANGELO project. Being the core part of work package 2 (Use Case & Architecture Analysis) they provide sensible end user requirements for the technical work packages, namely WP3 for the hypervisor improvements, WP4 for improvements to the guest operating system and WP5 for the integration into complete HPC and cloud stacks. On the other hand, the implementation and integration of these use cases using MIKELANGELO technology delivers concrete validation and evaluation (WP6) steering new iteration of design, implementation, integration and evaluation loop (Figure 1).

© Copyright Beneficiaries of the MIKELANGELO Project

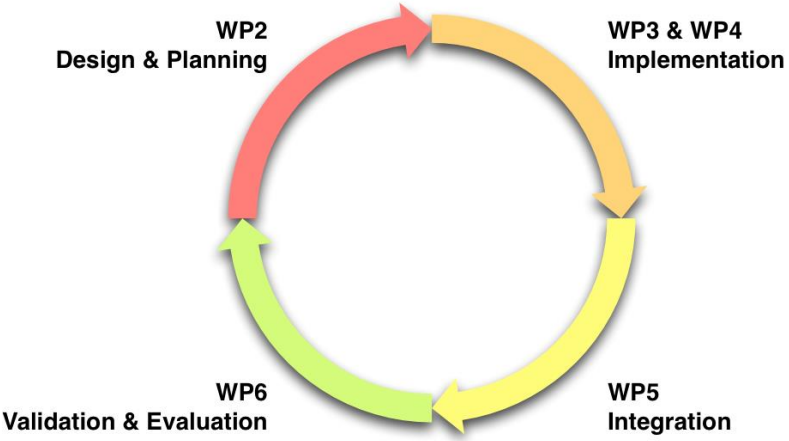

Figure 1. Development workflow of the MIKELANGELO project.

1.2 Evaluation of the Key Performance Indicators

A number of key performance indicators (KPIs) have been designed to facilitate the validation and evaluation of the advancements of the project. Their explanation along with the current evaluation in terms of the four presented use cases is given in the following table.

Table 1. Evaluation of Key Performance Indicators from the perspective of use cases.

KPI	Description	Current assessment
KPI1.1: relative efficiency of bursting a number of virtual machines	KPI relates to the cloud bursting use case that will directly exploit this feature of the enhanced technologies. Other use cases (big data and HPC) are going to benefit from the improved bursting because the time required to deploy virtual infrastructure will be shortened.	Replacing Linux with OSv shortens the overall time required to distribute and launch required virtual infrastructure. Work on the new API (Seastar) shows promising results that are going to be demonstrated on the cloud bursting use case.
KPI1.2: relative improvement of time to change the installed	Managing large cluster of applications is daunting task. The lightweight nature of the cloud	OpenFOAM Cloud application already exploits this

© Copyright Beneficiaries of the MIKELANGELO Project

KPI	Description	Current assessment
application on a number of virtual machines	operating system whose management layer is provided through a standard REST API, it is estimated that management will be simplified.	functionality allowing application management layer to contextualise applications on the fly.
KPI2.1: relative efficiency of virtualized I/O between KVM and sKVM (developed in the project)	KPI tackles on of the last remaining obstacles of virtual environments - slow I/O operations compared to native. All four use cases are going to benefit from this KPI.	Synthetic benchmark already demonstrate the value of this KPI. Although integrated in cloud and HPC, use cases have not performed a thorough analysis specifically for this KPI.
KPI3.1: The relative improvement of efficiency of MIKELANGELO OSv over the traditional guest OS.	Traditional guest OS in context of this KPI is general purpose Linux OS. It is estimated that the benefits of using a lightweight unikernel are going to will be higher than the cost of ensuring applications run on OSv on the long run.	Aerodynamics use case, and partially the big data use case, have already succeeded in supporting the application in OSv. No concrete gains in performance have been noticed to this point, apart from the significantly smaller size of images resulting in lower cost of deployment of VMs.
KPI3.2: The relative improvement of efficiency [size, speed of execution] between the	Use cases provide new requirements for the OSv operating system from the perspective of performance.	Although it was possible to provide several requirements for OSv directly (for

© Copyright Beneficiaries of the MIKELANGELO Project

KPI	Description	Current assessment
Baseline Guest OS vs. the MIKELANGELO OSv.		example, support for *multi-process* execution, NFS, ...), most of them have to be gathered with active collaboration between use case owners and OSv team. Using an iterative approach it was possible to specify as well as resolve several performance issues
KPI3.3: The relative improvement of compatibility between baseline and MIKELANGELO versions of OSv.	Similar to KPI3.2, but from the perspective of compatibility,	Consortium approaches compatibility improvements in a similar way to the performance improvements, i.e. with active collaboration between the use cases and OSv team.
KPI4.1: A system for appropriately packaging of applications in MIKELANGELO OSv is provided.	Three use cases are based on commonly used applications. A tool facilitating seamless packaging of these apps is mandatory from the perspective of these use cases as well as dissemination and exploitation activities.	The application packaging tool has already seen several improvements allowing some of the use cases to provide required packages. Integration was also demonstrated in OpenFOAM Cloud application.

© Copyright Beneficiaries of the MIKELANGELO Project

KPI	Description	Current assessment
KPI4.2: A relative improvement of efficiency [time] between deploying a packaged and non- packaged application.	KPI targets efficient composition of application packages and target VMs.	Deliverable D6.1 presents initial evaluation of the performance of the application packaging tool in different setups. For specific use cases, improvements have already been shown.
KPI5.1: Demonstration of a management infrastructure on diverse physical infrastructures.	Most of the use cases are suitable for execution in cloud and HPC environments. They will allow thorough evaluation of benefits of either one resulting in several best practices.	Use cases have already been tested in the two environments. HPC environment already supports the virtualisation of the cancellous bones use case and various big data workloads have been tested in the cloud infrastructure.
KPI5.2: Relative efficiency [time, CPU, disk overheads] of traditional HPC over Cloud HPC offered in MIKELANGELO.	HPC use cases will be used to evaluate the performance.	Initial evaluations done for cancellous bones and aerodynamics use case. Plan envisages continuous evaluation based on the newly implemented and integrated enhancements of the underlying components.
KPI6.1: Enumerated	This KPI remains to be defined in	Evaluation on the

© Copyright Beneficiaries of the MIKELANGELO Project

KPI	Description	Current assessment
security concerns solved on hypervisor level.	context of use cases. Currently, the main focus in on a synthetic use case showing enhancement of the security modules for a range of applications dealing with encryption.	synthetic demonstrator shows enhanced security.
KPI7.1: All use cases appropriately demonstrated	Final goal of the use case implementation and integration.	Currently, all use cases have been demonstrated in one form or the other. Plan is to ensure use cases evaluate as many different aspect of the MIKELANGELO project as possible.
KPI7.2: Documentation of using MIKELANGELO in Use cases - Best Practices tutorial.	Documentation is an important artefact that will support demonstration of results as well as facilitate the uptake of the MIKELANGELO technology.	Documentation of the use cases is currently only partial, depending on their current status.
KPI7.3: Documentation of using MIKELANGELO in Use cases - Documented Benefits.	An overarching documentation of best practices, thorough evaluation, benefits and caveats - all of this using the four business cases.	This KPI is going to be addressed in the future, once more detailed evaluation is possible and the different components are integrated.

1.3 Roadmap

The MIKELANGELO project has reached its halfway point and has already provided several improvements over the baselines (presented in this document as well as the accompanying evaluation deliverable D6.1). The current milestone (MS3) allowed the use cases to conduct the first batch of validations and evaluations from the perspective of target applications,

© Copyright Beneficiaries of the MIKELANGELO Project

rather than just synthetic benchmarks. As the various individual components are developed independently, and different combinations of them are employed by the different use cases, maturity levels of the use cases currently differ:

Cancellous bones use case focuses on the integration of existing environments based on Linux with the overarching HPC infrastructure and workload management tools.

Cloud bursting use case demonstrates individual improvements of the underlying components leading towards a fully assembled use case instrumenting the cluster as required.

Big data use case revolves around integration of low level improvements in the hypervisor and evaluation of the lightweight unikernel for widely used big data tools.

Aerodynamic map use case targets the HPC compatibility improvements on one hand and flexibility of application management on the other hand, both integrated with designed architectures.

With the initial integrated releases of the two stacks, the remaining four milestones, will focus on the finalisation of the implementation, integration and evaluation of business use cases.

M24 (December 2016) - MS4 - 3rd Platform Prototype. All four use cases will implement the initial prototypes suitable for validation and evaluation. Implementation will ensure applications are properly supported in the OSv unikernel in terms of compatibility improvements. Use cases will furthermore be adapted to exploit new modifications in the hypervisor (in particular IOcm and virtual RDMA). Several new requirements have already been contributed by use cases, however new ones are expected to be identified and resolved during this phase.

M30 (June 2017) - MS5 - Final Architecture. Use cases will be employed to finalise the architecture of individual components as well as the (unified) stacks for HPC and cloud environments. Use cases will be mostly finalised at this point, integrating the advanced features of the MIKELANGELO technology, in particular the ability to empower cross-level optimisations. Besides the source code release of use cases, this milestone is also going to offer deep evaluation of all the different components, separately and integrated. Continuous evaluation is already going to be used where applicable.

M33 (September 2017) - MS6 - Feature Freeze. Continuous validation and evaluation will continue to support technical work packages towards final versions of all components. No more major changes to the use cases are expected at this point as the goal of this phase is to improve identified bottlenecks.

© Copyright Beneficiaries of the MIKELANGELO Project

M36 (December 2017) - MS7 - Final Release. At the end of the project, final versions of all four use cases will be released. The release will include source code (where applicable) and detailed documentation consisting of its purpose, installation procedures and instructions on how to use it. Results of validation and evaluation will also be released as part of the open data pilot. In accordance with the KPI 7.3 all the know-how and best practices will be aggregated allowing broader uptake of the results of the MIKELANGELO project.

Use cases provide more detailed roadmaps where applicable in their corresponding sections.

© Copyright Beneficiaries of the MIKELANGELO Project

2 Cancellous Bone Simulation Use Case

In this section, an update on the cancellous bones simulation is given, as well as more insights into the simulation. The current status of the implementation and the strategy for the next period as well as the requirements for the use case will conclude this part.

The purpose of this simulation is to understand the structure of the cancellous, or spongy, parts of the human bone, in order to efficiently and safely attach required implants to it. Using this simulation, implants can be produced in a more accurate fashion, which increases production yield and at the same time improves the healing time for bone implant surgeries as well as the lifespan of the implant.

2.1 Dataset

One big challenge of HPC applications is scalability. To write an application that can make use of more than 10,000 cores is not a straightforward task. The scalability of an application depends on its internal limitations. Some applications are CPU bound, which scale relatively easily, whilst some are I/O bound. I/O bound applications scale to the point where more CPUs are waiting for input data, than CPUs that are computing. The cancellous bone simulation is, to a certain extent, I/O bound.

To get data to the CPUs quickly, it helps to store the data in an efficient way. The cancellous bone simulation uses its own file format (called *PureDat*) to scale better. The basic idea is to have a simple persistent storage of computed data on the file system. It should store data space efficiently (fewer data transfer) and reduce the amount of single files (fewer file read/writes). This all helps the simulation to handle I/O operations better.

A simple file format would write all information to one coordinate node (position, number and entry) in one entry. This would lead to a context switch each time a new data value has to be written or read from storage. To avoid this level of churn, all values of the same kind are stored in a common file. All data type gets his own file. This reduces the library calls that would otherwise be required to convert them from one format to another one. In addition, these files are stored in a binary type reducing the overall size of files.

