MIKELANGELO

MIKELANGELO

D7.18

The second update on the Data Management Plan

Workpackage	7	Exploitation, Dissemination, Communication and Collaboration	
Author(s)	Daniel Vladušič, Gregor Berginc		XLAB
	Uwe Schilling		USTUTT
	John Kennedy, Marcin Spoczynski		INTEL
	Gabriel Scalosub, Niv Gilboa		BGU
	Peter Chronz		GWDG
	Matej Andrejašič		PIPISTREL
	Nadav Har'El, Benoit Canet		SCYLLA
Reviewer	Matej Andrejašič		PIPISTREL
Reviewer	Gabriel Scalosub		BGU
Dissemination Level	PU		

Version History

Date	Author	Comments	Version	Status
2016-08-10	Daniel Vladušič	Initial draft	V0.0	Draft
2016-12-20	All authors	Added missing chapters, data	V0.7	Review
2016-12-22	All authors	/	V1.0	Final

Executive Summary

This is the third version of the deliverable presenting MIKELANGELO contributions to the Open Data initiative. The purpose of this deliverable is to describe the data used and generated in the MIKELANGELO project. This report accompanies the Open Data repository provided by the project and elaborates its content. The data collected and described in this deliverable consists of the input data used by different use cases, the actual raw data produced by our experiments (i.e., measurements, results) and the ways of generating data (i.e., scripts, environment descriptions). When appropriate, this document contains brief descriptions of the use cases, data generation mechanism, types of input and output data. The descriptions of data, provided in this document, allows external parties to analyse and reproduce the experiments performed in MIKELANGELO.

This document is an update of the D7.17 - The first update on the Data Management Plan and serves as the cover document for all the data provided by the MIKELANGELO project for Year 2.

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)

Table of Contents

Table of Figures

Table of Tables

1 Introduction

This deliverable is an update of D7.17 - The first update on the Data Management Plan [1] and it focuses on the data generated in Year 2 of the MIKELANGELO project.

In this document we assume the reader is familiar with Open Data Pilot and Data Management Plans in general.

The experimental data is available online, using an OwnCloud instance hosted at XLAB[2]. The data will be referenced on the MIKELANGELO website[3]. Data significance is determined with the use of data in official presentations, papers and press releases. The data that is significant enough will be pushed to Zenodo service.

The document is organised as follows. In the first part the MIKELANGELO use cases input data is briefly described - most notable changes are in the Aerodynamics and Big Data use cases, which have evolved compared with Year 1, thus D7.17. In the second part of the deliverable, the descriptions of the data generated in the MIKELANGELO project are provided.

It must be noted that gathering of data in such form is still a process that requires a lot of effort as data producers are not accustomed to such transparency and data gathering requirements.

2 The Use Cases Input Data

This section provides the shortened descriptions of the MIKELANGELO use cases and their data. The previous version of descriptions is part of deliverable D7.17 (please see Section 2). Report D6.3 (First report on the Use Case Implementations) [29] provides further information on the four use cases in terms of their implementation status and additional evaluation results.

2.1 The Aerodynamic Map Use Case

The aerodynamics use case data originates from Pipistrel. This project partner is providing their OpenFOAM [4] calculation data for aerodynamics of lightweight aeroplanes.

As already stated in D7.16 and D7.17 the initial case will study a 2D airfoil, whose polar curve will be obtained through OpenFOAM Cloud application. Once the methodology is proven, a computationally more expensive 3D case will be studied. The case will consider distributed propulsion system with a planar wing and multiple propellers. In order to allow for larger deployments of the OpenFOAM use case, a more complex input case has been prepared [32]. It differs from the simpler propeller-wing aerodynamic simulation, found in previous input cases, mostly by taking into account the complete geometry of the propeller. It also simulates more than one propeller. Each propeller is therefore not simulated as a pressure difference disc, as in the simpler case, but a more complex multi reference frame (MRF) simulation is being applied. The simulation is still using a steady state incompressible flow solver (simpleFoam), but the propellers and their near vicinity are being calculated in a rotating frame of reference. The case is prepared in such a way to support arbitrary number of propellers. By varying the number of propellers an OpenFOAM case of an arbitrary size (although discrete) can be prepared. Further details on the use case can be found in report D6.3 First report on the Use Case Implementations [29].

This example study consists of 3D CAD models of some of the geometries studied through the OpenFOAM Cloud application.

2.2 The Cloud Bursting Use Case

The Cloud Bursting use case involves installing Cassandra[5] and ScyllaDB[6], and then stimulating the database with a load generated with Cassandra stress tool. After the initial database cluster is fully loaded, new nodes are added to the cluster, simulating the scaling of the database cluster in case of increased workload. This kind of evaluation represents a

typical cloud bursting scenario. With this use case, we do not use any sensitive data - the data which Cassandra and ScyllaDB will receive from Cassandra stress is completely random.

2.3 The Cancellous Bones Use Case

The data used in this use case deals with the development of the material modelling of micro-structured cancellous bone tissues on the continuous mechanical scale from the engineering point of view. The dataset handed to the project was generated in the following way: The intact femoral head was scanned using a technical micro focus computer tomography system at the ''Institut für Bauweisen und Konstruktionsforschung – DLR Stuttgart''. The resulting data set is a volume data set consisting of a density field and the header data necessary to describe the regular grid. This is our source data set, which is briefly described below. This use case serves as a well-known test - USTUTT and more precisely, HLRS, know this use case very well and can use it as a benchmark for the tightly-coupled MPI-based problems. This means we shall not delve into specifics of the use case but rather provide a high-level description of the source data. The format of the data is called PureDat and is described below.

The source data set consists of 4 files:

1. The density field with 1680 x 1740 x 1752 image points of 4 Byte resolution.
2. The binary header describing the resolution of the regular grid by 3 x 8 byte floating point data.
3. The ASCII header holding the description of the data which is:''Micro-CT of a femoral head taken at the DLR Stuttgart on the 17. February 2010''
4. An ASCII header describing the relevant data chunk positions of the three files mentioned above.

These four files form the so called ''PureDat'' data format which in total consists of approximately 20.5GB of raw data which no longer contains any information related to the patient, neither medical nor personal, from whom the femoral head was taken.

The fundamental idea behind PureDat was to develop a file format which separates the data types in memory to minimize the necessary system and library calls. PureDat is used as the input file format as well as the output. The input Data for the Cancellous bone simulation is described in D2.1 [22].

The execution of the Cancellous Bones use case itself generates continuous mechanical material data of the cancellous bone structures on various resolution scales.

2.4 The Virtualised Big Data Software Stack Use Case

The goal of the big data use case is to integrate MIKELANGELO's virtualisation stack with big data technologies. As part of the evaluation of GWDG's big data stack, the stack as a service will be offered to partners within the Max Planck Society, University of Göttingen, and the State and University Library Göttingen.

In year 2 two real-world workload have been defined for the big data use case: the Mining Software Repositories (MSR) workload and the digital humanities (DH) workload. Both workload are described in more detail in D6.3 [D6.3]. Both workload are starting out with their applications and thus have growing data sets. The MSR workload performs analysis of software repositories in the field of "mining software repositories". The data consists of raw data grabbed from public git repositories. The DH workload performs quality analysis for the EUROPEANA project. There the data consists of metadata from thousands of European institutions that hold items of value to the humanities. Both workload are described in more detail in D6.3. The following paragraphs summarize the data sets.

