MIKELANGELO

D7.17

The first update on the Data Management Plan

Workpackage	7	Exploitation, Dissemination, Communication and Collaboration
Author(s)	Daniel Vladušič	XLAB
Reviewer	Peter Chronz	GWDG
Reviewer	Holm Rauchfuss	HUA
Dissemination Level	PU	

Date	Author	Comments	Version	Status
2015-12-01	Daniel Vladušič	Initial draft	V0.0	Draft
2015-12-18	Daniel Vladušič, Gregor Berginc	Document Ready for review	V1.0	Review
2015-12-28	Daniel Vladušič, Gregor Berginc	Document ready for submission	V2.0	Final

Executive Summary

The purpose of this deliverable is to describe the data from the experiments performed in the MIKELANGELO project. The data collected and described in this deliverable consists of the actual data (i.e., measurements, results) and the ways of generating data (i.e., scripts, environment descriptions). For the sake of completeness, this document contains brief descriptions of the use cases and their input data. For each of the use cases, the experiments are briefly described and the locations of the data are provided. 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.

This document is an update of the D7.16 - The initial data management plan deliverable and serves as the cover document for all the data provided by the MIKELANGELO project for year 1.

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 D7.16 - The initial Data Management plan[1]. In D7.16 we provided the framework of the Data Management Plan (DMP), containing verbatim copies from various guidelines and our MIKELANGELO Grant Agreement, and providing the processes, requirements for the DMP. We also provided the known descriptions of the use case data within MIKELANGELO and finally, the locations of the publicly available data, along with the backup description.

The first update of D7.16 is this document, the first update on the Data Management Plan. It focuses on the data that has been generated by the project during the execution of experiments in year 1.

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 we briefly present the data to be used and provided by MIKELANGELO - the descriptions are taken from D7.16. In the second part of the deliverable, the actual experiment and data descriptions for significant experiments performed within MIKELANGELO are provided. As this is is a living document which is updated on yearly basis, this first iteration provides the data for the experiments performed during year 1. 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 original descriptions are part of deliverable D7.16 (please see Section 3.1 The data used within the use-cases).

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 initial case we studied a family of proprietary 2D airfoils, i.e. planar contours defined by consecutive points in the plane, taken from the company library of airfoils. The chosen airfoils are used for propeller and wing design of Pipistrel's aircraft. The follow-up of this methodology is a concrete example study of a complete aircraft geometry. This example study consists of a 3D CAD model of external surfaces of the aircraft, taken from the technical database of the company.

2.2 The Cloud Bursting Use Case

The Cloud Bursting use case involves installing Cassandra[5] and ScyllaDB[6], and then abruptly increasing the request load and seeing how soon the setup can adjust to handle this huge load (i.e., typical cloud bursting scenario). With this use case, we do not use any sensitive data - the data which Cassandra returns from the queries is immaterial and can be 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 input file format as well as 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.

The data for this use case will be gathered through the services required by the partners within the Max Planck society. Other data useful for MIKELANGELO project will be generated using benchmarks for various components of the MIKELANGELO technology stack.

3 The data generated within the 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. Further, there are 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.