To tweak the format further, the data itself is not split with metadata information in between. The values are written in a plain way and stored in files with the same kind of datatype, so the knowledge about the space of a single entry is known by the time the file gets opened.

PureDat can handle six data types at the moment but can be expanded if needed:

1. Signed Integer with 1 Byte
2. Signed Integer with 2 Byte

© Copyright Beneficiaries of the MIKELANGELO Project

3. Signed Integer with 4 Byte
4. Signed Integer with 8 Byte
5. Floating Point Values with 8 Byte
6. ASCII Data with 1 Byte

Therefore, six files get stored in the file system, each with a different kind of data. A short example: for the simulation it is important to save and load coordinates (nodes) in an efficient way. Three writes are necessary to store this kind of data:

1. A list of node numbers (Integer)
2. A list of node coordinates (Floating Point)
3. The amount of nodes (Integer)

The third one can be dropped if the list of nodes is in order so the last node number is the amount of nodes. Each of these three writes can be stored in a separate file (which can be written in parallel). If we compare this with a "traditional" way of storing coordinates we can see why the PureDat approach is better. Traditionally the node number (Integer) gets stored followed by the coordinates (Floating Point), which leads to more library calls per node writes or reads. Depending on the lengths of the list this can scale to a few million library calls.

To get single block data out of this data "stream", the data is stored in a tree like structure. This tree can store raw data in different locations. As in other tree like data formats, there are leaves and branches. The information about this tree is stored in a separate file, to have all data fully separated.

This explanation should show that to get I/O bound applications to scale, the internal implementation details have to be known and should be optimized on an application level as much as possible, otherwise the problem stays. All optimisations to I/O will make this application scale further, over more nodes, over more cores and run faster or be able to produce more results in the same time. With the I/O optimisation of MIKELANGELO this will be possible with a cloudlike environment. The overhead of I/O for traditional VMs is too high at the moment to run such HPC application with 10,000 cores in a cloud. But with technologies like MIKELANGELO's dynamic I/O core dedication or vRDMA this is within the bounds of possibility.

2.2 Implementation

This section will explain the current, intermediate, procedure to get the Cancellous Bone Simulation Use Case running in a VM. Ultimately a more elegant way to do this will be realised.

© Copyright Beneficiaries of the MIKELANGELO Project

To run an application in a cloud-like environment, it has to be "containerised" in some fashion. At this point there are three possible approaches towards such an application bundle: Lightweight containers like LXC [2] or docker [3], microkernels like OSv [4], and the classical approach with a standard operating system inside a VM.

For comparison, we have first integrated the simulation into a standard operating system. This will give us a baseline to compare it with OSv integration. To get an overview how to get such an image of a VM, you need to do the following steps:

1. Get an Ubuntu Cloud Image
   This is an Image which is prebuilt by Canonical [5] and can be downloaded. It is most widely used in OpenStack environments, but this works equally well with Torque [6].
2. Define a default XML file
   To Boot a VM with libvirt, a domain XML file is needed. It specifies the virtual hardware from the hypervisor to the virtual machine. This can be done in different ways. One way is to install virt-manager, to get a GUI and some sort of guidance through the process. The other possibility is to read through the domain documentation [7] and build the domain XML by hand. We have done both, first generating with virt-manager a more generic XML and then editing it afterwards to specify exact hardware and software requirements.
3. Generate Metadata
   To get access to the VM it has to be provided with metadata information like user, password and SSH key. This is done via cloud-init [8], which allows additional information to be injected into a VM at boot time
4. Modify the Image
   To get the bones use case up and running, the VM has to be modified. The additional packages needed from the default ubuntu repository are: **Libgfortran3** a library which makes it possible to run fortran based programs. **Mpi-default-bin** this is needed to run MPI programs.
5. Moving the application
   To run the use case it is mandatory to have the binaries of the application inside the Image. It can be anywhere in the image, but we choose to use the root directory (/bones).
6. Finalize the Image
   To get the Image in the final state, it is needed to remove the cloud-init cache. Cloud-init remembers if the Image was started or not, this has to be removed in order to boot properly the next time.
7. Shrink the Image
   To speed up the toque startup process, it is recommended to shrink the Image. This can be done with virt-sparsify [9]. The disc will be filled completely with zeros and

© Copyright Beneficiaries of the MIKELANGELO Project

then compressed. This can shrink an image form 3.4GB to 300MB (a factor of 10). When Torque moves this image to a RAM-disc, this process is speeded-up tremendously.

This transition, from a HPC-Application to a more cloud-like application can be used as a blueprint for porting other applications to a more cloud-like state. As long as the hardware beneath the virtual machine is capable of running the new container it works on every hardware. This leads to a state where an application developer can test their HPC software on a laptop or desktop machine and then transfer it to a cloud/HPC provider like GWDG or USTUTT.

For the porting of bones to OSv similar steps are necessary. At the time of writing this process has not started yet, but will start in the third quarter of this year. With this use case we focused on functional fully-integrated complete stack first, which proofs that this approach work in general.

2.3 Evaluation and Validation

In this section is a short summary of the first measurements of the simulation inside the HPC stack. For a more complete overview can be found in D6.1 [26].

At this stage the use case is fully integrated into the MIKELANGELO HPC stack and can be executed directly on the compute nodes as well as inside a virtual machine. To evaluate the functionality of the use case, it has to be executed in one of the two ways.

Bare Metal execution (directly on the host):

```
qsub -l nodes=1:ppn=16 /home/hpcuschi/projects/mikelangelo-
bones/bones-jobscript-metal.sh
```

Where nodes define the host environment, in this case one node with at least 16 cores (ppn processes per node). And a job script which runs on the first allocated node. The job script handles the data transfer between the home file system and the shared scratch, defines a config file and starts the execution of the simulation.

VM execution:

As explained in the Implementation section above, a VM image is needed to execute the job. The image contains the binaries and libraries needed to successfully run the simulation. The data, the configuration and the job script are not part of this image, this modularity makes it easy to set up different configuration without touching the images at all. This image can be understand as “fat” binary, a single file which contains the smallest set of setup to run the simulation.

© Copyright Beneficiaries of the MIKELANGELO Project

```
qsub -l nodes=1:ppn=16 -vm
img=/images/pool/ubuntu_bonesV01.51c.img,iocm=false,vcpus=16
/home/hpcuschi/projects/mikelangelo-bones/bones-jobscript.sh
```

The command is similar to the bare metal execution, but contains additional parameters. `-vm` is the indicator that shows qsub that this job has to be executed differently. Followed by the img the location of the VM image, `iocm=false` which will disable the iocm functionality to get more overall cores. `vcpus=16` which tells qsub that the number of virtual CPUs should be the same as the physical CPUs (`ppn=16`).

All the `-vm` are part of the new developed MIKELANGELO HPC stack. A detailed list of the functionality can be found in the deliverable D2.20.

```
=========================================================
== Log-FILE for PBS job running struct_process_mpi.sh ==
=========================================================
Job start time : Tue Jun 21 11:36:45 CEST 2016

The requested process variables were :
--------------------------------------
Project name                     : regression
Requested execution system       : CLOUD
Restart                          : N

Lower bounds of working domain : 0
                                 : 0
                                 : 0
Upper bounds of working domain : 6
                                 : 1
                                 : 1
Physical domain size             : 0.3
Binary search path               : /bones/bin
mu-CT puredat path               :
/workspace/2294.vsbase2.hlrs.de/data/
mu-CT puredat project name       : Scalar_data
Phi threshold for geometry extraction : 14250
Element order on micro scale          : HEX20
Element order on macro scale          : 1
Average RVE strain                    : 1.E-06

MM Bradcasting serialized root of size [Byte]  153040

[...]

Job finished on Tue Jun 21 11:40:55 CEST 2016
```


© Copyright Beneficiaries of the MIKELANGELO Project

The use case has been executed with several different core counts and has run multiple times with the same settings. The performance measurements can be found in D6.1 [26].

2.4 Use Case Requirements

This simulation has some explicit restrictions that have to be fulfilled to complete the computation. It depends on MPI [10] to handle the communication between workers, both on a single node and for internode communication. For Ubuntu this is handled by the "mpi-default-bin" [11] package that can be installed via apt-get from the universe repository [12]. The second requirement is the libgfortran [13]. This is the run-time library for the simulation. The application code is written in Fortran (one of the common languages used in HPC). For Ubuntu this can also be installed via the package manager apt-get. To do this in a single command:

```
$ sudo apt-get install libgfortran3 mpi-default-bin -y
```

These are the requirements for the operating system. There are additional requirements for the environment. For storing and sharing data among the separate compute nodes some sort of shared file system is needed. This is normal HPC behavior for a simulation. Most of the workers (compute nodes) need only a fraction of the data. Shipping all data to all nodes would lead to an inefficient way of calculation. In our example, a cancellous bone simulation splits the full data set into smaller pieces, which are then analysed separately. Due to the fact that the human bones have parts with denser structures and parts with less dense structures, the calculation time varies over the separate pieces. This behavior is normal, and which part will take longer to calculate cannot be predicted. Thus, it is not possible to bundle subsets which will take approximately the same time to calculate. The distribution of the work and the data has to be done dynamically. A cancellous bone simulation can use any network file system like Lustre [14] or NFS [15] to make files accessible to for all nodes. In our case we are using NFS to deliver a simple solution which can be rebuilt easily.

To increase the performance of the calculation a fast network is needed. The simulation can use all possible networks compatible with MPI, but to get maximum performance, Infiniband is the technology of choice. This is not a hard requirement for the use case to run, but is required when considering to scale up to 10,000 cores or more. The communication increases with each node used therefore a fast response time and high throughput helps to lower the boundary of the maximum scalability.

From a user perspective and for debugging of the HPC application it is important to get some sort of performance measurements. Snap is used for this and therefore is a requirement for this use case as well. As explained earlier, bottlenecks hurt scaling over more

© Copyright Beneficiaries of the MIKELANGELO Project

nodes. It is important to identify this as soon as possible, to adjust the implementation to a more performant arrangement.

To get maximum overall performance it is important to have a hypervisor that is as fast as possible. For simulations that are running for perhaps a month to complete, a single percent of slowdown per node can result in days-longer execution time. So sKVM as a faster hypervisor is needed to use the resources as efficiently as possible.

Even if the overall booting time seems negligible compared to the overall runtime, using a fast-booting operating system like OSv would also help reduce overhead and allow the most effective, and ecologically friendly, use of infrastructure.

2.5 Roadmap

The future work on this use case is to automate the performance measurements further and start to scale up the setup to a point where it reaches nearly a real production scale setup. This includes more domains to calculate and bigger domains. Due to the small size (in comparison with a real HPC setup) of the test bed, the limit of this use case will even then be below its production scale.

On the other hand the use case will be ported to OSv, which shrinks down all sorts of overheads (multiple process/thread handlers), startup times and the size of the image. This process is not a straightforward task due to the heavily changed environment. But the porting of the OpenFOAM use case demonstrates how it can be done.

The following figure presents the timeline for these developments until the end of the project.