The MSR workload has accumulated less than 100GB of data. However, we expect the data to grow to more than 10TB. The data is coming in burst from the data collection on the HPC. The HPC cluster can run more than 100 processes at the same time leading to a large amount of data in a short amount of time. We expect peeks of as much as 1 GB/min, possibly even more. MongoDB stores all incoming data, i.e., the data format is JSON. The JSON contains both large text-blocks (e.g., issue descriptions, source code), as well as numerical metadata (e.g., timestamps, software metrics). The incoming data is raw and directly reflects the information contained in the software repositories. The algorithms that analyze the data may introduce uncertain data. For example, computed labels for bugfixes, bug inducing commits using heuristics, or results of machine learning approaches for intention mining. We will use the data to answer research questions about software development processes.

The DH data set consists of 46 million records, formatted as JSON. The volume amounts to 420 GB. We expect the data to grow by 10-20 percent annually. In this phase of the project we add new data only once a year. In the next phase data will come in monthly cycles or continuously. The data format is called Europeana Data Model (EDM), and is serialized as JSON. Originally it was harvested via OAI-PMH protocol. EDM provides about 130+ data fields, and lots of flexibility, so within this schema the individual records show big heterogeneity. Originally more than 3500 individual institutions have created the data. During their life cycle various users have transformed the data from one format and data schema to another. The research aims to find reliable and less reliable data.

3 The data generated within the project

3.1 Aerodynamic Map Open Data

In this section we describe the content of the initial version of the open data related to the Aerodynamic Map use case of the MIKELANGELO project. Various experiments have been executed using the same set of input cases allowing initial evaluation of benchmarks. The main description of the use case data is provided in Section 2.1 - The Aerodynamic Map Use Case.

3.1.1 Hardware and Software Setup

Experiments have been executed in two different environments. The first environment is one single node, while the other environment is the proper HPC testbed, provided by USTUTT. Descriptions of both environments are presented or referenced below.

3.1.1.1 Single Node

Hardware setup

- CPU: Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
- Memory: 4x4GiB (DIMM DDR3 Synchronous 1600 MHz (0.6 ns)
- Cache description and sizes
    - L1: 128KiB
    - L2: 1MiB
    - L3: 8MiB

Software components:

- Host OS: Ubuntu Trusty (14.04.2 LTS)
- Guest OS: Ubuntu Trusty (14.04.2 LTS)
- Linux kernel: 3.13.0-43-generic
- qemu: 2.0.0+dfsg-2ubuntu1.11 (default package)
- OSv: built from source, exact version within each OSv run in the output, e.g., "OSv v0.20-4-g212f20b"
- OpenFOAM 2.4.0 (default configuration and compilation options used in all environments)

3.1.1.2 HPC testbed

HPC testbed is provided by the MIKELANGELO partner USTUTT. The following is an excerpt from deliverable D5.4 First report on the Integration of sKVM and OSv with HPC [7]. Please refer also to deliverable D2.19 [8] for a detailed description of the HPC testbed.

The software and hardware stack of the test system mirrors, as close as possible, the production HPC environment (see Figure 1).

There is a dedicated front-end server that is accessible via the Internet, this frontend is located on a physical host system. This machine hosts several MIKELANGELO services. These services, deployed in separate virtual machines, are the frontend, the InfluxDB [33], main snap daemon [34] and Jenkins [35] slave. The frontend is the main, for most users also the only, entry point to the entire HPC cluster. It is where the end users are allowed to interact with the batch system (vTorque). InfluxDB virtual machine hosts the time series database where all the snap telemetry data are stored. The main snap daemon controls the data collector plugins and ensures data are stored in the InfluxDB. Finally, the Jenkins slave is an additional service that integrates the HPC testbed with the continuous integration solution deployed by GWDG. Albeit being functional already, this will prove beneficial in the future allowing all use cases to deploy workloads in an automated and well-defined manner.

The testbed further hosts 14 separate physical compute nodes. In addition to these nodes there are four more nodes needed. Two of them are equipped with Mellanox ConnectX-3 VPI cards [9] feasible for RDMA over Converged Ethernet (RoCE) tests, and two nodes are dedicated to kernel building and testing. These four separate nodes are not integrated into the batch-system.

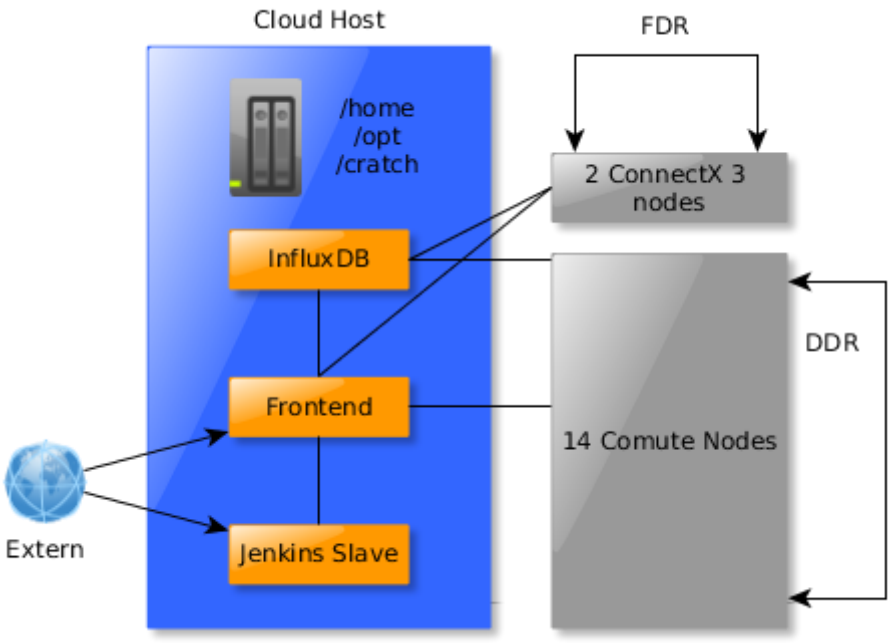

Figure 1: The infrastructure diagram of the HPC testbed as provided by USTUTT.

Each compute node has two sockets, each with 8 Core Intel Xeon CPU X5560 (2.80GHz) and 96GB of ECC DDR3 RAM. They have one 1 Gbit Ethernet network card and a 10 Gbit Infiniband network that supports RDMA.

An NFS server provides shared file-systems for:

- the user's `/home`,
- the global application installation dir `/opt`,
- and a fast workspace mounted under `/scratch`.

3.1.1.3 Cloud Testbed

Even though the aerodynamics maps simulation is a typical HPC use case, cloud environment is also being used for running different evaluations. This testbed, hosted by GWDG, is presented in Section 3.4 Big Data Open Data. The same OpenStack private cloud is used for the evaluation of the MIKELANGELO stack.

3.1.2 Input Cases

OpenFOAM input case is a collection of files organised in a case file structure [10]. Four example cases are currently being used for running initial evaluation of various architectures. All are freely available for download at [11]:

- **mik3d_15min** (small case) is a simplified input case allowing quick tests. This use case is primarily used for functional testing. It is quickly deployed.
- **mik3d_1h** (middle case) is a slightly more complex input case. An intermediate case that can be used in small parallel runs (up to two nodes).
- **mik3d_4h** (large case) is a more complex input case with a much more detailed 3D mesh. Allows evaluation on slightly larger deployments (4-8 nodes).
- **mid3d_largeCase** represents a more realistic input case and is going to be used for complex evaluation of the virtual HPC environment.

The times specified in file names are the ones obtained by running a single OpenFOAM solver on Intel Xeon 2.4 GHz. Note that names small, middle and large are strictly in the context of this evaluation as OpenFOAM cases can be of arbitrary complexity, i.e. even the largest case can be small in most realistic scenarios.

