D3.2 Initial Specifications for HPC Scalability Optimisation, HPDA Model Implementation, Data Management, Visualisation and Coupling Technologies

Document Identification			
Status	Final	**Due Date**	31/05/2019
Version	1.0	**Submission Date**	21/06/2019

Related WP	WP3	**Document Reference**	D3.2
Related Deliverable(s)	D3.1, D4.1, D5.1, D6.1, D6.2, D6.3	**Dissemination Level (*)**	PU
Lead Participant	PSNC	**Lead Author**	Marcin Lawenda
Contributors	ICCS, BUL, KNOW, USTUTT, PLUS, SZE ECMWF	**Reviewers**	Milana Vuckovic (ECMWF)
			David Bell (BUL)

Keywords:
High Performance Computing (HPC), Big Data, High Performance Data Analytics (HPDA) Benchmarking, Profiling, Scalability

This document is issued within the frame and for the purpose of the HiDALGO project. This project has received funding from the European Union's Horizon2020 Framework Programme under Grant Agreement No. 824115. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission.

Document Information

List of Contributors	
Name	**Partner**
Marcin Lawenda	PSNC
Piotr Dzierżak	PSNC
Dimitrios Tsoumakos	ICCS
Konstantinos Nikas	ICCS
Petros Anastasiadis	ICCS
Derek Groen	BUL
Nikela Papadopoulou	ICCS
Siergiy Gogolenko	USTUTT
Manuela Rauch	KNOW
Christine Gfrerer	PLUS
Zoltán Horváth	SZE
John Hanley	ECMWF

Document History			
Ver.	**Date**	**Change editors**	**Changes**
0.1	09/04/2019	PSNC	Created initial document, table of contents.
0.2	23/4/2019	ICCS, PSNC	Added content on Big Data computation engines and environments
0.3	30/5/2019	BUL, KNOW, SZE, ICCS, PLUS	Chapter 6
0.4	10/5/2019	ICCS, PSNC, USTUTT	Chapter 2, chapter 3
0.48	25/5/2019	PSNC, KNOW	Chapter 4, chapter 5, update of chapter 3
0.5	28/5/2019	PSNC	Before review
0.76	12/06/2019	PSNC, BUL, ICCS, KNOW, ECMWF, SZE, USTUTT, PLUS	After first review
1.0	19/06/2019	PSNC	Final version

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	Marcin Lawenda (PSNC)	19/06/2019
Quality manager	Marcin Lawenda (PSNC)	19/06/2019
Project Coordinator	Francisco Javier Nieto (ATOS)	21/06/2019

Table of Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation / acronym	Description
Dx.y	Deliverable number y belonging to WP x
EC	European Commission
ENS	Members' Ensemble
GIL	Global Interpreter Lock (Python)
HLRS	High Performance Computing Center Stuttgart
HPC	High Performance Computing
HPCG	High Performance Conjugate Gradient
HPDA	High Performance Data Analytics
HPL	High Performance LINPACK
HRES	High Resolution Weather Forecast
MPI	Message Passing Interface
PSNC	Poznań Supercomputing and Networking Center
TCP	Transmission Control Protocol
VM	Virtual Machine
WP	Work Package

Executive Summary

This document provides the initial strategies for optimising applications and implementing novel algorithms and methods. In particular, initial strategies for coupling applications in conjunction with WP4 are provided.

The deliverable starts with report on performance investigation of simulation tools. Due to complexity of the process we are not able, this time, to provide substantial findings only update on performed actions.

Since the scope of specific requirements for data analytics is still under investigation in this deliverable we focused on evaluation of available tools and test their performance. It would give us a rudimentary information on capabilities of systems and applications. This information constitutes a basis of choice for methodology and tools applicable for its implementation.

There are several frameworks described in the Chapter 3 including: Spark, Hadoop, Flink, Dask and R. Next, most of them are elaborated in respect of benchmarking software and approaches as well as in-depth performance analysis.

Requirements towards data managements system and findings on performance tests are discussed in the consecutive chapter. The topic of appropriate system optimization is based on buffer memory size allocation.

In the Chapter 5 we introduce four software tools for data visualization are described, three of them are benchmarked. These are Visualizer - a web-based tool for visualizing tabular data, TSA (Time Series Analysis) - a tool for analysing large-scale time series data, Signal search - a tool for sensor measurements investigation and finally, a Web Graph enabling users to visually investigate network data.

Chapter 6 presents coupling technologies which are used to combine different (existing) applications in order to make them work together for an overarching purpose. Coupling technologies are a staple in the multiscale and hybrid simulation approaches, and a range of generic technologies have emerged in recent years, each with their unique added values.

This document does not aim to propose any specific application or methodology for specific use cases. Rather it is intended to make an inventory of tools and tests their capabilities in order to check if requirements could be completely full filled. Later, in the project life time, when specific methodologies for development will be defined, these deliverable findings will serve as indicators for selection of the best tools and approaches for implementation.