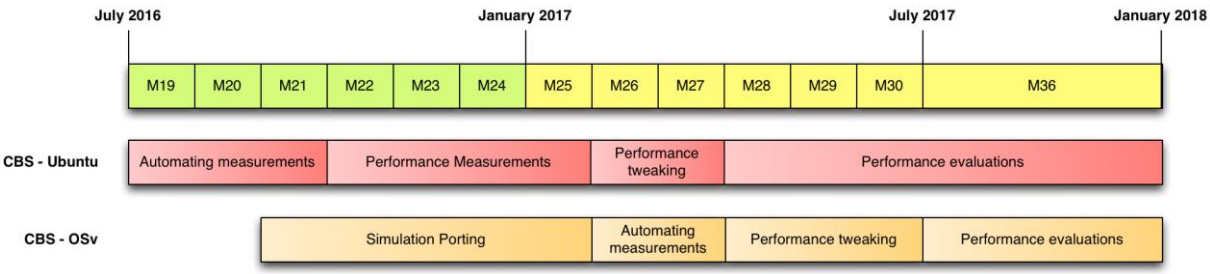

Figure 2. Time plan for the implementation, integration and evaluation of the Cancellous Bone Simulation (CBS) Use Case.

© Copyright Beneficiaries of the MIKELANGELO Project

3 Cloud Bursting Use Case

One of the key concepts of cloud services is elasticity. Elasticity is the capacity to quickly grow or shrink the size of a cloud computing resource allocation. When the workload suddenly increases the computing resource allocation can grow, consuming more hardware, and when the workload decreases the computing resource can shrink, consuming less hardware. Elasticity is in essence the core of the cloud computing promise. To do so the computing resource (in the case of the cloud bursting use case this is a database) must be able to evolve in size very dynamically without altering its nominal behavior. The extreme case of elasticity is given by Amazon Lambda [16] because it allocates computing resources to execute code snippets on the fly and the granularity of the allocation is very small (a code function). Here in the cloud bursting use case elasticity must be a property of a ScyllaDB [18] cluster (A group of machines working together to serve a ScyllaDB database). The data structure and the workload used to test this case is the cassandra-stress [17] package bundled with ScyllaDB which creates its own database schema then populates it while measuring for speed.

3.1 Dataset: Cassandra-Stress Database Schema

This schema is a couple of blobs (binary large object values) gathered in a column family the ScyllaDB equivalent of an relational database table. These blobs will be filled as fast as the stress tool can. The query per second count will be measured while the cluster is growing (by adding new machines to it) as fast as the database can allow it. Using the cqlsh command the CQL schema used by cassandra-stress can be dumped.

```
cqlsh> DESC SCHEMA
CREATE KEYSPACE keyspace1 WITH replication = {'class':
'SimpleStrategy', 'replication_factor': '3'} AND durable_writes =
true;
CREATE TABLE keyspace1.standard1 (
   key blob PRIMARY KEY,
   "C0" blob,
   "C1" blob,
   "C2" blob,
   "C3" blob,
   "C4" blob
) WITH COMPACT STORAGE
   AND bloom_filter_fp_chance = 0.01
   AND caching = '{"keys":"ALL","rows_per_partition":"ALL"}'
   AND comment = ''
   AND compaction = {'class': 'SizeTieredCompactionStrategy'}
   AND compression = {}
   AND dclocal_read_repair_chance = 0.1
   AND default_time_to_live = 0
```


© Copyright Beneficiaries of the MIKELANGELO Project

```
AND gc_grace_seconds = 864000
AND max_index_interval = 2048
AND memtable_flush_period_in_ms = 0
AND min_index_interval = 128
AND read_repair_chance = 0.0
AND speculative_retry = '99.0PERCENTILE';
```

3.2 Implementation

The plan to achieve this is to optimize every blockers that will come along our ways as we discover them.

To balance the conflicting requirements of running the regular database operations together with the streaming operations of cloud bursting, we wrote an IO scheduler inside seastar the C+14 asynchronous programming framework ScyllaDB is built upon.

The two blog entries given in [17] and [18] expose how the IO scheduler allows a fair share of resources to be given to every running seastar process. We also implemented the other task mentioned below. Then we proceeded to some small micro optimization on the ScyllaDB code. We will now implement and integrate a CPU and a network scheduler to help the resource usage get balanced between streaming and request serving. Further optimisation will be done after this if other blockers are found.

Now that the IO scheduler is implemented, work has begun on creating a cloud bursting reproducer use case, and integrating the new IO scheduler code into ScyllaDB so the streaming processes do not impact regular operation more than a given percentage.

The following is the technical work which was involved in the cloud bursting use case.

1. **Remote Procedure Call (RPC) in Seastar:**
   Seastar runs inside a single machine, but many Seastar applications, including the ScyllaDB distributed database which we use for the "Cloud Bursting" use case, offer a distributed service and therefore need convenient primitives for communicating between different machines running the same application. Thus we designed and implemented RPC capabilities for Seastar.
2. **Seastar I/O scheduling**
   One of the key requirements that arose in the "Cloud Bursting" use case was to practically support the actual cloud-bursting process.. When a Cassandra cluster grows, the new nodes need to copy existing data from the old nodes, forcing the old nodes use their disk for both streaming data to new nodes, and to serve ordinary requests. It becomes crucial to control the division of the available disk bandwidth between the two demands. For this, we designed an I/O scheduler for Seastar.

© Copyright Beneficiaries of the MIKELANGELO Project

3. **IOtune:**
   Seastar's disk API is completely asynchronous. This means that an application can start a million I/O requests to disk concurrently. However, real disks as well as layers above them cannot actually perform a million requests in parallel: if you send too many, some will be performed immediately and some will be queued in some queue invisible to Seastar introducing latency. We can no longer ensure the desired I/O scheduling, because when a new high-priority request comes in, we cannot put it in front of all the queued requests. The challenge is identifying a reasonable number of requests that can be sent in parallel.
   This requirement to tune the I/O parallelism to what the disk can actually handle led to the development of "IOtune", a tool that runs on the intended machine, tries to do disk I/O with various levels of parallelism, and discovers the optimal parallelism automatically.
4. **Seastar threads and scheduling:**
   Seastar applications do not use standard OS-level threads, and instead use much lighter-weight continuations. However, there is a lot of convenience in programming with threads, and so we added the concept of "Seastar threads" - these are not real threads but rather continuations which have their own stack, and switch out every time a future is waited-on with the "get()" method. Seastar threads also support rudimentary control over the amount of CPU time which each thread gets. This feature is useful to guarantee that low-priority background tasks in the server cannot monopolize the CPU.
5. **Miscellaneous performance improvements:**
   We discovered several places where the Cloud Bursting use case was losing performance because of inefficient code, and fixed them. For example, we improved networking performance by better batching, improved asynchronous disk I/O, and reimplemented output streams (used for both disk and network I/O).
6. **Encrypted sockets:**
   The Cassandra re-implementation for the "Cloud Bursting" use case needed support for encrypted sockets (TLS, what was formerly known as "SSL"), to allow clients to communicate securely with the server. Thus support for encrypted sockets was added to Seastar

During the current development period (M18/M24) we are focusing on CPU and network schedulers in order to be able to keep resource usage balanced during the cluster growth. Because the use case is a work in progress, we are also going to resolve any other open issues as they come along.

Cloud-bursting is currently a work in progress so no results comparable to the competition are available right now.

© Copyright Beneficiaries of the MIKELANGELO Project

3.3 Evaluation and Validation

Evaluation will be done by using the cassandra-stress tool which creates its own database schema (see above) on the fly then populates it as fast as possible (for write testing) or query it as fast as possible (for read testing).

Cassandra and ScyllaDB will be compared side by side and the comparison criterion will be the time it takes for the cluster to grow from a number of machines N1 to a number of machines N2. The query per second count must stay constant during the whole duration of the cluster growth and that is the difficult issue we are currently solving via various patches.

The following evaluation shows an intermediate result of the work on the cloud bursting use case (full comparison is not yet possible at this point). The two graphs compare behaviour of the ScyllaDB cluster while adding new database node to a cluster. The first graph (Figure 3) shows a huge decrease in the number of operations when streaming occurs, while the second one (Figure 4) shows smoother transition as a result of our improved implementation of the Seastar framework and ScyllaDB database.

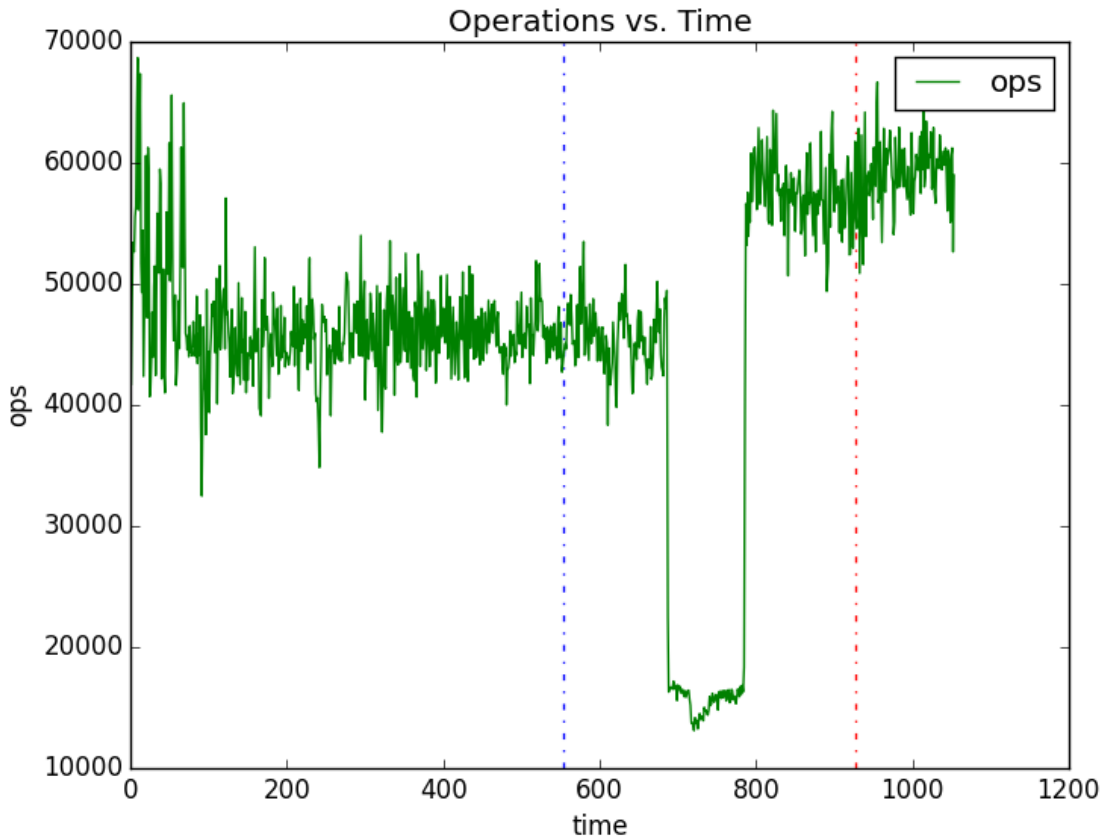

Figure 3. An example of a drop in number of operations when scaling ScyllaDB 1.1 cluster.