3.1.3 Results

Based on the experiment setup there are results for the single node experiment and for the HPC testbed experiment. The resulting data is stored and treated separately.

3.1.3.1 Single Node

Summary of execution times and other metrics on the single node presented above are available at [12]. Simulations have been executed in various configurations specified in accordance to the specification on the last page of the report [12] (entitled “legend”).

3.1.3.2 HPC Testbed

Results of simulations as executed in the HPC testbed are available at [13]. We have executed simulations for the middle (1h) and the large case (4h), each of them using 1, 2, 4, 8, 16, 32 and 64 cores. The corresponding output files are available at [14] for the middle case and [15] for the large case. The summary of all the execution times is presented in [16].

3.1.4 OpenFOAM Analysis

We have furthermore conducted an analysis of the parallel execution of the OpenFOAM solver (simpleFoam) using the strace Linux command.

First, we analysed the way files are used by individual processes. To this end, we used the following trace command:

```
$ strace -o mpi.log -f -ff -e trace=open,close mpirun -np 2 simpleFoam -parallel
```

The above strace command actually spawns three processes:

- the main mpirun process
- two simpleFoam processes that are started by the mpirun (-np 2 specifies the number of processes)

As a result, all file open and close operations that occurred during the execution of the MPI application are logged.

The following is an excerpt from the log [17] that results from executing the above command. It reveals that processes are not sharing file descriptors of used files. This greatly simplifies the transition from processes to threads inside OSv.

```
mpi.log.17829:open("/home/lemmy/work/openfoam/airFoil2D_05-mpi/system/forceCoeffs",
O_RDONLY) = 11

mpi.log.17829:open("/home/lemmy/work/openfoam/airFoil2D_05-mpi/system/fvSchemes",
O_RDONLY) = 10

mpi.log.17829:open("/home/lemmy/work/openfoam/airFoil2D_05-mpi/system/fvSolution",
O_RDONLY) = 10

mpi.log.17829:open("/home/lemmy/work/openfoam/airFoil2D_05-mpi/system/readFields",
O_RDONLY) = 11
```

Further analysis of used signals and systems calls revealed that the execution does not require any unsupported signal or system calls, apart from the *clone* system call which is used by the mpirun command when starting the two processes.

3.1.5 OpenStack Evaluation Results

For the purposes of validation and evaluation of the OpenStack deployment, an orchestration template [36] has been prepared. This template [37] can be used to provision a cluster of 2 or more instances of the virtual machine image. This image must be uploaded to OpenStack prior to being used by the orchestration engine. The template can be parameterised by the end user as follows:

- image: OSv image to be used for server; must be a valid OpenStack image. Prebuilt image was used for all experiments [38].
- flavor: Flavor to be used for the server; must be a valid OpenStack flavor.
- private_network: Private network to attach server to. This network is required for the virtual machines to be able to communicate with each other.

- public_network: Public network with floating IP addresses. It will be used to assign the master worker node a publicly accessible IP.
- cluster_size: Number of OSv slave instances. Defines the number of worker nodes that are deployed along the master worker node. The overall cluster size is thus cluster_size+1.
- worker_processes: Number of worker processes, i.e. the OpenFOAM processes that are started by the MPI application.

The template will use these parameters and create the following resources automatically:

- Security group required by OpenFOAM processes to be able to communicate between each other.
- OSv cluster as a set of slave worker instances that are used by the master worker instance.
- Master worker instance using the same VM image as the slave virtual machines. The only reason for separating the master from the rest of the cluster is that it is given a different role in the simulation. Master instance is also associated with a floating (public) IP as it needs to be accessible over the internet.
- Finally, the template uses the cloud-init module of OSv [39].

The template uses cloud-init to automatically mount the data over the network share and launch the simulation. The relevant cloud-init section for OSv instances is shown in the following code listing.

```
user_data:
    str_replace:
        params:
            $NP: { get_param: worker_processes }
            $WORKERS: { list_join: [',', get_attr: [osv_cluster, ip]] }
        template: |
            run:
                - POST: /env/WM_PROJECT_DIR
                  val: /openfoam
                - PUT: /app
                  command: /tools/mount-nfs.so nfs://10.254.1.27/export/?uid=0 /data
                - PUT: /app
                  command: /usr/bin/mpirun -np $NP -x WM_PROJECT_DIR=/openfoam -x
MPI_BUFFER_SIZE=20000000 -H localhost,$WORKERS --allow-run-as-root /usr/bin/simpleFoam.so
-case /data/mik3d_1h-$NP -parallel
```

This does the following:

1. Ensures required OpenFOAM environment variable (WM_PROJECT_DIR) is exported and set to the correct directory.
2. Mounts the network data volume using NFS [40].

3. Requests parallel execution of the simpleFoam application (part of OpenFOAM distribution) with parameterized data. The $NP parameter is replaced with the number of worker processes and $WORKERS is replaced with the list of slave worker IPs.

Each configuration was deployed three times using the following command:

```
openstack stack create -e lib/env.yaml -t heat-1.yaml --parameter="image=16552b2f-bc4e-
4f2a-918d-329dd91c1f0f;flavor=kvm.m1.large;cluster_size=3;worker_processes=16"
osv_redirect-cluster4-workers16
```

Raw simulation results are available at [41]. Each run consists of the information about the deployed OpenStack orchestration stack and a complete OpenFOAM log. Report D6.3 discusses these benchmarks in more detail.

3.2 Cloud Bursting Open Data

The data in this section has been taken from report D2.2 First Cloud-Bursting Use Case Implementation Strategy [18]. For additional information about the use case, refer to the aforementioned report D2.2.

3.2.1 Testbed

To evaluate Cassandra versus ScyllaDB behavior under the cloud bursting scenario, we executed a benchmark designed to measure Cassandra and ScyllaDB scale up speed. This section explains the setup of the benchmark, followed by the initial benchmark results.

- Cassandra nodes: EC2 i2.8xlarge (Amazon EC2 instance)
- Loaders nodes: EC2 i2.8xlarge (as above)
- Cassandra version: 2.1.15
- Linux, Ubuntu 16.04 for Cassandra, latest Centos for ScyllaDB

Our plan is to use the same setup later in the project to compare new ScyllaDB updates to Cassandra.

3.2.2 Test Execution

The loader starts a process cassandra-stress [20] “write” workload. We start from a Cassandra cluster consisting of 3 nodes initialized with 120 GB of data, adding a new node to the cluster at least 2 minutes apart (more on this in report D2.4 [18]), until a total of 6 nodes is reached.

3.2.3 Benchmark Results

Figure 2 presents the throughput, as sum of what was measured by the loaders during the test. We see the performance gain once each node has been finished being inserted.

Axes of the following charts are as follows

- X axis is the time.
- Y axis is either the number of operations (requests) per second for Figure 2 or latency in microseconds for Figure 3 and Figure 4.