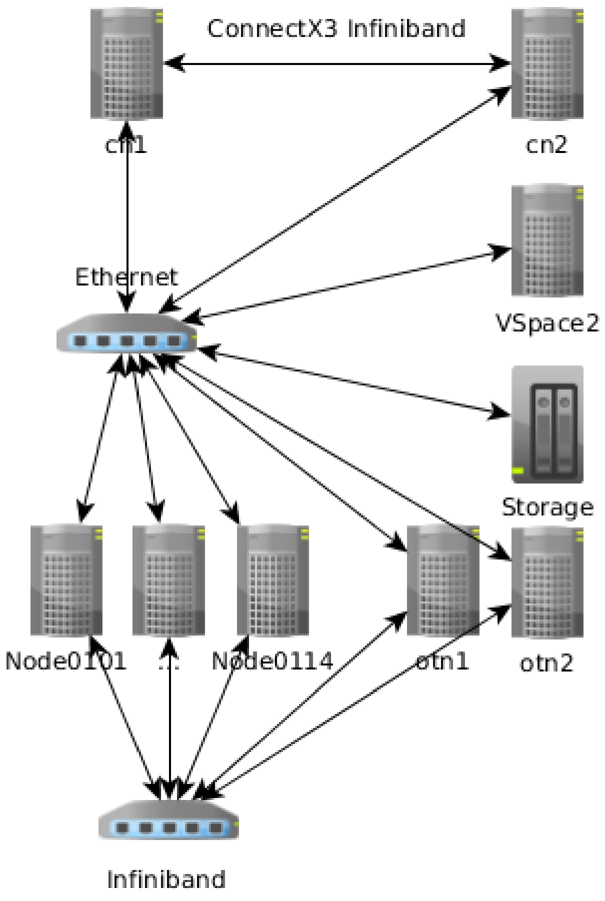

Figure 1: The infrastructure diagram of the HPC testbed as provided by USTUTT.

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.

A NFS server provides shared file-systems for:

- the user’s `/home`,
- the global application installation dir `/opt`,

and a fast workspace mounted under `/scratch`.

3.1.2 Input Cases

OpenFOAM input case is a collection of files organised in a case file structure [10]. Three example cases have been prepared 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.
- **mik3d_1h** (middle case) is a slightly more complex input case.
- **mik3d_4h** (large case) is a more complex input case with much more detailed 3D mesh.

The times specified in file names are the ones obtained by running 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.

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 (entitled “legend”).

3.1.3.2 HPC Testbed

Results of simulations as executed in 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 middle case and [15] for 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 -case . -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.

3.2 Cloud Bursting Open Data

The data in this section has been taken from report D2.4 First Cloud-Bursting Use Case Implementation Strategy [18]. For additional information about the use case, refer to the aforementioned report D2.4.

3.2.1 Testbed

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

- Cassandra nodes: EC2 m3.large (each node has 2 vcpus, and 7.5 GB of RAM)
- Loaders nodes: EC2 c3.8xlarge
- Cassandra version: 2.1.8

- Linux, Ubuntu 14.04

Our plan is to use the same setup later in the project to compare new Cassandra updates as well as Scylla database.

Our own grown Ansible framework [19] takes care of launching the instances, setting up the Cassandra cluster, running the loaders and collect the results.

3.2.2 Test Execution

Each of the loaders (2 in this case) runs a process cassandra-stress [20]'s "write" workload. We start from a Cassandra cluster consisting of 2 nodes, adding a new node to the cluster at least 2 minutes apart (more on this in report D2.4 [18]), until a total of 10 nodes is reached.

The following commands were used to run the test:

```
./ec2-setup-cassandra.sh -e "cluster_nodes=2" -e "instance_type=m3.large"

./ec2-setup-loadgen.sh -e "load_nodes=2" -e "instance_type=c3.8xlarge"

./ec2-stress.sh 1 -e "load_name=init.v1" -e "server=Cassandra" -e
"stress_options='-errors ignore'" -e "command_options=duration=3m" -e
"threads=750" -e "clean_data=true"

./ec2-add-node-to-cluster.sh -e "server=Cassandra" -e "cluster_nodes=8" -e
"stopped=true" -e "instance_type=m3.large"

./ec2-stress.sh 1 -e "load_name=scale_from_2_to_10.m3.large.v1" -e
"server=Cassandra" -e "stress_options='-errors ignore'" -e
"command_options=duration=45m" -e "threads=750" -e "clean_data=true" -v 2>&1 |
tee stress.log & sleep 600 ; ./ec2-start-server.sh -e
"wait_after_adding_server=120"
```

3.2.3 Benchmark Results

Figure 2 presents the throughput, as sum of what was measured by the loaders during the test. The drops of throughput when adding new nodes are clearly visible in the graph. After a short period of time during which the stabilization of the cluster occurs, this drop is restored and the obvious gain is visible for each additional node.