© Copyright Beneficiaries of the MIKELANGELO Project

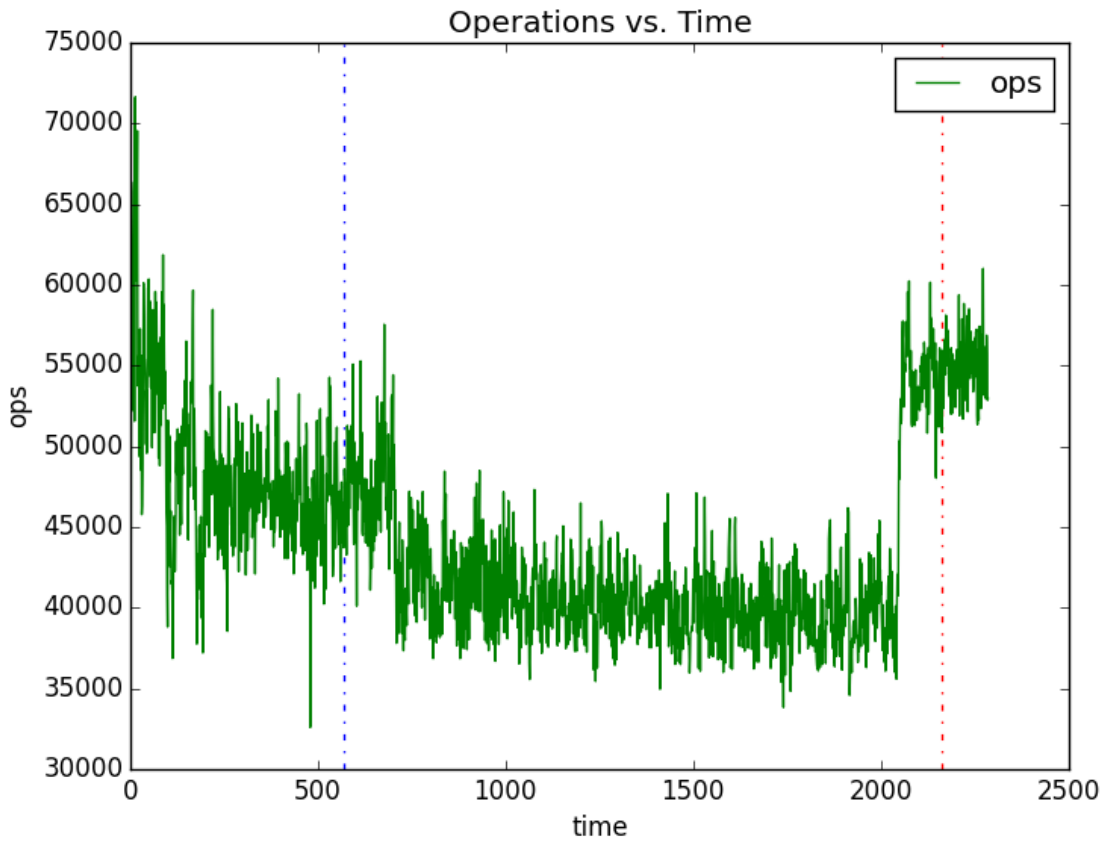

Figure 4. Example of a reduced drop when scaling Scylla 1.3 cluster.

More work is being done to improve both Seastar and ScyllaDB at the time of writing to reduce these op/s cliffs. Further progress in this cloud-bursting use case will be documented in future deliverables.

3.4 Use Case Requirements

The main requirement of Cloud Bursting is that the ScyllaDB cluster must be able to grow faster than a Cassandra Cluster while keeping its nominal performance. We can measure the growth of the cluster because we add hardware resources to the ScyllaDB cluster to make it grow. These resources can be VM instances, containers or bare metal machines.

More concrete requirements are frequently updated as part of SCYLLA's progress. These requirements are kept as generic as possible so that users of OSv, Seastar and ScyllaDB can benefit from them in their own applications.

© Copyright Beneficiaries of the MIKELANGELO Project

4 Virtualized Big Data Use Case

Here we provide an overview on the progress of the big data use case and future plans. The progress is described in the context of datasets used, the actual implementation of the use case, and the evaluation and validation of the use case. Currently, the use case builds on synthetic data sets. In future releases we aim for the use of real world data sets. The implementation of the use case has so far provided an automation of the roll-out of a big data platform and a big data benchmark. The evaluation and validation has evaluated various MIKELANGELO components, such as IOcm, snap, and OSv. Future progress is described further below for the same criteria. Datasets will be extended by real-world datasets. The implementation will be extended to include more of MIKELANGELO's components and a tighter integration with OpenStack via Sahara and Heat. Evaluation and validation will be extended by providing further benchmarks, a real world application, and integration with the upcoming continuous integration system.

The subsequent subsections describe the progress of the virtualized big data use case. Furthermore, an evaluation of the MIKELANGELO components from the perspective of the use case is provided. First, used datasets are presented. Second, the implementation progress of the use case is described. Third, an evaluation of MIKELANGELO components from the perspective of the use case is provided. Fourth, open use case requirements for other components are summarized.

4.1 Datasets

Since the big data use case has multiple phases with different data sets, those data sets need to be discussed separately. The first phase consists of the use of a big data benchmark, which comes with its own set of data. The second phase concerns itself with real-world application. As of writing this report the final real world applications, and hence their data, are not chosen yet. However, we are in the final stages of choosing appropriate use cases. In this deliverable, we are going to briefly present one of those use cases.

The data sets for the benchmarking setup have been presented already in D2.7 [21]. Each benchmark generates its own data set before computation begins. The data set sizes fall in the range of 1-4 GiB, depending on the actual algorithm. Those ranges allow us to evaluate the performance of the MIKELANGELO architecture during a relatively short duration of execution. The data set sizes have been chosen so that the workloads run for approximately 1-3 minutes. Based on our experiments it turns out that longer run times do not reveal additional insights. Thus, longer running jobs would be a waste of resources. Details regarding the generation of the data sets for HiBench can be found on its original website [23].

© Copyright Beneficiaries of the MIKELANGELO Project

Beyond the synthetic data sets generated by HiBench, this use case will operate on real world data. Currently, the two most likely real world big data applications to be used in this use case come from the University of Göttingen, which is tightly partnered with GWDG. The first use case is concerned with the analysis of open source software repositories. The use case originates at a software quality research group at the university's institute for computer science[29]. The goal of the use case is to collect data from publicly accessible git repositories and then to analyze these repositories for indicators of code quality. The application takes a hybrid approach, making use of both HPC and big data resources. The HPC part crawls for git repositories on the internet, performs preprocessing and stores the resulting data for analysis in GWDG's storage systems. The big data part then analyzes this data for various patterns of interest for software engineering research. While the first part performed on a HPC cluster lies outside of MIKELANGELO, the second part will be developed in conjunction with the MIKELANGELO big data use case, if this application will be chosen as a case study. The initial corpus of data spans only a few GiB, which fits the current state of the MIKELANGELO cloud test bed and the state of the big data use case well. The corpus of data is expected to grow continually over a period of three years. A major benefit of this application is that the big data part can be developed in tight cooperation between GWDG and the responsible group at the computer science institute at the University of Göttingen (Ifi).

The second potential real world big data application for this use case originates at the bioinformatics group at the University of Göttingen. This group is a major user of GWDG's HPC resources. The group consists mostly of biologists and chemists who run custom-designed domain-specific algorithms. A benefit of this applications is that the group operates on data sizes of many Terabyte. A downside is that the available algorithms would need porting onto the big data framework, which may require large effort. Thus, more detailed analysis of the potential to fit the application with the big data use case needs to be conducted. Furthermore, the large available corpus of data warrants a cooperation with MIKELANGELO at a more mature state.

4.2 Implementation

The use case implementation contains four major aspects: automation of the deployment of a big data cluster, integration with OpenStack, integration with MIKELANGELO components, and application-specific implementation. In future, the use case will include the development of optimization strategies to take full advantage of our stack's innovations.

Until now most of our effort went into the automation of the deployment of big data stacks in Docker and OpenStack. The automated roll-out is an important aspect of the use case since it will allow fine-grained control over the big data stack in a virtualized environment and it will be the basis for big data self-service. Furthermore, the automated roll-out will

© Copyright Beneficiaries of the MIKELANGELO Project

support our scientific work as it becomes easy to reproduce experiments. The deployment of equivalent setup clusters with Docker and OpenStack will allow a comparison in performance. This comparison will allow the consortium to narrow down any gap in performance between container-based deployments and those running in virtual machines. The current state of automation allows a semi-automated deployment of the Cloudera distribution of Hadoop (CDH) for both Docker and OpenStack. Our automation scripts use Puppet to deploy the base installation for CDH and a Cloudera Manager component. Then the user can use the Cloudera Manager to continue with the deployment. The Cloudera Manager offers an easy to use web front-end which allows the user to configure desired components to run in the already established cluster.

Currently, we are working on a tighter integration of the big data stack with the OpenStack testbed. Specifically, we have set up OpenStack Heat for workflow management. OpenStack Heat will allow us to deploy and run HiBench easily in a reproducible manner. Furthermore, Heat is the basis for OpenStack Sahara. Sahara is a project that deals with the deployment of big data stacks in OpenStack. Sahara runs Apache Hadoop out of the box. Furthermore, its plugin architecture allows integration with the Cloudera Distribution of Hadoop. Thus, we obtain the choice to use any of those two Hadoop distributions for further tests and for a big data on-demand offering at GWDG at later development stages. Sahara is expected to become part of our cross-layer optimization strategy for big data-related workloads.

The big data use case has begun to use integrated components from MIKELANGELO. On one hand the use case already runs on the MIKELANGELO cloud testbed. Furthermore, a deeper integration with OpenStack for simpler deployment and optimization will be targeted. In addition, the cluster deployments happen on nodes which contain either KVM or sKVM as hypervisors. Thus, an integration with sKVM is already available. However, this integration only contains IOcm, not SCAM or Virtual-RDMA. SCAM depends on the actual hardware architecture and first needs to be ported to the hardware available in the MIKELANGELO cloud testbed. Thus, SCAM will be deployed later in the project. Virtual-RDMA is being developed in multiple stages, called prototypes. Currently, only the first prototype is available, which requires RoCE-capable NICs, which are not part of GWDG's testbed, as they are currently not typically installed for cloud deployments. With the availability of prototype 2 we will begin integration of Virtual-RDMA as it will allow for zero-copy shared memory communication between VMs on the same host via IVSHMEM. In addition to sKVM Snap is being used in the big data use case to record telemetry data during the run of big data experiments. This data is already being used in a collaboration with the Nephele project [22] to predict net traffic.

OSv is currently receiving the focus of the implementation of the big data use case. We are investigating the use of OSv for various big data components. OSv promises to deliver

© Copyright Beneficiaries of the MIKELANGELO Project

improved IO and fast boot times in comparison to other operating systems. Preliminary tests have shown that OSv provides fast boot times in a Cassandra cluster for the first few machines when compared to a Ubuntu-based cluster. However, when more than four machines are already available, OSv stalls for a long time, such that the Ubuntu cluster catches up. With regards to IO the Cassandra cluster is able to perform twice as many queries per second with Ubuntu than with OSv, given the same hardware resources. Similar tests estimating the throughput of HDFS with Ubuntu and with OSv show considerable performance degradation when using OSv. The source of this performance degradation is currently under investigation. Further implementation of the big data use case on the basis of OSv is pending until those issues are resolved.

Most of the abovementioned implementation tasks are expected to be resolved in the near future. The next major piece of work for the use case will be the optimization of big data deployments in a cloud environment. Optimization here will include keeping the overhead through virtualization low, managing data locality, optimizing the use of IO resources to improve IO latencies and IO throughput, and minimizing interference with existing workloads in the cloud. The latter aspect is of special interest. When workload interference can be kept low, running VMs with big data jobs would allow significantly increased CPU utilization in a cloud deployment. This is of special interest since typical cloud CPU utilization is notoriously low. Cross-layer optimization at the application layer can incorporate application awareness with low-layer (sKVM, OSv) control mechanisms to optimize for CPU utilization and high IO throughput.