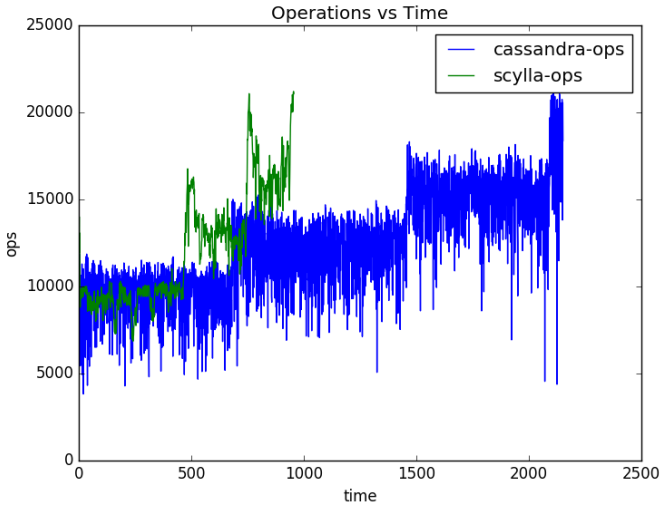

Figure 2: Scale Out Benchmark: Inserts per second

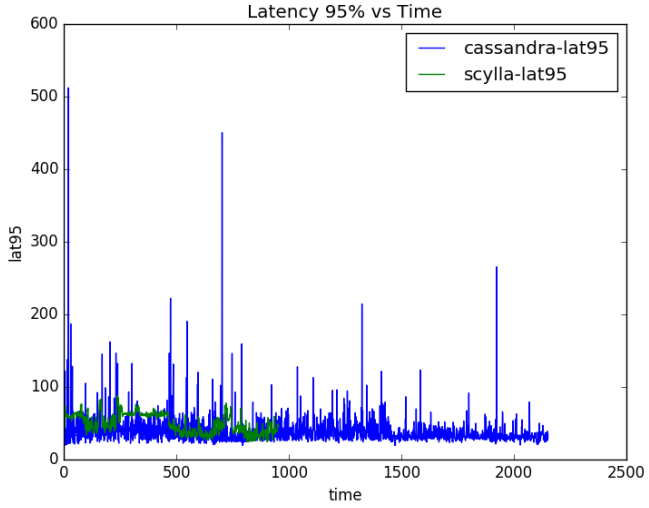

Figure 3: Scale out benchmark: latency 95%

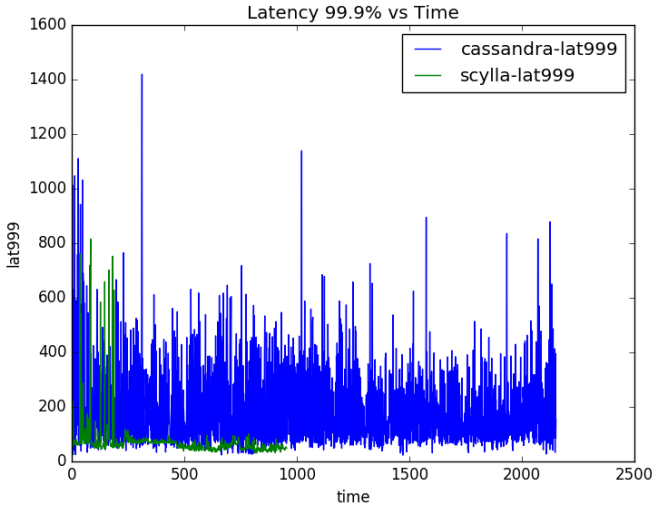

Figure 4: Scale out benchmark: 0.999% Latency

Figure 3 presents mean latency of Cassandra insert request for the same test in milliseconds, and Figure 4 shows latency of the worst 0.001% of the requests.

Full results and raw data are available at [21]. Also refer to section on the cloud bursting use case in report D6.3 First report on the Use Case Implementations [29].

3.3 Cancellous bone simulation Open Data

This section describes the content of the initial version of the open data related to the Cancellous Bones use case of the MIKELANGELO project. Various experiments have been executed using the same set of input cases allowing initial evaluation of benchmarks.

3.3.1 Hardware and Software Setup

HPC testbed: In total 14 compute nodes, each compute node has two sockets, each with 8 Core Intel Xeon CPU X5560 (2.80GHz)and 96GB of ECC DDR3 RAM. All these tests were executed using the Cancellous Bone Simulator (SVN revision 429[1]).

Please refer to Section 3.1.1.2 - HPC Testbed for a detailed description of the HPC testbed.

[1] The Cancellous Bones Simulator has not been released as open source yet, but it will eventually become available which is the reason for publishing the actual revision number.

3.3.2 Input Data

The file format PureDat, used in the bones application is described in Section 2.3 The Cancellous Bones use case. Input Data is available in the data folder of the compressed file [23].

3.3.3 Configuration Script

The application for the Cancellous Bones simulation is not open sourced at the time of writing this report. The configuration script, available at [24], provides an overview of the settings used for the execution of the simulation, for the current version of the simulation application.

3.3.4 Results

For the initial runs, the inputs to the Cancellous Bones use case simulations were shrunk to the smaller partitions of the full input data versions. The actual meaning of the input and output data is insignificant, as the meaning of these experiments is to provide the benchmarks of the actual performance and to test execution of the use case on the MIKELANGELO platform. To verify the correctness of the executed code, we verified the outputs and found them to be correct. The output of each simulation run is stored in a subfolder `regression_check_mpi_xx_yy` (xx = date, yy=month) [23]. Input parameters used for the simulation (copied from the configuration script) are available in `struct_process.input`. `regression_check_mpi2.mon` specifies the MPI rank and the `regression_check_mpi2.log` shows the results of the run.

HPC Testbed: We have executed simulations on one node bare metal and on one node with a VM. Results of simulations as executed in the HPC testbed are available in [25]. Refer to deliverable D2.1 [22] for detailed hardware and software specifications.

3.4 Big Data Open Data

In this section we present generated data that relates to the big data use case. First, we present the generated data from year 1 and then we summarize the generated data from year 2. The experiments from year 1 have not been repeated. Thus, there is no update on that data in year 2. We have not repeated the experiments from year 1 as no new features that would have provided performance gains were available. OSv has been tested for use with HDFS in year 2. The results suggest that using HDFS with OSv in the big data use case would lead to a significant slow down. The results of this test are described below in Section 3.4.2.

Additionally, we ran experiments to analyze the impact of a variety of parameters on the performance of a virtualized big data cluster. The resulting data of the study has undergone preliminary analysis. A full analysis will be performed later on in this project. However, the resulting data from that analysis has been published and is summarized in Section 3.4.2.

3.4.1 Year 1

The purpose of this section is to describe the content of the initial version of the open data related to the Big Data use case of the MIKELANGELO project. The data in this document has been taken from report D2.7 First Virtualised Big Data use case Implementation Strategy [26]. For additional information about the use case, refer to the aforementioned report D2.7.

The Big Data use case has served as the main source of the requirements for the implementation and deployment of a continuous experimentation system (Scotty) designed specifically for running experiments in a controlled environment. The main achievements of Scotty are the speed-up of the complete research cycle, reproducibility of experimental results and falsifiability. Report D5.2 [42] introduces Scotty, the underlying motivation that lead us to the implementation and presents some preliminary findings.

Our baseline measurements come from two similar sets of experiments. The first set of experiments runs HiBench [27] directly on three hosts. The second set of experiments runs HiBench on the same hosts, but inside VMs running on top of KVM. The raw version of data is available at [28].

Figure 5 shows the setup for the direct deployment of HiBench on the three hosts. Hadoop, HDFS, Spark, and HiBench are installed directly on the Ubuntu host system. The first node acts as a master node and worker node at the same time. First, for each of the micro-workloads data is generated and then stored in HDFS. Then the algorithms for each of the workloads are executed on all three hosts. Hadoop uses remote procedure calls to distribute work among the workers, which then run independently. Each of the worker nodes in turn runs multi-threaded algorithms.

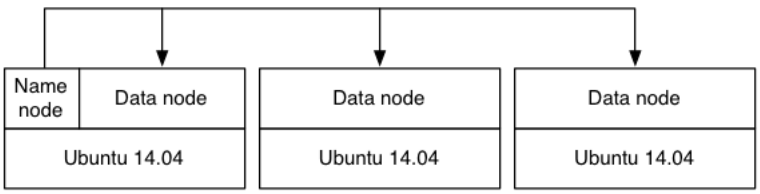

Figure 5: Deployment of HiBench directly on the hosts.