Axes of the following charts are as follows

- X axis is the time, with 10 seconds between each point.
- Y axis is either the number of operations (requests) per second for Figure 2 or latency in microseconds for Figures 3 and 4.

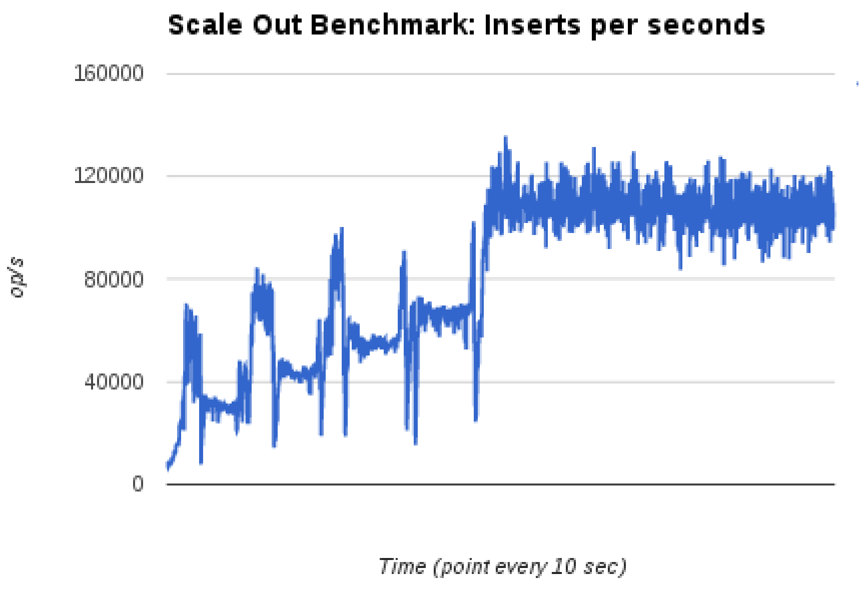

Figure 2: Scale Out Benchmark: Inserts per second.

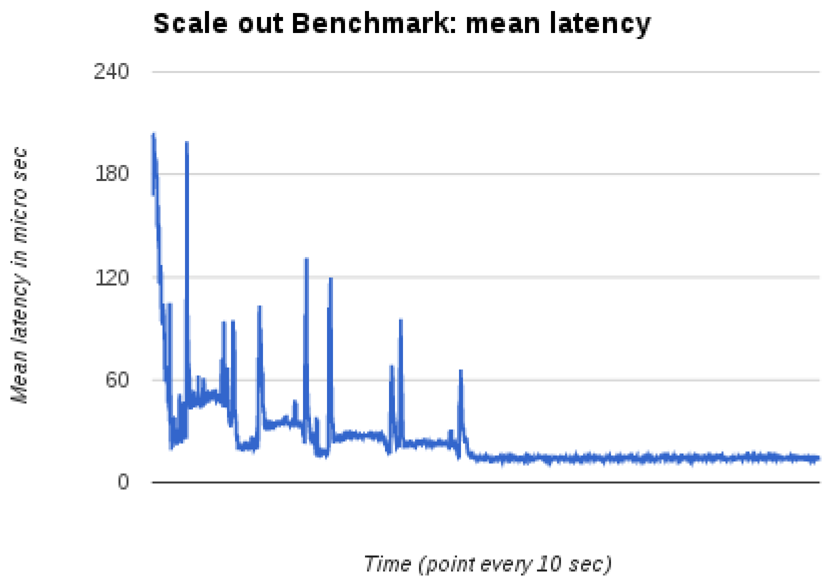

Figure 3: Scale out benchmark: mean latency.