1 Introduction

1.1.1 Purpose of the document

This document provides the initial strategies for optimising applications and implementing novel algorithms and methods. In particular, strategies for coupling applications in conjunction with WP4 will be provided.

Along with this document we are delivering basic knowledge about HPDA applications and their capabilities by providing performance test findings. It states a ground for coupling use case requirements and application(s) that will be used for their implementation. Moreover, we are discussing different approaches to huge datasets visualization. Four different applications are presented and three of them are benchmarked. Base on that we can pick out the best possible solution for specific demands addresses by pilots.

This deliverable is also intended as a starting discussion point on strategies for coupling technologies that are used to combine different (existing) applications. Making them work together could bring additional benefits in the multiscale and hybrid simulation approaches.

1.1.2 Relation to other project work

This document constitutes initialization of the discussion on HPDA and visualization applications and their capabilities therefore in its elaboration takes into account the following other work:

- D3.1 Report on Benchmarking and Optimisation
- D4.1 Initial Status of the Pilot Applications
- D5.1 HiDALGO System Environment
- D6.1 Requirements Process and Results Definition
- D6.2 Workflow and Services Definition
- D6.3 Artificial Intelligence Approach

Taking into consideration the status of the pilot applications (D4.1), preliminary knowledge on benchmarking and optimization provided in D3.1 along with available infrastructure (D5.1) we propose the most compelling solutions for data analytics and visualization. Furthermore, information included in D6.1 and D6.2 about overarching process requirements as well as first approach to the workflow and services definition are serving as grasp for analysis how to couple different existing technologies in order to get added value.

1.1.3 Structure of the document

The document is structured into 6 major chapters.

In Chapter 2 we give a short report on simulation application profiling.

Chapter 3 is about HPDA frameworks and libraries. Benchmarking tools and approaches are presented along with the first findings in prototyping environment.

Data management system is discussed in Chapter 4. We are concentrating on set of requirements addressed by Global Challenges systems and provide performance results in respect of imported data size and optimization factors related to buffer memory size.

Chapter 5 is about visualization and tools can be useful for that purpose: Visualizer - a web-based tool for visualizing tabular data, TSA (Time Series Analysis) - a tool for analysing large-scale time series data, Signal search - a tool for sensor measurements investigation and finally, Web Graph enabling users to visually investigate network data.

The initial findings from coupling technologies investigation are presented in the Chapter 6. They are used to combine different (existing) applications to make them work together for an overarching purpose.

2 HPC Benchmarking Findings

This section is intended to provide necessary update of the status on pilot applications benchmarking. Due to the complexity and time-consuming nature of the entire process, the simulation application profiling requires an in-depth knowledge of not only the application structure but also the available infrastructure solutions. Therefore, we are not able to provide any substantial results in this deliverable.

This note is just to inform and ensure that process is suitably investigated and findings will be presented along with next deliverables. The general overview of the undertaken activities is presented below.

2.1 HiDALGO Pilots

Following the guidelines for reporting set in D3.1, we are currently working towards modifying the various modules of the HiDALGO pilots to conform to those guidelines. Each application module is responsible for marking important application phases and reporting accurate timings for each phase and the application as a whole. We plan to extend reporting towards other metrics of interest for each application (performance counters, MPI-specific metrics, I/O-specific metrics, application-specific metrics etc.). We currently do not opt for a common interface for reporting, given the variety of programming languages used by the HiDALGO Pilots, however we intend to work towards a common API for reporting, using the tools described in D3.1, in later stages of the project.

In addition, we have set up the HiDALGO project repository, hosted by HLRS at https://projects.hlrs.de. Benchmarking and tracing results included in D3.1 are stored in the HiDLAGO project repository, following the repository structure, naming conventions and reporting guidelines defined in Section 2 of D3.1.

2.1.1 Migration Pilot

In D3.1, we show that Flee scales well for a small number of nodes on both available systems, Hazelhen and Eagle. We have identified minor performance bottlenecks that can be fixed with changes in the parallelization of Flee, while next steps for Flee include performance evaluation for larger simulations and I/O performance. The optimization of the I/O performance covers analysis aspects related to the yield and frequency of read/write operations as well as using a more efficient storage container for parallel processing (e.g. using HDF5 instead of CSV format). Current engineering effort around the Migration Pilot focuses on coupling (see chapter 6). Further benchmarking will be performed upon completion of this effort.

2.1.2 Urban Pollution Pilot

The native installation of the core component of the Urban Pollution Pilot, the 3DAirQualityPrediction component, is currently in progress on both Eagle and Hazelhen. As a result, benchmarking of the component will be performed once the native installation is complete.

2.1.3 Social Networks Pilot

In D3.1, we have performed initial scalability tests for the Validation module of the Social Networks Pilot. Our work on Hazelhen currently focuses on profiling the memory usage of the module on different number of cores, and how the memory usage relates to various parameters of the PETSc library. This is necessary to allow for efficient execution of the module for large-scale graphs. On Eagle, we are examining scalability issues that stem from the placement of MPI processes and OpenMP threads. In addition, the extreme memory and communication requirements of the module result in a variety of errors during the execution of the module with large graphs. We are working towards profiling and resolving these issues before proceeding with scalability measurements of the Validation module on Eagle. Next steps involve detailed communication and I/O profiling for the Validation module on both systems.