Figure 6 shows the benchmarking setup with VMs. Each host runs three VMs with Ubuntu as guest OS. The first host runs a relatively small VM, as specified in Table 3, for the master node. The worker VMs on all hosts have the same configuration. The configuration of both types of VMs is described in Table 1. In the second setup we keep the problem size and general benchmarking configuration the same as for the host-based setup.

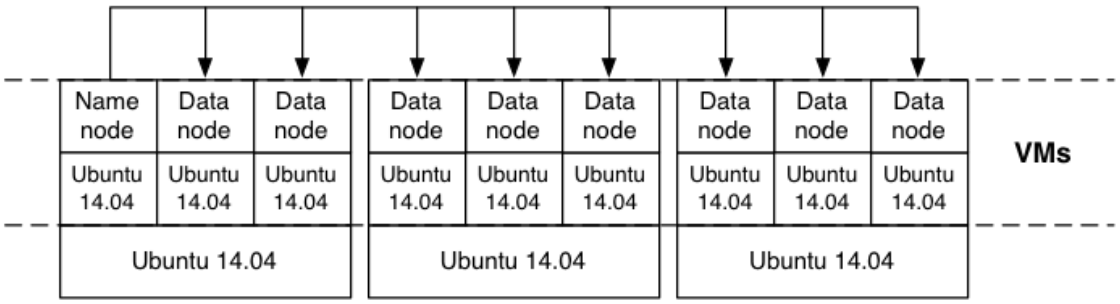

Figure 6: Deployment of HiBench in VMs

Table 1: VM configuration for benchmarking

Property	Name Node	Data Node
Quantity	1	8
Memory	8 GB	40 GB
Storage	40GB	40GB

The results of running HiBench in the host-based experiment are shown in Table 2 below. Some of those tests did not complete due to problems in configuration and bugs in the software. Table 3 shows analogous results for the experiment using VMs to execute the workload.

Table 2: Numerical results from host-based benchmarking.

Workload	Input Data Size	Duration (s)	Throughput (B/s)	Throughput per node (B/s)
HadoopAggregation	37276711	35.413	1052627	350875
JavaSparkAggregation	37276833	55.806	667971	222657
ScalaSparkAggregation	37276833	55.769	668414	222804

Workload	Input Data Size	Duration (s)	Throughput (B/s)	Throughput per node (B/s)
PythonSparkAggregation	37276833	56.960	654438	218146
HadoopJoin	1000350	63.467	15761	5253
JavaSparkJoin	190683757	64.727	2945969	981989
ScalaSparkJoin	195929471	64.204	3051670	1017223
PythonSparkJoin	195929471	65.539	2989509	996503
HadoopKmeans	4016371691	216.879	18518951	6172983
JavaSparkKmeans	4016371691	175.469	22889351	7629783
ScalaSparkKmeans	4016371691	119.113	33719003	11239667
PythonSparkKmeans	4016371691	225.720	17793601	5931200
HadoopPagerank	259928115	214.100	1214050	404683
JavaSparkPagerank	259928115	97.275	2672095	890698
ScalaSparkPagerank	259928115	93.249	2787462	929154
PythonSparkPagerank	259928115	108.506	2395518	798506
HadoopScan	186646828	44.345	4208971	1402990
ScalaSparkScan	186647745	54.391	3431592	1143864
PythonSparkScan	186647745	57.390	3252269	1084089
HadoopSleep	0	18.149	0	0
JavaSparkSleep	0	124.422	0	0
ScalaSparkSleep	0	124.418	0	0
PythonSparkSleep	0	126.125	0	0
HadoopSort	328492566	20.117	16329103	5443034
JavaSparkSort	328492566	38.614	8507084	2835694
ScalaSparkSort	328492566	38.899	8444756	2814918

Workload	Input Data Size	Duration (s)	Throughput (B/s)	Throughput per node (B/s)
PythonSparkSort	328492566	39.589	8297571	2765857
HadoopWordcount	2204463831	36.156	60970899	20323633
JavaSparkWordcount	2204463831	45.904	48023349	16007783
ScalaSparkWordcount	2204463831	42.498	51872178	17290726
PythonSparkWordcount	2204463831	55.586	39658616	13219538
HadoopBayes	575056284	43.703	13158279	4386093
JavaSparkBayes	575056284	153.322	3750644	1250214
ScalaSparkBayes	575056284	51.662	11131127	3710375

Table 3: Numerical results from VM-based benchmarking.

Workload	Input Data Size	Duration (s)	Throughput (B/s)	Throughput per node (B/s)
HadoopAggregation	37276711	78.142	477038	59629
JavaSparkAggregation	37276711	60.269	618507	77313
ScalaSparkAggregation	37276711	63.109	590674	73834
PythonSparkAggregation	37276711	60.709	614024	76753
HadoopJoin	1000350	110.542	9049	1131
JavaSparkJoin	188832410	74.725	2527031	315878
ScalaSparkJoin	194078124	74.137	2617830	327228
PythonSparkJoin	194078124	74.402	2608506	326063
HadoopKmeans	4016371702	617.808	6501003	928714
JavaSparkKmeans	4016371702	323.785	12404440	1550555
ScalaSparkKmeans	4016371702	107.479	37368897	4671112

Workload	Input Data Size	Duration (s)	Throughput (B/s)	Throughput per node (B/s)
PythonSparkKmeans	4016371702	507.817	7909092	988636
HadoopPagerank	259928115	378.638	686481	85810
JavaSparkPagerank	259928115	160.677	1617705	202213
ScalaSparkPagerank	259928115	202.571	1283145	160393
PythonSparkPagerank	259928115	251.692	1032722	147531
HadoopScan	184795601	93.462	1977227	247153
ScalaSparkScan	184796438	58.220	3386534	423316
PythonSparkScan	184796438	57.726	3201268	400158
HadoopSleep	0	24.694	0	0
JavaSparkSleep	0	110.888	0	0
ScalaSparkSleep	0	111.025	0	0
PythonSparkSleep	0	127.452	0	0
HadoopSort	328493084	36.558	8985532	1497588
JavaSparkSort	328493084	30.920	10623967	1770661
ScalaSparkSort	328493084	38.008	8642735	1440455
PythonSparkSort	328493084	32.069	10243321	2560830
HadoopWordcount	2204473443	105.248	20945513	2618189
JavaSparkWordcount	2204473443	52.221	42214309	5276788
ScalaSparkWordcount	2204473443	39.189	56252352	7031544
PythonSparkWordcount	2204473443	77.069	28603893	3575486
HadoopTerasort	3200000000	127.281	25141222	3142652
JavaSparkTerasort	3200000000	64.074	49942254	6242781
ScalaSparkTerasort	3200000000	62.601	51117394	6389674

Figure 7 and Figure 8 visualise the duration and aggregate I/O throughput of each micro workload. These bar charts compare the durations for the same experiment as run on the directly on the host and in VMs.