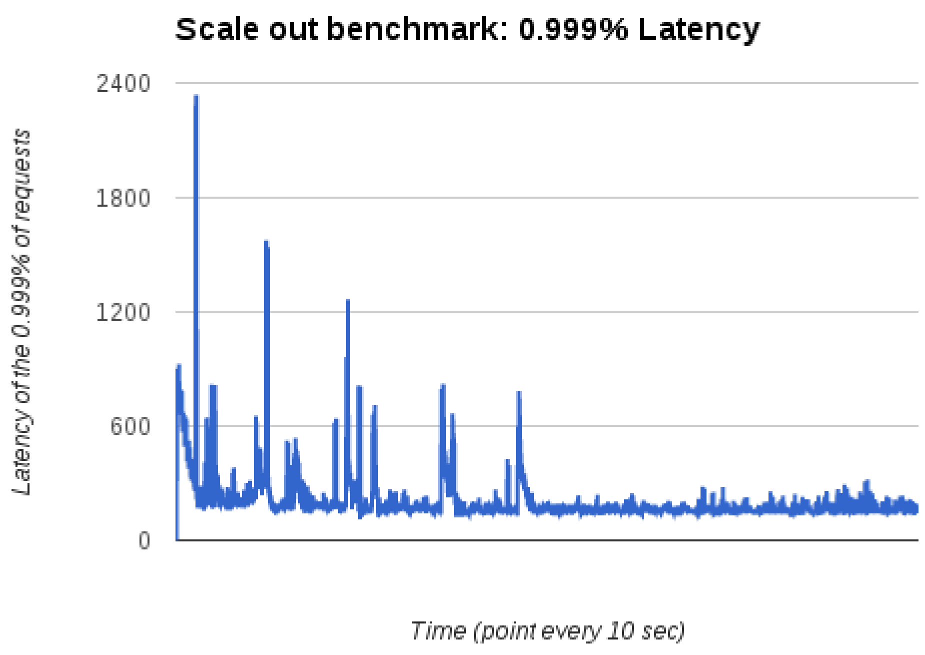

Figure 4: Scale out benchmark: 0.999% Latency.

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].

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 it 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, but it will be in the next phase of the project. The configuration script, available at [24], provides an overview of the settings used for the execution of the simulation.

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. We however 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 node with a VM. Results of simulations as executed in HPC testbed are available in [25]. Refer to deliverable D2.1 [22] for a detailed hardware and software specifications.

3.4 Big Data Open Data

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.

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.

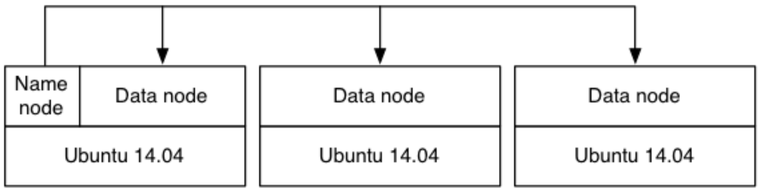

Figure 5: *Deployment of HiBench directly on the hosts.*

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.

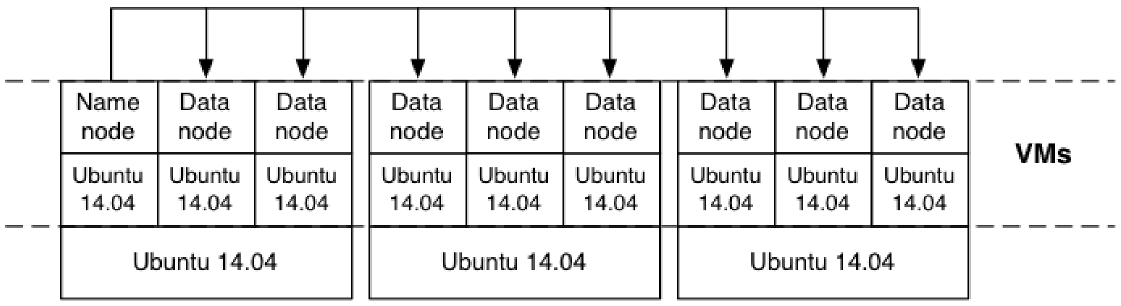

Figure 6: Deployment of HiBench in VMs.

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
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

Workload	Input Data Size	Duration (s)	Throughput (B/s)	Throughput per node (B/s)
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
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

Workload	Input Data Size	Duration (s)	Throughput (B/s)	Throughput per node (B/s)
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

Workload	Input Data Size	Duration (s)	Throughput (B/s)	Throughput per node (B/s)
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.