4.3 Evaluation and Validation

Three types of evaluations have been performed in the big data use case so far: comparison of HiBench [23] experiments[1] with Docker and OpenStack, evaluation of the effectiveness of IO cores on big data performance, and evaluation of the IO performance of OSv.

The comparison of HiBench runs in Docker and OpenStack has been used to establish a baseline performance. While Docker is expected to give peak performance, KVM is expected to show worst performance. With time, sKVM performance should fall between both. Initial results contrasting those two deployments have been published in D2.7 [21]. For compute-bound workloads there was no obvious performance gain of Docker over KVM. For IO-heavy workloads Docker fared somewhat better. The workloads ran in both cases on the same hosts, with no interfering workloads running. Both Docker and KVM have allocated all of the

[1] HiBench is a benchmarking suite for big data platforms. It is developed by Intel and has been automated for use in the cloud by GWDG within MIKELANGELO. Source code is available at github [23] and the use of HiBench in MIKELANGELO is described in D2.7 [21].

© Copyright Beneficiaries of the MIKELANGELO Project

available resources. More detailed analysis of those experiments and additional experiments will be conducted once the Sahara-based integration with HiBench and Heat is finalised.

The second type of evaluation concerns the effects of using dedicated IO cores via IOcm. An early version of IOcm has been deployed as part of sKVM in part of MIKELANGELO's cloud testbed. The big data benchmarks have been run with various numbers of IO cores. A run with no IO cores, where all of the CPU's HyperThreads are available for computation is equivalent to using normal KVM. Early experiments have not shown any speedup by adding IO cores. The exact reasons are being investigated by the IOcm developers at IBM.

The third type of evaluation concerns integration with OSv. To assess OSv for cloud bursting and for IO throughput the experiments described in the previous Implementation section have been conducted. So far the experiments show worse performance in comparison to Ubuntu as a guest OS. For the big data case, in principle, both fast boot times and high IO throughput are important. Fast boot times allow applications to burst where the workload suddenly increases at irregular intervals. High IO throughput naturally is of interest for the big data realm for data transfers. The transfer of data to the HDFS nodes is particularly important.

4.4 Use Case Requirements

Various special requirements are posed by the big data use case. The requirements concern most of MIKELANGELO's components. At the host-OS layer all three modifications to KVM, that is IOcm, Virtual-RDMA, and SCAM play an important role for the big data use case. IOcm is required to improve IO throughput, so that more CPU cycles can be used for efficient computation instead of managing IO operations. Here, both net IO and block IO are important. Net IO is required to be fast for data transfer between nodes, which happens especially at the beginning of jobs, end of jobs, and in between, when data is re-shuffled. Block IO is important, since the HDFS data nodes are expected to lie on the hosts' local block storage, in contrast to the VM images, which reside in central storage. In sKVM, Virtual-RDMA is required to improve communication between VMs on the same host by shared memory communication. Inter-host inter-VM communication can gain additional benefits by reducing communication overheads. SCAM is required to protect sensitive data such as SSH keys that are used within the big data cluster's management. In addition, for sensitive data SCAM is required to ensure the data's privacy. OSv is required to provide improved application support for key components that can benefit from a deployment in OSv. Here especially HDFS is of interest since it needs to handle a lot of IO for each Hadoop and Spark job.

4.5 Roadmap

We have devised the implementation and evaluation of the big data use case into four lines of work, as seen in Figure 5. We are going to continue to work on Sahara automation,

© Copyright Beneficiaries of the MIKELANGELO Project

focusing in particular to support both Cloudera and Apache Hadoop distributions via Sahara in OpenStack. The goal of this is to have a complete system, capable of deploying Hadoop dynamically based on users' needs.

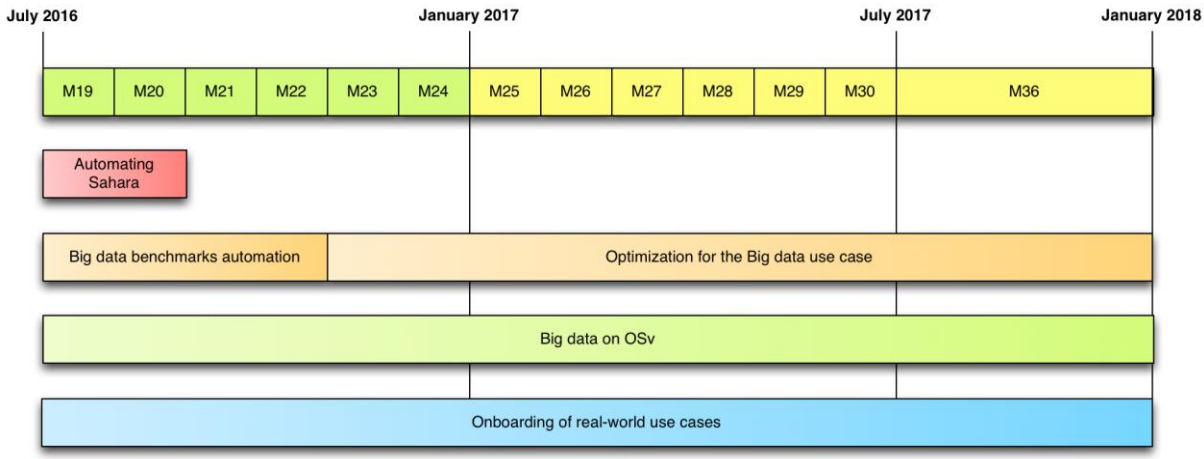

Figure 5. Big data roadmap.

For the purpose of thorough evaluation we are also working on automation of big data benchmarks (HiBench in particular). Integration is required with the custom Puppet scripts and OpenStack Heat orchestration engine. Automation is mandatory allowing continuous evaluation and optimisations of various layers of the overall architecture, which is a topic of our next implementation phase (Optimization for the Big data use case). This phase is going to be the one we intend to put most emphasis on.

In order to evaluate unikernel approach we are also going to work on the integration and evaluation of different big data tools with OSv. Preliminary studies are promising and show that integration is feasible, however a thorough performance evaluation is necessary to assess the benefits of using alternative operating systems.

The goal of the last task, running throughout the remainder of the project, is to acquire at least one real-world big data use case, based on the technologies integrated and evaluated in other tasks.

© Copyright Beneficiaries of the MIKELANGELO Project

5 Aerodynamic Maps Use Case

5.1 Introduction

The aerodynamic-map use case represents one of the two HPC related use cases in the MIKELANGELO project. By "HPC related" it is meant that such simulations are typically run on HPC infrastructure. Cloud providers have been blurring the line between HPC and cloud infrastructures. The main objective of this use case is therefore to transfer these simulations to MIKELANGELO cloud infrastructure. Expected benefits of this transition are acceleration of the aerodynamic analyses, improved agility and elasticity of application deployment and its monitoring, and simplicity of the post processing phase.

The overarching plan is to use cloud computing in order to run two different types of aerodynamic analyses. The first analysis deals with numerous similar OpenFOAM cases run simultaneously, i.e. a parametric study of a typical simple aerodynamic industrial configuration (2D airfoil and a 3D propeller in front of a wing) in order to obtain results at all required sets of parameters. Since a single OpenFOAM case examining a single parameter isn't computationally very demanding, large numbers (depending on the number of available computer resources) of such simulations can be run at the same time. An industrially relevant example could contain from 100 to 1000 such OpenFOAM cases. A new GUI will allow Pipistrel and other potential users to deploy and monitor all simulations and, at the end, to collect and display all relevant results. The transition from current (bash) scripts on a small in-house cluster to a MIKELANGELO stack deployed on much larger computing infrastructure will, using a user-friendly GUI, simplify and accelerate Pipistrel's workflow and possibly also reduce fixed operating costs in the future.

The second type of analysis deals with a single computationally very intensive simulation (distributed propulsion system with multiple propellers in front of a single wing) using all available computing resources. Such a need arises when a set of parameters is already chosen but a physically or numerically more precise (high fidelity) simulation is required. This case is a typical HPC problem and is ideal to assess the MIKELANGELO technology stack as a baseline in order to help increase the virtualised I/O efficiency of MIKELANGELO cloud stack.

The simple 2D airfoil analysis and the simple 3D propeller in front of a wing analysis, both of which typically take minutes to converge, are being used to quickly evaluate the implemented tools. A comprehensive study of full 3D configurations, typically requiring hours or even days to compute, will demonstrate the industrial relevance of the proposed tools. All OpenFOAM cases that are being used/studied in this use case are presented in D2.10 [24] and D7.16 [25].

© Copyright Beneficiaries of the MIKELANGELO Project

In order to evaluate and validate the MIKELANGELO technology stack using the Aerodynamic-map use case an OpenFOAM Cloud application has been designed and implemented as an OpenStack dashboard (more in D6.1 [26]). The OpenFOAM core and simpleFoam solver are provided as OSv packages, along with several others that play important roles in the management of distributed workloads. The OpenFOAM Cloud application is still in a prototype phase serving primarily for testing purposes. Its development is going according to the description in D2.10 [24] under the section "Expectation from the MIKELANGELO Stack".

5.2 Dataset

Computational Fluid Dynamic (CFD) simulations can be, just like other kinds of numerical calculations, very unpredictable. It can quickly happen that a bad solution or even no solution is found at all. Even though Pipistrel and XLAB are running well known and tested OpenFOAM cases in the current design phase, this won't be the case in the following phases of the MIKELANGELO project and especially not after the project. That is why it would be very convenient for end users to be able to check the status of all simulations, observe their progress and notice possible deviations from the expectations - all during execution. The OpenFOAM Cloud application should therefore enable end users to follow their simulations and alert them in case of a divergent simulation.

The progress of each simulation in OpenFOAM can be redirected from terminal display to a "log" file, which is saved in its own OpenFOAM case directory. Besides all vital information of the corresponding simulation, this log file also includes the current simulation iteration ("Time"), the elapsed time at each iteration (ExecutionTime) and, most importantly, the numerical residuals of all physical quantities (initial residual for pressure, velocity, density, etc.) the OpenFOAM solver is calculating (Figure 6). For example, the residual of pressure p represents a relative difference of pressure between two consecutive numerical iterations for one cell of the mesh. The maximal residual of each physical quantity is displayed in the log file. If residuals are decreasing over time, the simulation is converging. Typical steady simulation can be considered as converged when all residuals drop to values below 10^(-5). In such a case, the solver found a numerical solution and we can be quite confident that there are no other solutions as good as this one (for the used mesh at chosen initial and boundary condition and other setup parameters). By running the OpenFOAM command "foamLog" in the terminal, individual residuals together with the corresponding time iteration are extracted from log file and saved to separate files (p_0, Ux_0, Uy_0, ...) in folder logs/. In the case where the number of different simulations is small, these parameters are easily extracted, but in the case of large parameter searches, monitoring and extracting this data becomes unmanageable.