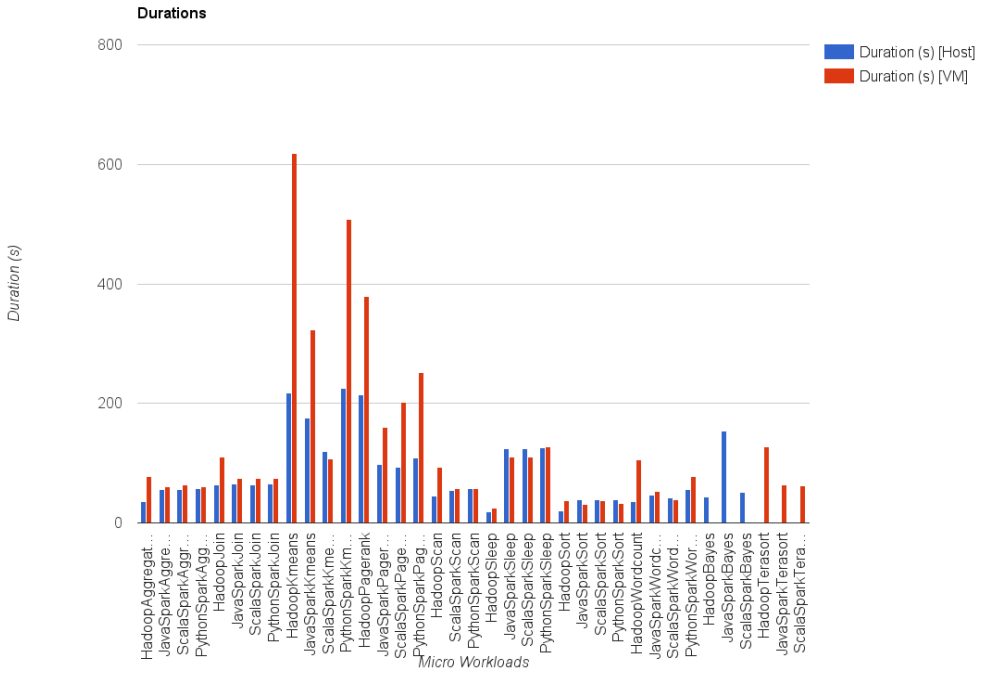

Figure 7: Comparison of benchmarking duration between host-based and VM-based runs.

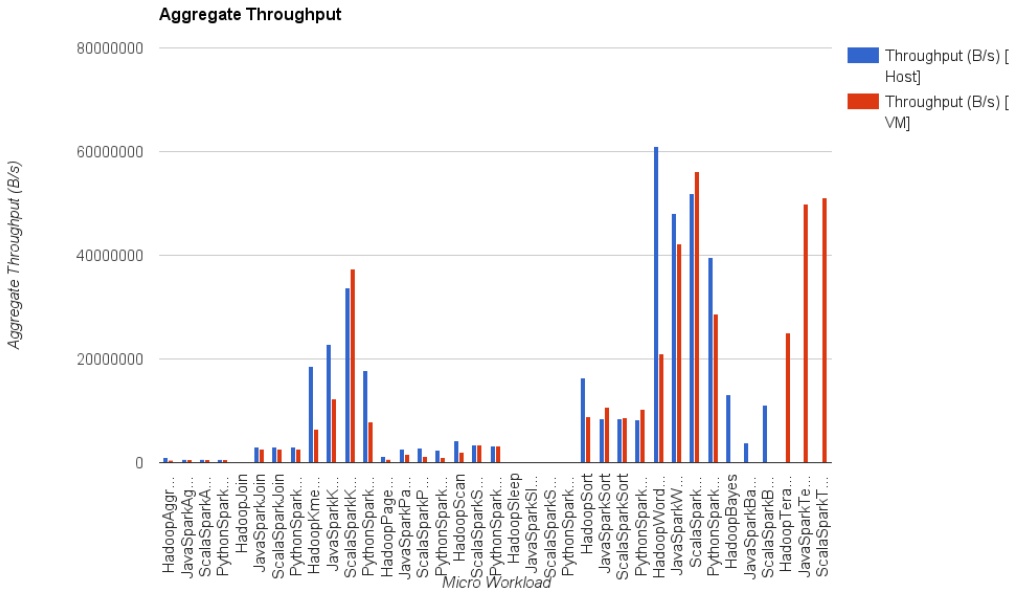

Figure 8: Comparison of aggregate throughput between host-based and VM-based runs.

3.4.2 Year 2

To gauge the expected problems in using OSv, we have conducted a study to ***deploy HDFS on OSv***. In the course of these experiments, we have discovered that running OSv in HDFS leads to significant slow-downs. These slow downs are in the range of x5-x10. The data is publicly available as a spreadsheet in our OpenData repository [30]. The experiments are described in more detail in D6.3 [29]. Here we briefly summarize the experiment, as shown below in Figure 9.

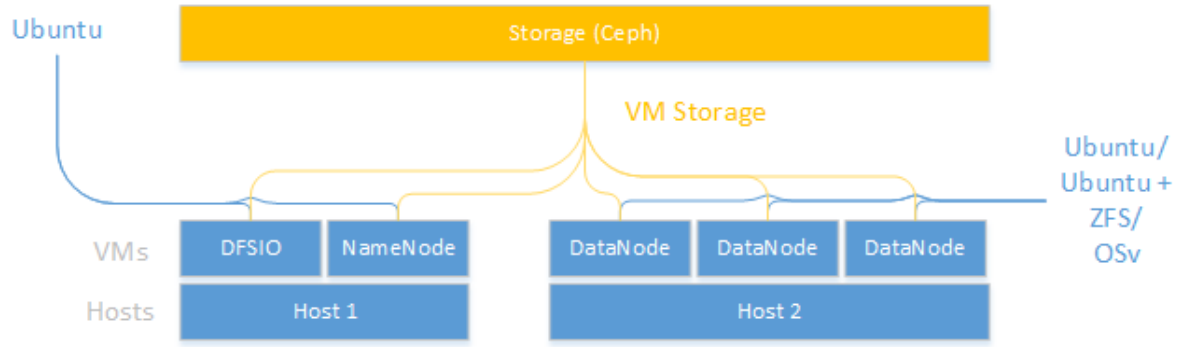

Figure 9: The Big Data experiment set up.

The experiment evaluated the performance of HDFS in VMs with DFSIO, the standard benchmarking tool for HDFS. The evaluation has taken into account three different conditions. In all cases we have controlled for effects of sizing, distribution, and configuration of DFSIO and HDFS. The first experiment ran HDFS in Ubuntu to establish a baseline. The second experiment ran HDFS in OSv. The third experiment ran HDFS in Ubuntu with ZFS. ZFS has been tried out in Ubuntu to control for the fact that OSv uses ZFS by default. Thus, we wanted to assess how much of the expected performance gap between OSv and Ubuntu results from the use of ZFS. ZFS in general should perform better than the default ext-filesystem in Ubuntu. The experimental setup is shown in Figure 9. We have repeated the experiments multiple times and recorded the data to control for statistical variations.

The collected data consists of a spreadsheet that records the results. For each setup of software and VMs we ran 4-5 experiments each with a different configuration of DFSIO. We primarily changed the size and quantity of uploaded files. We ran each of those experiments four times to control for stochastic variations. The document contains multiple sheets, one for each setup. For initial cases, we have performed ANOVA. However, in this analysis and when reviewing the remaining results the performance gap between OSv and Ubuntu was obvious - in our experiment set-up, OSv was slower.

To gauge the impact of parameters of a virtual big data platform on the performance, we have conducted a review of the literature and we have performed a study ourselves. The review of the literature did not provide conclusive results and it was scattered across various experiments with varying setups. We have extracted hypotheses from the literature review and analyzed them in our experiments. These experiments are described in more detail in D6.3 [29]. The results of this study are available in our OpenData archive [31]. The data consists of input data for the experimental setup and for the workload. The experimental setup comprises the configuration of the hardware, virtual infrastructure, host OS, and application layer. The workload parameters comprise all available parameters in HiBench [27] - a big data benchmarking suite. The recorded data furthermore consists of the results of each HiBench run and of recorded time series. The HiBench results contain mostly information about execution times and throughput of the big data cluster. These data are summaries provided by HiBench. The time series data comprises metrics across the whole deployment from hardware to the application layer. The time series have first been collected by snap and stored by InfluxDB. However, to provide a sustainable data format for other researchers, we provided the recorded time series as CSV files.