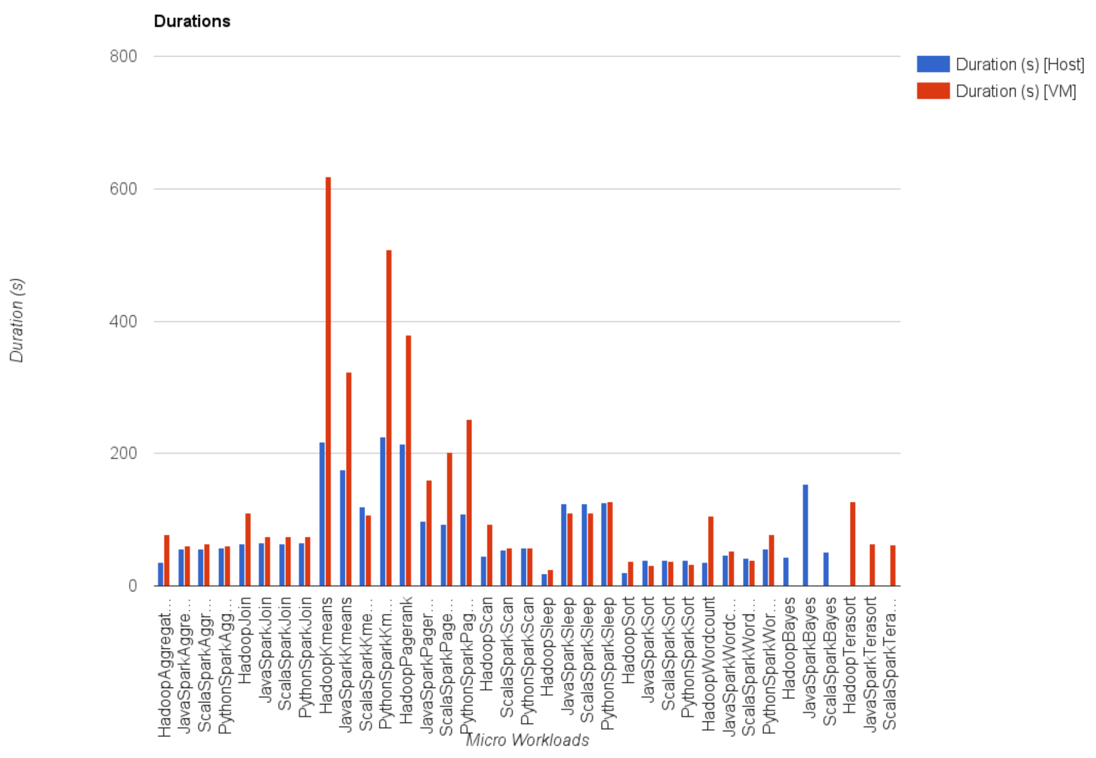

Figure 7: Comparison of benchmarking duration between host-based and VM-based runs.