© Copyright Beneficiaries of the MIKELANGELO Project

```
bounding k, min: -0.00114578896 max: 689.648272 average: 5.6858817
ExecutionTime = 9667.57 s  ClockTime = 9665 s

Time = 1979

smoothSolver:  Solving for Ux, Initial residual = 6.27671336e-05, Final residual = 2.67203995e-06, No Iterations 3
smoothSolver:  Solving for Uy, Initial residual = 0.000106978187, Final residual = 4.72336162e-06, No Iterations 3
smoothSolver:  Solving for Uz, Initial residual = 9.18781902e-05, Final residual = 3.90229461e-06, No Iterations 3
GAMG:  Solving for p, Initial residual = 3.18253267e-05, Final residual = 1.56815122e-07, No Iterations 4
GAMG:  Solving for p, Initial residual = 2.42826566e-06, Final residual = 7.26507293e-08, No Iterations 2
time step continuity errors : sum local = 1.66826663e-07, global = -2.90822378e-08, cumulative = -7.47091397e-06
smoothSolver:  Solving for omega, Initial residual = 8.05991101e-07, Final residual = 4.53621562e-08, No Iterations 4
smoothSolver:  Solving for k, Initial residual = 7.82710392e-06, Final residual = 5.13635683e-07, No Iterations 4
bounding k, min: -0.000641512743 max: 689.642383 average: 5.68557846
ExecutionTime = 9669.93 s  ClockTime = 9668 s

Time = 1980

smoothSolver:  Solving for Ux, Initial residual = 6.28244308e-05, Final residual = 2.66689123e-06, No Iterations 3
smoothSolver:  Solving for Uy, Initial residual = 0.000107054069, Final residual = 4.72021646e-06, No Iterations 3
smoothSolver:  Solving for Uz, Initial residual = 9.18198958e-05, Final residual = 3.89342596e-06, No Iterations 3
GAMG:  Solving for p, Initial residual = 3.22007392e-05, Final residual = 1.52604569e-07, No Iterations 4
GAMG:  Solving for p, Initial residual = 2.42559427e-06, Final residual = 6.78793381e-08, No Iterations 2
time step continuity errors : sum local = 1.55875336e-07, global = -2.70865592e-08, cumulative = -7.49800053e-06
smoothSolver:  Solving for omega, Initial residual = 8.03142676e-07, Final residual = 4.51429271e-08, No Iterations 4
smoothSolver:  Solving for k, Initial residual = 7.81633991e-06, Final residual = 5.12240925e-07, No Iterations 4
bounding k, min: -0.000390606063 max: 689.638055 average: 5.68522958
ExecutionTime = 9672.3 s  ClockTime = 9670 s

forces forceCoeffs1 output:
    sum of forces:
        pressure : (-63.6732876 -1.23690818 -0.200950653)
        viscous  : (0.115722045 -0.0203749105 -0.00509786779)
        porous   : (0 0 0)
    sum of moments:
        pressure : (3.73118564 -0.680250604 1.12532157)
        viscous  : (0.117870649 -0.00415004142 0.018555342)
        porous   : (0 0 0)
```

Figure 6. A section of a log file of a typical OpenFOAM simulation indicating two time iterations of a simulation and an intermediate forces calculation.

If needed, OpenFOAM can also calculate aerodynamic forces and moments that act on aerodynamic bodies as a consequence of the fluid passing by. For example, when simulating an airflow passing an airplane wing, such wing produces a lift and a drag force and also moments in all three axes. These forces and moments are for Pipistrel one of the most important outputs of the simulation. That is why it is very important for these quantities to converge as well. It can happen that even though all residuals converged, the forces and moments still don't stabilize. That is why they have to be observed during the simulation. Calculated forces and moments are saved in the log file beside all residuals (Figure 6), and also in a separate file "forces.dat" located in folder postProcessing/ (Figure 7).

© Copyright Beneficiaries of the MIKELANGELO Project

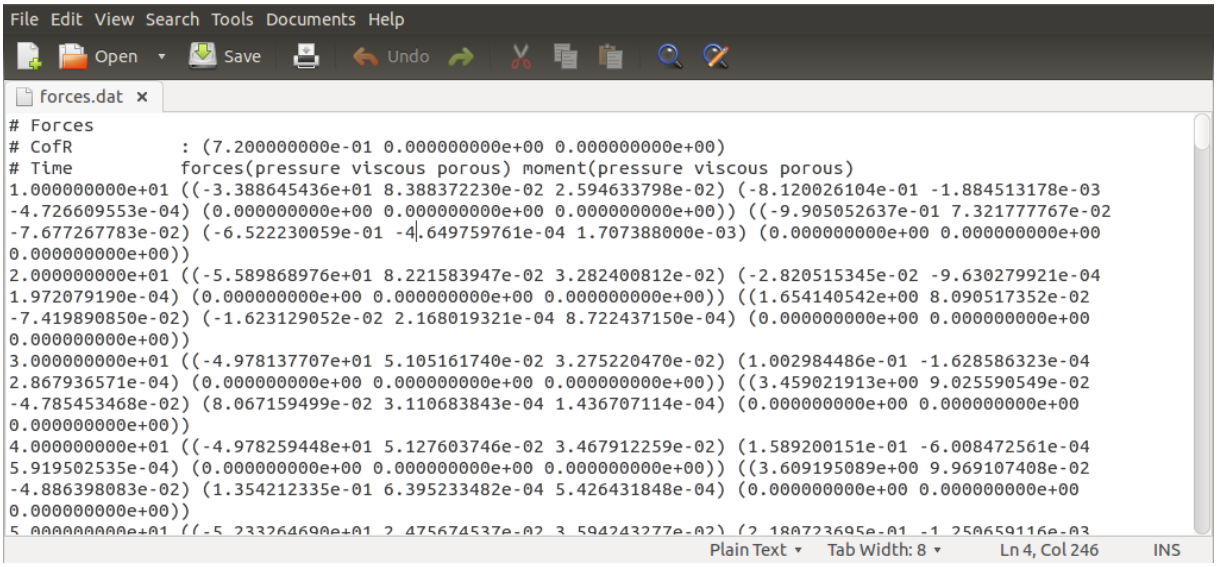

Figure 7. Beginning of a forces.dat file containing forces and moments calculated on chosen aerodynamic surfaces.

The current version of OpenFOAM Cloud enables the end user to separately display log files of each simulation. The prototype version of OFC also enables the users to graphically display chosen residuals for each simulation separately in Grafana. Data are collected using the snap telemetry framework and published into a central time-series database. All data are furthermore properly tagged (using simulation id, parameter specification, etc) allowing the historical revision of processed data. If the number of simultaneous simulations is small, there is indeed no problem for the end user to check log files or display residuals of all simulations. But if the number of simulations increases to 100 or 1000, it is not practical to manually check residuals and forces of all simulations.

A smart way of displaying the collection of all residuals and forces therefore needs to be implemented in order to enable the end user to postprocess the results with confidence that all simulations converged. The visualisation should focus on two aspects. First, the visualisation of real-time data over simulation time. This aspect is important as it allows scientists to assess the behaviour of the input case under given conditions. The second consideration requires aggregation of data across simulations, for example visualising a single simulated parameter over all input parameter variations. Using the same data, collected with snap, different types of automated alerts may furthermore be introduced.

5.3 Implementation

As we have seen in the previous sections, the purpose of the Aerodynamics Map use case in the MIKELANGELO project is twofold. The use case itself can be recognised as an HPC use case validating and evaluating the overall performance improvements of the MIKELANGELO technology stack. On the other hand, the broader purpose of the use case, namely a cloud

© Copyright Beneficiaries of the MIKELANGELO Project

based OpenFOAM simulation infrastructure, offers an important insight into the manageability improvements provided by the MIKELANGELO project over traditional approaches. These approaches are presented in detail in the deliverable D6.1 [26].

In order to differentiate between the two aforementioned scenarios, this use case has been further refined by offering two variants. In order to demonstrate the benefits in an HPC environment a large OpenFOAM input case has been designed. On the other hand, regarding evaluation of benefits of MIKELANGELO with respect to management agility, a number of smaller cases have been designed supporting execution of simplified workloads in public/private clouds.

The following subsections focus on these two aspects of this use case. First, a brief description of the existing work is presented followed by a description of additional requirements for the underlying stack.

5.3.1 OpenFOAM Cloud

OpenFOAM Cloud is a web based simulation management application. It integrates several components of the MIKELANGELO project into a single, unified application. A high level architecture is presented in diagram in Figure 8. The current prototype implementation relies on an OpenStack-based infrastructure for computational resources. The user facing application is integrated directly into OpenStack dashboard (Horizon) and communicates via REST API with the backend logic. This guarantees separation of the application from the underlying infrastructure provider, allowing it to be supported on top of well-known public cloud offerings such as Amazon AWS or Microsoft Azure.

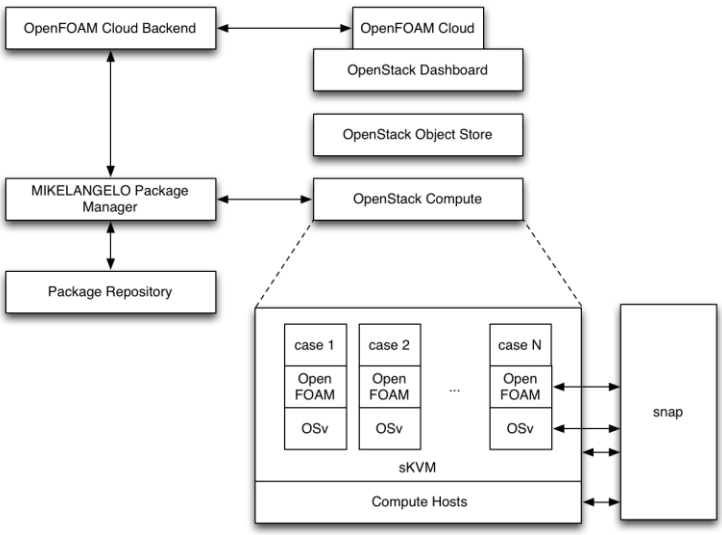

Figure 8. Architecture diagram of the OpenFOAM Cloud application.

© Copyright Beneficiaries of the MIKELANGELO Project

As seen in the diagram in Figure 8 OpenFOAM Cloud integrates most of the components of the MIKELANGELO project. The following list presents the details about the integration of individual components and the way the components have been used to offer a high level cloud-enabled application.

sKVM. Thanks to its full backwards compatibility with KVM, the integration of sKVM required no additional work on the application layer. Due to the current hardware limitations of the cloud testbed, the most notable component used in this use case was the IOcm [30], allowing dynamic allocation of cores responsible for I/O intensive workloads. vRDMA and SCAM components have not been used in the cloud version of the OpenFOAM use case.

OSv. The main benefit of OSv compared to other general purpose operating systems is its small size and fast boot time allowing OpenFOAM simulations to be launched in almost no time, even on a distributed system. OSv furthermore provides a high level management interface (HTTP REST API) suitable for controlling and monitoring of the simulations. One of the most challenging tasks during the initial phase of the project was to ensure full support for OpenFOAM applications on top of OSv as described in deliverables D2.10 [24] and D4.4 [28].

MIKELANGELO Package Management. This tool is integrated into the OpenFOAM Cloud, supporting on-the-fly generation of virtual machine images based on users' requirements. Target VMs are very small containing only the components necessary to execute an OpenFOAM simulation. This allows for highly optimised deployment on a distributed infrastructure where images have to be stored in a centralised image repository and deployed on target compute nodes.