3.5 Node Meta-Data Collector

To document the status of the nodes inside a testbed at HLRS, a meta data collector was implemented. The main focus for this software is to provide additional data about the experiments. It may also serve well in other cases, for example as a tool to monitor software drift in large cluster deployments. After an experiment, not only the data and the experiment itself is of interest, but the surroundings of the experiment is important as well. For infrastructure test, the following things are of interest:

- Installed software and their version
- Processes running on the host
- CPU information like type, cache size and architecture
- The environment variables (for Linux based executions)

The above parameters can make a huge difference in the results. The documentation of this surroundings are collected automatically by the node-metadata-collector. The collector produces a JSON document for each node which contains all the necessary information. Following the experiment execution, these individual documents are merged into a single document representing the state of the nodes that were used during the simulation.

The following is an example of the data produced by the metadata collector for a single compute node:


```
{
  "nodeName": {
   "processes": [
     {
        "pid": 1,
        "name": "systemd"
      },
      {
       [...]
      }
    ]
   "packages": {
      "python-libxml2": "[python-libxml2=2.9.4+dfsg1-2, second
version]"
       [...]
   },
    "cpu": {
      "count": 4
      [...]
   },
    "env": {
     "LANG": "en_US.UTF-8"
     [...]
   },
    "time": "2016-12-12 11:07:02.199288"
  }
   "vTorque": {
    "8e458f5"
  }
   "mpi": {
    "Path" : "/opt/mpi"
    "version" : "mpirun (Open MPI) 1.6.5"
  }
}
```

The JSON structures allows a simple data processing and storing, it can be expanded with additional information or stored in a noSQL database like MongoDB.

An example of the data collected on a subset of the HPC cluster nodes hosted by USTUTT is available at [43]. The metadata collector is available in MIKELANGELO's GitHub project repository [44].

3.6 SCAM Open Data

This section describes the open data provided by the SCAM security module developed within MIKELANGELO. This data serves as the basis for the experimental results described in

deliverable D3.2 [45]. We refer the reader to this deliverable for a full description of the techniques and methodology used to generate this data. We distinguish between two types of data: (1) the data produced by the monitoring module, and (2) the data gathered to evaluate the performance of the noisification module. We first describe the overall architecture of the BGU testbed.

As can be seen in Figure 10, our system consists of a target TLS server co-located with the attacker module (described in detail in deliverable D3.4). The Host OS has a monitoring and noisification processes running in as user-space applications. A remote TLS client connects to the TLS server to trigger the decryption procedure on the server side.

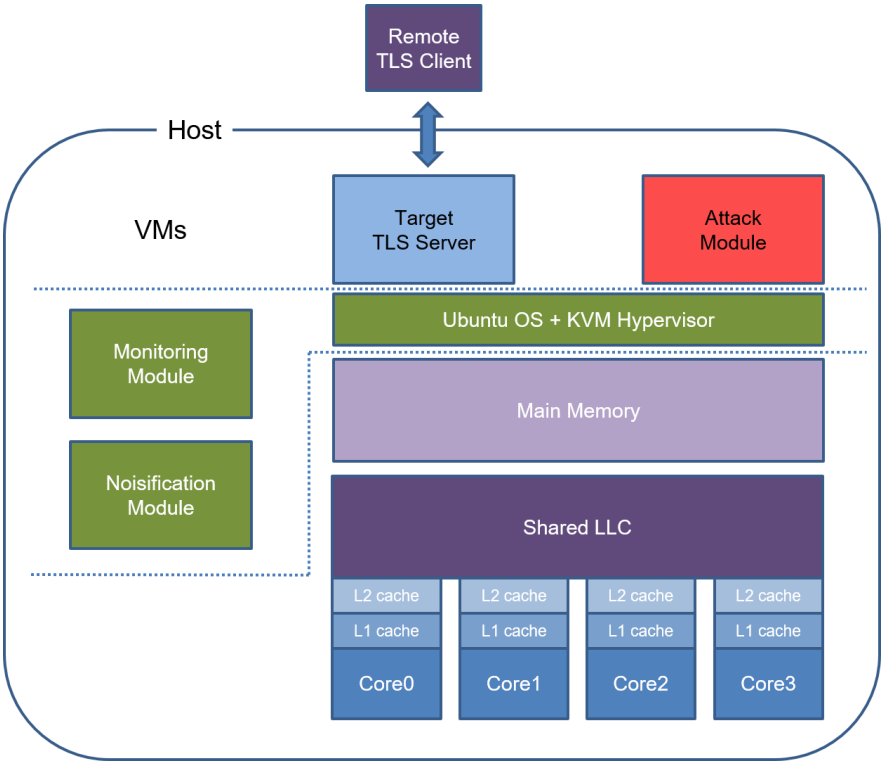

Figure 10: System setup for SCAM validation and data

The system settings are as follows:

Host: 4-core Intel intel i7 6700hq processor 16GB RAM 6144KB LLC, 8192 sets, 8 slices, 12-way associative OS Kernel: 4.4.0-42	**Attacker VM:** 2 virtual cores, 4GB RAM OS Kernel: 3.19.0-49 **TLS Server VM:** 1 virtual core, 4GB RAM: OS Kernel: 3.19.0-49

3.6.1 Monitoring

There are two types of files provided in the repository, files that present the measurements of the monitoring module and files with information on the performance penalty of monitoring.

There are three measurement files: bm_server.csv, bm_attacker.csv and merged_in.csv. The first, bm_server.csv, is the "Benchmark" for the server, measuring its hit and miss rates when running as the only application in the system (aside from the monitoring module). In bm_attackere.csv the measurements are of the attacker running without any other application. The merged_in.csv file includes measurements of both processes running in parallel to one another. Each row in a measurement file represents a measurement of the Last Level Cache (LLC) counters. Measurements are taken every 100 μs. The fields in the first two files are:

- **PAPI_L3_TCA**: the total number of accesses the process made to the level 3 cache, i.e the LLC, since the last measurement.
- **PAPI_L3_TCM**: the number of LLC misses since the last measurement.

The fields in the last two files are:

- **PAPI_L3_TCA**: the total number of accesses the attacker made to the LLC since the last measurement.
- **PAPI_L3_TCM**: the total number of cache misses the attacker had in the LLC since the last measurement.
- **PAPI_L3_TCA(server)**: the total number of accesses the server made to the LLC since the last measurement.
- **PAPI_L3_TCM(server)**: the total number of cache misses the server had in the LLC since the last measurement.

There are three performance files: bm_server_per.csv, server_per.csv and monitor_per.csv. The first measures the number of instructions, machine cycles, cache accesses and misses that the server performs while running without monitoring, the second presents the same data with active monitoring and the third measures the same data for the monitoring module itself. Each row in a performance file represents a single measurement, with measurements taken every 100 μs. The fields in each file are:

- **PAPI_TOT_CYC**: counting the number of cycles since the last measurement (as given by the PAPI interface of the PMU).
- **PAPI_TOT_INS**: counting the number of instructions since the last measurement (as given by the PAPI interface of the PMU).
- **PAPI_L3_TCA**: number of LLC accesses since last measurement.