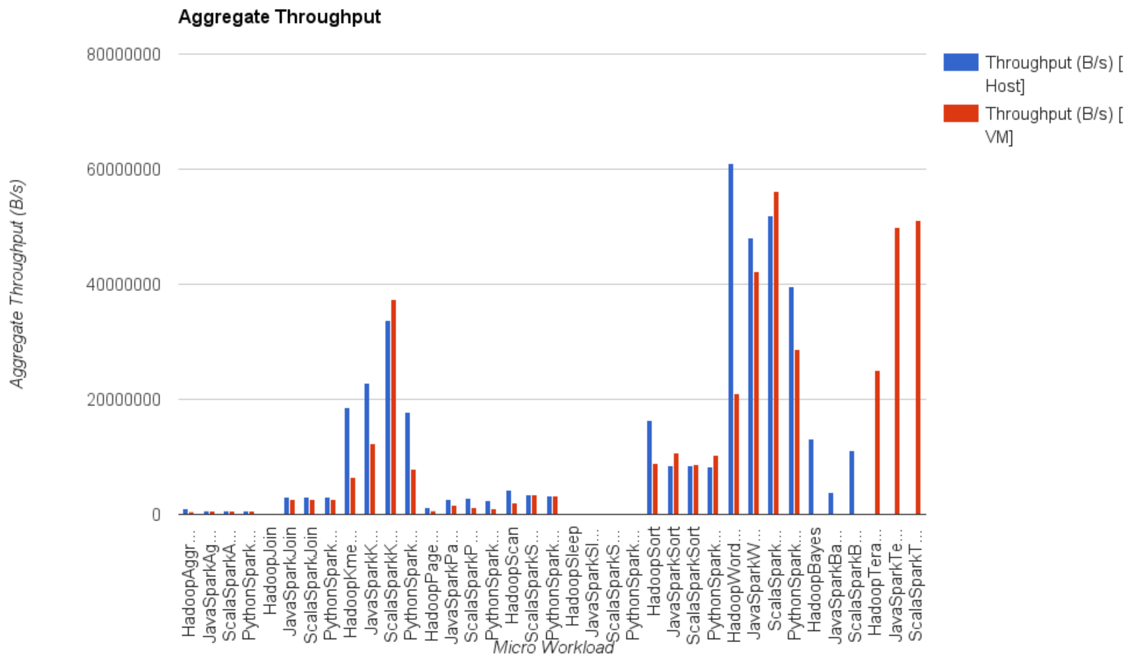

Figure 8: Comparison of aggregate throughput between host-based and VM-based runs.

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 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). For each of the aforementioned cases, we provide 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.

This document is an update of the D7.16 - The initial data management plan deliverable, which holds all the necessary legal and implementation framework descriptions for the MIKELANGELO Data Management Plan and participation in the Open Data Pilot activities. This update serves as the cover document for all the data provided by the MIKELANGELO project for year 1. The next update is D7.18, The second update on the Data Management Plan, will provide an overview of data gathered in year 2.

5 References and Applicable Documents

[1] MIKELANGELO Report D7.16 The initial Data Management Plan: http://www.mikelangelo-project.eu/deliverables/deliverable-d7-16/

[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] MIKELANGELO Report D5.4 First report on the Integration of sKVM and OSv with HPC, https://www.mikelangelo-project.eu/deliverables/deliverable-d5-4

[8] MIKELANGELO Report D2.19 D2.19 The first MIKELANGELO architecture, 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=132fd962e1d53adf9f53819535584e28

[13] OpenFOAM output results for Aerodynamics Use case, http://opendata.mikelangelo-project.eu/public.php?service=files&t=b79d4ae7c272e3ecd88a53ea3ab2324c

[14] OpenFOAM 1h outputs, http://opendata.mikelangelo-project.eu/public.php?service=files&t=c44e67261db8d993454a01ed54ee16aa

[15] OpenFOAM 4h outputs, http://opendata.mikelangelo-project.eu/public.php?service=files&t=3e363bdcb18a22e17dea405002b5b238

[16] Summary of execution times for HPC testbed, http://opendata.mikelangelo-project.eu/public.php?service=files&t=77a289e1434fa9f9b9846161645553fc

[17] OpenFOAM signal trace, http://opendata.mikelangelo-project.eu/public.php?service=files&t=1ca2818e217c783b6284e48e44176cfa

[18] MIKELANGELO Report D2.4 First Cloud-Bursting Use Case Implementation Strategy, https://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=c66391025208acf2499655be1879e343

[22] MIKELANGELO Report 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=3dec759d83ab9ebba8d39ce03503f298

[24] Input case onfiguration script, http://opendata.mikelangelo-project.eu/public.php?service=files&t=8c6ca37e8583ba329e2d807719dd028e

[25] Results of the initial measurement, http://opendata.mikelangelo-project.eu/public.php?service=files&t=ffbbf1c33223e53b83a27c5b54414509

[26] MIKELANGELO Report D2.7 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=f6e52b21cd8d3da98af55dd2c73f5912