Snap. The section on datasets above presented a vast amount of data that OpenFOAM simulations provide to the scientists during and after the simulation execution. Scientists typically create special scripts for extracting the valuable information out of the data and presenting it in a suitable form. However, this approach becomes tedious when several simulations need to be executed in parallel, varying the initial conditions of the simulation. OpenFOAM Cloud addresses this by integrating snap [31] telemetry framework. Each launched simulation is accompanied with a separate snap data collector feeding simulation parameters into a live charting tool (Figure 9) for the scientist to analyse crucial data. On top of this, snap is used to monitor all aspects of the physical and virtualized infrastructure (CPU utilization, disk and network I/O metrics etc.).

OpenStack Horizon. On the topmost layer, OpenFOAM Cloud is an OpenStack dashboard (Figure 10), fully integrated into OpenStack Horizon application. This allows tight integration with the cloud middleware, similar to that of OpenStack Sahara for big data.

© Copyright Beneficiaries of the MIKELANGELO Project

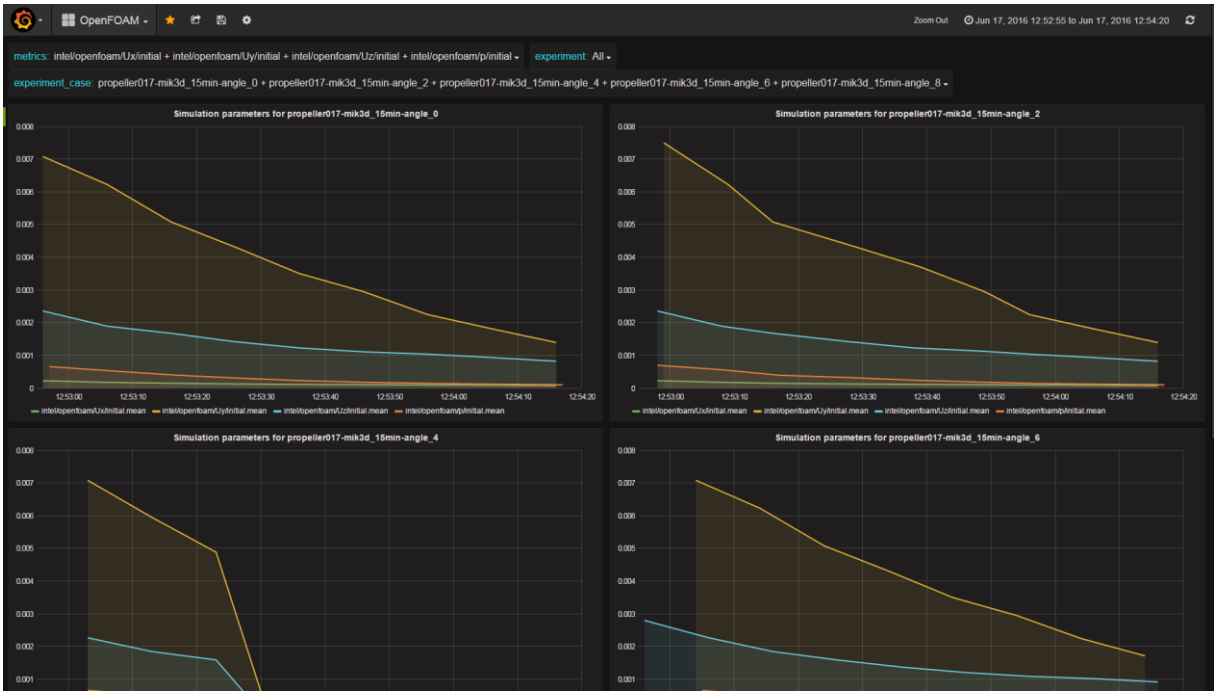

Figure 9. Simulation data of four OpenFOAM cases run simultaneously, collected by snap and visualised in Grafana application.

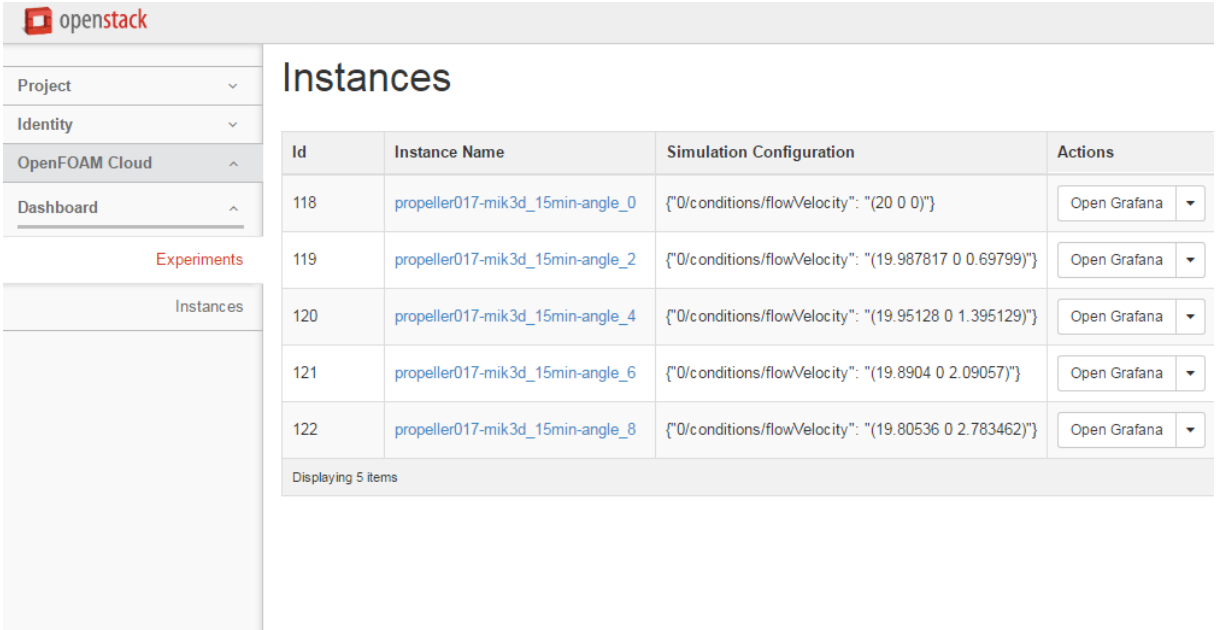

Figure 10. OpenFOAM Cloud as an OpenStack dashboard.

5.3.2 OpenFOAM HPC

Contrary to the cloud version of the Aerodynamics use case presented in the previous section, the main focus of the HPC-based use case is to validate and evaluate components specific to the HPC environment: MPI (Message Passing Interface), NFS (Network File System)

© Copyright Beneficiaries of the MIKELANGELO Project

and virtualized RDMA for efficient communication between workers in a multi-node cluster setup.

In order to run parallel workloads, OpenFOAM relies on Open MPI, an open source implementation of the MPI interface. Parallel work is distributed to several processes running on one or more compute nodes and these worker processes communicate over a shared channel provided by the MPI infrastructure. However, because OSv is a unikernel, it has no notion of multiple processes - the entire guest virtual machine is running as a single process. To support parallel execution in OSv, the Open MPI needed to be extended to integrate OSv and its management calls for running separated threads instead of processes. An important consequence of this work is that there have been no modifications to the OpenFOAM application in order to support modified Open MPI. This will allow all other applications running on top of Open MPI to run virtualized in OSv.

The integration of the NFS and vRDMA required no additional work apart from the implementation of the corresponding components, again ensuring unmodified applications may utilise the new MIKELANGELO technologies. Deliverable D6.1 provides an extensive evaluation of individual components.

5.4 Use Case Requirements

Even though the MIKELANGELO project primarily focuses on the performance improvements, sheer performance is not a vital part of this particular use case. The Aerodynamics use case assesses the MIKELANGELO project results mostly from the perspective of ease of manageability. The OpenFOAM application has been under active testing and development since the project start so many of the missing features of underlying technologies have already been resolved to the degree that they are working fully integrated. However, a thorough evaluation of this use case, presented in deliverable D6.1 [26], has shown several drawbacks that can be summarised as follows:

- **Compatibility issues.** When running applications in OSv (or any other unikernel for that matter) applications have to be at least recompiled. More frequently, the unikernel or the application itself need to be modified to support or overcome any missing functionalities.
- **Performance penalties.** Current research in the MIKELANGELO project has shown some potential causes for the performance losses that have been observed to-date, and are currently being investigated.

Requirements presented in the following subsections are all high-level in that they do not provide a clear technical specification. The MIKELANGELO project is still in a very active research phase with short development cycles consisting of implementation of new features

© Copyright Beneficiaries of the MIKELANGELO Project

and evaluation thereof, producing the results for the next iteration. The requirements therefore describe the final goal for this use case and not the actual low-level features that are required to achieve these goals.

5.4.1 End-user Requirements

Optimising a physical model of airplanes (or any other OpenFOAM experiment) requires running numerous simulations, comparing the results of various what-if scenarios. Typically, this is achieved with a set of shell scripts extracting simulation data and exporting them into charts and spreadsheets analysed by the scientist. Using the OpenFOAM Cloud application, and in particular the various components it integrates, has already demonstrated significant simplification over a traditional approach by offering the scientist the data they require during and after the simulation.

In order to support the potential reach of this use case for the purposes of the further dissemination and exploitation of the MIKELANGELO project, we propose the following requirements for the implementation of this use case. These requirements are strictly related to the implementation of this use case - requirements related to underlying components of MIKELANGELO are presented in the following sections.

- **Extend application composition capabilities**. The current version only supports one single OpenFOAM application solver (simpleFoam), but in practice users have to work with different solvers, potentially even providing their own.
- **Management of input data.** The current application workflow requires users to upload their data onto a dedicated storage service (for example OpenStack Swift or Amazon S3). A more natural way of submitting the data would be to upload the data along with the submitted simulation request.
- **Management of output data.** Even though snap telemetry provides an overview of the simulation data, it is still important for the user to acquire the results in a way suitable for further analysis in a tool such as ParaView. Output data related to simulations should be stored and easily accessible through a common application interface.
- **Parameter set-up.** Migration from the current, rather cumbersome, procedure of providing a set of parameters through JavaScript Object Notation (JSON) to an additional, Excel like, input document.
- **Simulation data plotting.** The expanded possibility of displaying the collection of all residuals and aerodynamic forces needs to be implemented. As explained before, visualisation of real-time data over simulation time, and an aggregation of data across all input parameter variations, should both be possible. Different types of automated alerts may also be introduced.

© Copyright Beneficiaries of the MIKELANGELO Project

5.4.2 OSv Requirements

The true potential of the OSv unikernel in the case of the OpenFOAM application remains to be seen in the remainder of the project. It is already much easier to manage a set of compact virtual machines running OSv, however the performance penalties imposed by the current version of the OSv are still slightly higher than expected (see D6.1 [26] for more information). According to this evaluation, the most notable requirement in that respect is the proper use of NUMA (Non-Uniform Memory Access) information provided by the virtualization layer (hypervisor) to the guest operating system. Linux guests are capable of delivering this information to user applications running on top, allowing Open MPI to optimise the pinning of worker processes to actual CPU cores. On the other hand OSv does not expose this information, resulting in a sub-optimal pinning of worker threads.

OSv must also fully support execution of the mpirun command in parallel to the worker thread running in the same virtual machine. Currently, it is required that a separate (head) VM is launched just for the mpirun itself and the worker threads are then spawned in a new virtual machine. From the perspective of this use case and the other HPC-related use cases, requiring a separate head VM is not desirable as it imposes another hard requirement for the integration layer.