- **PAPI_L3_TCM**: number of LLC misses since last measurement.

3.6.2 Noisification

The data provided in the repository consists of 5 CSV files with the statistics of results. Files are named: no_noise.csv, k1.csv, k2.csv, and k5.csv, corresponding to the experiments performed without noisification, and with noisification where intensity parameter k equals 1, 2, and 5, respectively.

Each file has the following fields:

- **Test Num**: serial ID of the experiment
- **Mit_set_found**: number of sets found by the noisification cache mapping phase
- **Mit_Pot_sets_found**: number of sets found by the noisification target reconnaissance phase
- **Attacker_sets_found**: number of sets found by the attacker's cache mapping phase
- **Attacked_set_num**: iteration number for the number of sets examined by the attacker
- **Attacker_Samples_Found_for_this_set**: number of valid samples found by the attacker
- **Num_of_wrong_bits**: number of wrong bits in the key extracted by the attacker (-1 if insufficient samples)

For more information, we refer the reader to the README.txt file in the MIKELANGELO Opendata repository [2].

4 Conclusions

This document is a cover document for the MIKELANGELO data management activities and reports on the MIKELANGELO compliance with the Open Data Pilot programme.

We described all the experiments and data collected in the Year 2 of the MIKELANGELO project, ranging from the use case and input data descriptions, to the ways of generating data (i.e., scripts, environment descriptions), to the actual data (i.e., measurements, results) and even meta-data collectors. For each of the aforementioned cases, we provided the location of the data. The data collected from all of the use cases, along with experiment descriptions, allows external parties to analyse and reproduce the experiments performed in MIKELANGELO.

5 References and Applicable Documents

[1] Deliverable D7.17 - The first update on the Data Management Plan: https://www.mikelangelo-project.eu/wp-content/uploads/2016/06/MIKELANGELO-WP7.17-XLAB-v2.0.pdf

[2] The OwnCloud instance hosting open data base address: http://opendata.mikelangelo-project.eu/ with specific URL for the open data: http://opendata.mikelangelo-project.eu/public.php?service=files&t=b19368219ba31f377561a3113e332956

[3] MIKELANGELO website - http://www.mikelangelo-project.eu/

[4] http://www.openfoam.com/

[5] https://cassandra.apache.org/

[6] http://www.scylladb.com/

[7] D5.4 First report on the Integration of sKVM and OSv with HPC, https://www.mikelangelo-project.eu/deliverables/

[8] http://www.mikelangelo-project.eu/deliverables/deliverable-d2-19/

[9] Mellanox ConnectX-3 EN 10/40/56GbE Network Interface Cards, http://www.mellanox.com/page/products_dyn?product_family=127

[10] OpenFOAM Input Case, http://cfd.direct/openfoam/user-guide/cases/#x16-920004

[11] OpenFOAM input cases for Aerodynamics Use case, http://opendata.mikelangelo-project.eu/public.php?service=files&t=13b3f1adeba018c12a0a4b88e2a8aec5

[12] Summary of execution times for single node, http://opendata.mikelangelo-project.eu/public.php?service=files&t=5f8e8a731a511d9861130b776bc3b6d0

[13] OpenFOAM output results for Aerodynamics Use case, http://opendata.mikelangelo-project.eu/public.php?service=files&t=f9cb158ac3d8603726aba35708276579

[14] OpenFOAM 1h outputs, http://opendata.mikelangelo-project.eu/public.php?service=files&t=fbfffb0c7aeef055703c296ebf89eba7

[15] OpenFOAM 4h outputs, http://opendata.mikelangelo-project.eu/public.php?service=files&t=97a9cb81bd7dd1a7af24765895029d40

[16] Summary of execution times for HPC testbed, http://opendata.mikelangelo-project.eu/public.php?service=files&t=9441e9a631e120a20bac7424073d3318

[17] OpenFOAM signal trace, http://opendata.mikelangelo-project.eu/public.php?service=files&t=80c71ba06bf6486e85c69a7b117b2524

[18] First Cloud-Bursting Use Case Implementation Strategy, http://www.mikelangelo-project.eu/deliverables/deliverable-d2-4/

[19] https://github.com/scylladb/cassandra-test-and-deploy

[20] http://docs.datastax.com/en/cassandra/2.1/cassandra/tools/toolsCStress_t.html

[21] http://opendata.mikelangelo-project.eu/public.php?service=files&t=d44cf89d90f8f0e1d35359d6e14826bb

[22] D2.1 First Cancellous bone simulation Use Case Implementation strategy, https://www.mikelangelo-project.eu/deliverables/deliverable-d2-1/
[23] Cancellous bone simulation test input Data, http://opendata.mikelangelo-project.eu/public.php?service=files&t=d60276e6ef066d9979f58535b043120f
[24] Input case configuration script, http://opendata.mikelangelo-project.eu/public.php?service=files&t=13a0a0a34cd8f505267e35b31734e729
[25] Results of the initial measurement, http://opendata.mikelangelo-project.eu/public.php?service=files&t=be4360843d268e7f3e6e973c71aa10c6
[26] First Cloud-Bursting Use Case Implementation Strategy, https://www.mikelangelo-project.eu/deliverables/deliverable-d2-7/
[27] https://github.com/intel-hadoop/HiBench
[28] http://opendata.mikelangelo-project.eu/public.php?service=files&t=9f7ad49004e9b1806de289af6518a57f
[29] D6.3 First report on the Use Case Implementations http://www.mikelangelo-project.eu/deliverables/deliverable-d6-3/
[30] OSv HDFS Experiments http://opendata.mikelangelo-project.eu/public.php?service=files&t=4cbda1ba6495445de73f9ef684f4af1f
[31] Big Data Platform Study http://opendata.mikelangelo-project.eu/public.php?service=files&t=d33994234df60ae4b428ec44021e06aa
[32] OpenFOAM large input case with rotating propellers. http://opendata.mikelangelo-project.eu/public.php?service=files&t=f92d2d25282bb1b3a82c3f271d42eb74
[33] InfluxDB Homepage: https://www.influxdata.com
[34] Snap Telemetry Homepage: http://snap-telemetry.io
[35] Jenkins Homepage: https://jenkins.io
[36] Heat Orchestration Template (HOT) Guide http://docs.openstack.org/developer/heat/template_guide/hot_guide.html
[37] OpenFOAM Orchestration Stack Template, http://opendata.mikelangelo-project.eu/public.php?service=files&t=5e6842a8e2e6ad66b98c593bed8ed749
[38] OSv OpenFOAM Image, http://opendata.mikelangelo-project.eu/public.php?service=files&t=56c4f50b1411b30593a47d0306840232
[39] OSv - Guest Operating System - Intermediate Version http://www.mikelangelo-project.eu/deliverables/deliverable-d4-5/
[40] Network File System https://en.wikipedia.org/wiki/Network_File_System
[41] Raw OpenFOAM simulation results for the cloud testbed http://opendata.mikelangelo-project.eu/public.php?service=files&t=ca80acb5aedcf8f7c877bd65c93b19f0

[42] D5.2 Intermediate report on the integration of sKVM and OSv with Cloud and HPC http://www.mikelangelo-project.eu/deliverables/deliverable-d5-2/

[43] Metadata of the HPC testbed, http://opendata.mikelangelo-project.eu/public.php?service=files&t=af3e160219a950038cf2fb83298e2e03

[44] Node metadata collector Github repository https://github.com/mikelangelo-project/node-metadata-collector

[45] Report D3.2 Intermediate Super KVM - Fast virtual I/O hypervisor http://www.mikelangelo-project.eu/deliverables/deliverable-d3-2/