5.4.3 Package Management Requirements

One of the greatest features of OpenFOAM is the vast number of existing applications users can easily reuse to run their simulations. In MIKELANGELO we have focused on two special cases that are actually used by the Aerodynamics use case. However, in order to allow for efficient exploitation of this use case, more applications (OpenFOAM solvers) should be validated against the MIKELANGELO technology stack.

The application management tool must ensure support for various cloud middlewares, for example Amazon EC2 and OpenStack. Even though the current version of the tool produces runnable images, it is an impediment to broader use if these images must be deployed in target environments manually.

5.4.4 The Snap Requirements

Snap has already proven extremely useful for gathering all kinds of data related to the execution of the OpenFOAM simulations. One of the most important functionalities missing at the time of writing this report is the ability to dynamically adjust monitored metrics and the frequency of the data collection. For example, in the early stages of a simulation the user might be interested in just a few main metrics being captured more frequently as they are analysing whether the model is converging. Following the initial phase, the user (or the

© Copyright Beneficiaries of the MIKELANGELO Project

system) should be able to change the list of metrics as well as the frequency of collecting the data.

Efforts are underway to address these issues. An Anomaly Detection plugin is being released shortly which will cache telemetry data locally, then use the Tukey Method to automatically identify statistically noticeable changes in the readings, and publish the data surrounding the 'anomaly' at maximum resolution. Relatively steady-state telemetry data, on the other hand, is published at a much reduced resolution to minimise overhead. This new plugin may fully address the request to adjust the frequency of the data collected, but adjustments are automatic rather than manual, which may or may not be an issue. Further analysis is required to see how best to address the request to dynamically adjust the list of metrics being monitored.

5.4.5 HPC Environment Requirements

Besides supporting several of the most popular cloud middlewares, this use case focuses on the HPC environment as well. With the built-in support for Open MPI mentioned in the section about OSv requirements above, the OpenFOAM use case should fully utilise the HPC infrastructure offered by the Torque scheduler. Instantiation, contextualisation and configuration of OpenFOAM simulations must be done independent of the underlying technology (for example: Linux vs OSv, KVM vs sKVM).

5.5 Roadmap

We have presented two alternatives to running the Aerodynamics use case, namely the cloud and HPC version. The purpose of the former is to validate and evaluate flexibility, manageability as well as performance improvements based solely on the components of the MIKELANGELO project. The HPC environment primarily targets the performance evaluation in specific environment equipped with high performing hardware components. To this end, the implementation in HPC environment focuses on:

1. Continuous evaluation. Running simple and complex workloads under various configurations. These include VM configuration (number of VMs, their placement in the physical cluster), operating systems and MIKELANGELO component configuration (for example, IOcm cores or virtual RDMA capabilities).
2. Support for MPI. Because of its omnipresence it is essential that the entire technology stack is optimised for MPI. This use case is the main driver for setting new requirements for other work packages as it facilitates thorough validation.
3. Development of alternative OpenFOAM applications. So far we have focused on the specific OpenFOAM application used by the concrete Pipistrel use case. By the end of year 2 we are going to provide support for additional standard applications based on

© Copyright Beneficiaries of the MIKELANGELO Project

OpenFOAM. The overarching goal would be to support all the applications that are freely available by OpenFOAM, however there may be additional specific caveats. To this end, we intend to support at least 5 additional applications, provided as pre-packaged applications.

The OpenFOAM Cloud application on the other hand has far greater exploitation potential and is going to be further extended by the end of the second project year. Besides showcasing the results of the MIKELANGELO project, this will allow Pipistrel and XLAB to reach out to interested audiences much more easily and concretely.

The first goal is to allow configuration of arbitrary workloads. Instead of allowing only a single worker per simulation, it should be possible to submit jobs to the cloud with several workers on a single VM or even multiple VMs. Cloud environments, private or public, may also be limited by the available resources. This limits prevent the current prototype from spawning a parameter search experiment where the cumulative resources exceed available ones. The backend logic is going to be extended in a way to allow scheduling of jobs. This will further be supported by preservation of all relevant results in a common object storage.

For the final year of the MIKELANGELO project we plan to focus on the following topics:

1. Extended integration with snap. Given the flexibility snap offers, we are interested in additional data collection capabilities as well as some processing. For example, the collector should allow flexible selection of monitored parameters, depending on the simulation itself. Additionally, we would like to be able to aggregate data from different simulations.
2. Support for various OpenFOAM applications. Similarly, to the HPC environment, we would like to integrate support for additional OpenFOAM applications.
3. Integrate with at least one public cloud provider. Although the OpenStack based application is suitable for use case presentations, integration with a public cloud such as Amazon AWS is inevitable. The design of the application is mostly independent of the infrastructure itself allowing seamless integration.

The following figure presents the provisional timeline for the two main development branches, namely the OpenFOAM HPC and Cloud. Performance evaluations are going to run periodically as indicated by the red activity boxes. Other activities are indicated by the expected delivery date.

© Copyright Beneficiaries of the MIKELANGELO Project

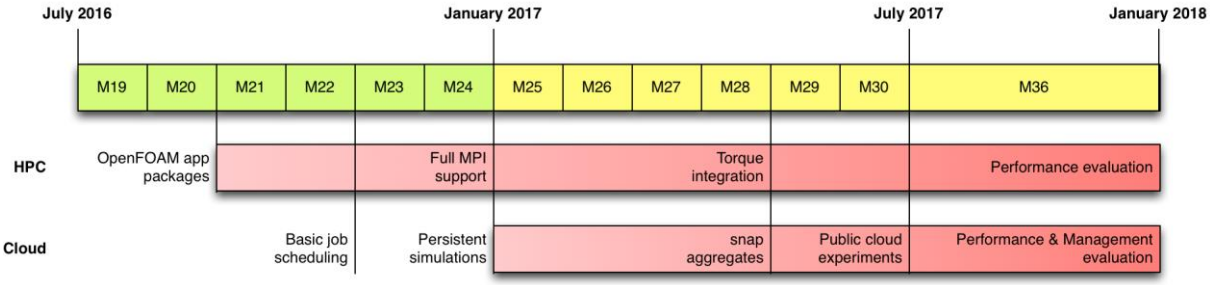

Figure 11. Time plan for the implementation, integration and evaluation of the Aerodynamics use case.

The plan is going to be closely monitored and adapted. The focus of this use case implementation remains to be the integration, validation, evaluation and exploitation of MIKELANGELO results.

© Copyright Beneficiaries of the MIKELANGELO Project

6 Concluding Remarks

HPC systems, mainly supercomputers, often have powerful but also expensive computer hardware and network infrastructure as they are designed to achieve the highest performance. On the other side, cloud systems often have more conventional computer hardware and network infrastructure which is less expensive and easier to expand.

However, despite HPC systems being faster, most of the scientific data processing applications are recently developed for the cloud. This is because cloud systems offer virtualization to the programmers: they don't need to change their application depending on the hardware of the target infrastructure. The cloud also provides a more efficient use of the resources, because the hardware can be shared between different jobs from different users and this doesn't mean a loss of performance necessarily as some applications are CPU bound that can run simultaneously with others that are I/O bound on the same hardware, without any interference or significant performance degradation.

Four use cases are presented in the MIKELANGELO project that try to apply virtualization in HPC systems like supercomputers, and enhance traditional clouds with less overhead. The advantages mentioned above for Cloud systems will apply now also to HPC systems: the application will get the benefits of high performance computer hardware and network infrastructure in terms of execution time and storage capacity, and also this hardware and network can be used in a more efficient way, for example with sparse nodes used on demand, in the event of a hardware defect or maintenance.

To keep the high performance execution of HPC systems, different optimizations have been performed in these uses cases to minimize as much as possible the overheads of virtualization. Optimisations include the use of a unikernel operation system - OSv - with less overhead regarding process handling, fast boot and shutdown times, and an optimized hypervisor - sKVM - with I/O enhancements like IOcm and infiniband virtualization with virtual RDMA.

The use cases presented here are all at different levels of maturity. However, progress is being made daily and the advances in one often benefit the others as they share several software components such as OSv and sKVM. At this intermediate stage of the project there are challenges that need to be overcome in all the use cases, but promising progress has already been made as this implementation strategy is pursued.

© Copyright Beneficiaries of the MIKELANGELO Project

7 References and Applicable Documents

[1] The MIKELANGELO project, http://www.mikelangelo-project.eu/
[2] LXC Project page, https://linuxcontainers.org
[3] Docker project page, https://www.docker.com
[4] OSv project page, http://osv.io/
[5] Ubuntu Cloud images, https://cloud-images.ubuntu.com/
[6] Torque project page, http://www.adaptivecomputing.com/products/open-source/torque/
[7] Libvirt domain XML documentation, https://libvirt.org/formatdomain.html
[8] Cloud-init documentation, https://cloudinit.readthedocs.io/en/latest/
[9] Documentation for sparcify, http://libguestfs.org/virt-sparsify.1.html
[10] MPI project page, http://www.mcs.anl.gov/research/projects/mpi/
[11] MPI package for ubuntu, http://packages.ubuntu.com/en/precise/mpi-default-bin
[12] Ubuntu univers Reposetroy, https://help.ubuntu.com/community/Repositories/Ubuntu
[13] Package for libgfortran, http://packages.ubuntu.com/de/precise/libgfortran3
[14] Lustre project page, http://lustre.org/
[15] NFS information, https://en.wikipedia.org/wiki/Network_File_System
[16] Amazon Lambda , https://aws.amazon.com/lambda/details
[17] Apache Cassandra, supplying tools for Scylla, https://github.com/scylladb/scylla-tools-java
[18] ScyllaDB, http://www.scylladb.com
[19] Designing a Userspace Disk I/O Scheduler for Modern Datastores: the Scylla example (Part 1), http://www.scylladb.com/2016/04/14/io-scheduler-1
[20] Designing a Userspace Disk I/O Scheduler for Modern Datastores: the Scylla example (Part 2), http://www.scylladb.com/2016/04/29/io-scheduler-2/
[21] D2.7 - The First Virtualised Big Data Use Case Implementation Strategy, https://www.mikelangelo-project.eu/deliverables/deliverable-d2-7/
[22] NEPHELE project page, http://www.nepheleproject.eu/
[23] HiBench project page, https://github.com/intel-hadoop/HiBench
[24] D2.10 - The First Aerodynamic Map Use Case Implementation Strategy: https://www.mikelangelo-project.eu/deliverables/deliverable-d2-10/
[25] D7.16 - The initial Data Management Plan: https://www.mikelangelo-project.eu/deliverables/deliverable-d7-16/
[26] D6.1 - First report on the Architecture and Implementation Evaluation: https://www.mikelangelo-project.eu/deliverables/deliverable-d6-1/
[27] D3.1 - The First Super KVM - Fast virtual I/O hypervisor: https://www.mikelangelo-project.eu/wp-content/uploads/deliverable-d3-1/

© Copyright Beneficiaries of the MIKELANGELO Project

[28] D4.4 - OSv – Guest Operating System – First Version: https://www.mikelangelo-project.eu/wp-content/uploads/deliverable-d4-4/
[29] Uni Göttingen webpage, https://www.uni-goettingen.de/en/138524.html
[30] D3.1 The First Super KVM - Fast virtual I/O hypervisor, https://www.mikelangelo-project.eu/wp-content/uploads/2016/06/MIKELANGELO-WP3.1-IBM-v1.0.pdf
[31] Intel snap monitoring, https://github.com/intelsdi-x/snap

© Copyright Beneficiaries of the MIKELANGELO Project