3 HPDA applications and libraries

The superior goal of the HIDALGO project is advancing the uptake of HPC and HPDA by implementation of a synergy factor stemming from the combination of both infrastructures and tools. This chapter presents candidates for data analytics frameworks as well as first benchmarking results which provide preliminary information about their capabilities.

3.1 Overview

The core aim of HiDALGO is to explore novel technologies and methods to enable the modelling of complex global challenges. The inherent complexity and scale of such modelling will necessitate processing vast amounts of data. The associated workflows can be complex, but for the most part they consist of two core phases: simulation and data analytics. For such large-scale problems the core simulation typically runs on a dedicated HPC while HPDA methods and tools for large datasets have mostly been developed and optimized to run in a cloud environment. A workflow coupling HPC-based simulation with cloud-based HPDA therefore has the advantage of leveraging existing optimizations; the disadvantage of this approach is the cost of data transfer between the simulation and HPDA components, coupled with potential inefficiencies on the cloud in terms of inter-node communication.

An alternative workflow is to migrate the HPDA functionality from the cloud to the HPC environment where the simulation is running to leverage the greater throughput and lower latency that such co-location affords. This approach has the advantage of potentially significantly reducing the time needed to transfer large volumes of data between the simulation and HPDA phases; the disadvantage of this approach is that it requires co-location of the simulation and HPDA components and a dedicated HPDA infrastructure which, by its nature, will not offer the same on-demand flexibility as a public cloud.

The performance and cost of the end-to-end workflow will naturally vary depending on the user's requirements. While users are typically interested in optimising for cost and overall performance, they may wish to instead focus on greater flexibility in terms of deployment at the cost of overall performance. The core aim of WP3 is to enable the construction of workflows which will offer the end-user the possibility of assessing which approach best suits their needs.

To enable the proposed alternative workflow of co-locating the HPDA functionality with the core HPC simulation we will explore two possibilities: the first involves no modification of the

HPDA applications and plans on using batch processing to transfer data (in the form of output files) between the simulation and HPDA processes; the second will involve continuous communication between the simulation and HPDA components either via stream processing using TCP or via MPI-bindings. The challenge of migrating HPDA in this manner will require a consistent and a unified software stack [1] and should be assessed using a variety of technologies, specifically in the areas of network communication and data file transfer.

This chapter aims to set the scene for HPDA installation and implementation for HiDALGO workflows without focusing on the technical details of such implementation; this will be covered in detail in future deliverables. First, we selected and described a number of applications and libraries that could be applicable for development of required procedures. In the next step, a benchmarking tools and methods were picked up for those tools. Finally, we performed these tests in order to get know more about performance, scalability, boundaries and limitations. This knowledge will be used later in the project lifetime under data analytics methods optimization. The more advanced processing scenarios will be investigated in the future WP3 work and presented in subsequent deliverables.

3.2 Applications

Due to the huge interest in many areas the domain of data analytics frameworks is developing very rapidly. This resulted in the creation of many tools dedicated to calculations in various applications. It seems to be obvious that there is no one all-purpose tool, which could be applicable for all kind of use case scenarios. Therefore, it is crucial to understand requirements and context of data processing. There are several facets relevant for this analysis: data size, way of computation, how processing can be natively facilitated by the framework.

At this stage of the project we are not able to comprehensively answer all these questions since the process of analysis of complex workflows is still undergoing.

The preliminary investigation against required list of needs was done in deliverables D6.1 and D6.2, where the first approach to the scenarios' definition was drafted. We can learn from this elaboration that coupling of data sources (e.g. scenario SCO-PIL-001) is indispensable to take benefits from synergy. In order to make it efficiently data must be preprocessed by validation, identifying key correlations or trends. Moreover, generalized HiDALGO workflow (D6.2, chapter 2.1) defines a number of data flows between specific modules which can be considered as HPC and HPDA convergence e.g.:

- "Data processing & Feature extraction" with "Model generation"
- "Simulation" with "Validation"

- Whenever huge datasets generated as output by one module must be processed by another one (in the execution sequence) data processing (e.g. filtering) is implied. One of prominent example is visualization of middle (transitional) data.

Furthermore, data analytics is also recognized as complementary (preliminary and closing) step of Artificial Intelligence (AI) processing. In most cases only limited subset of data organized in demanded way are transferred for AI processing. Afterwards, results (AI output) are transferred back DA where concluding processing steps are performed. A number of potential applications for different HiDALGO use case scenarios are drafted in D6.3, chapter 2. All of them will be further investigated in close collaboration with WP6 in order to define possible convergence and collaboration spots.

Having above reasoning in mind we have preliminary shortlisted the applications, which could be implied as a best fit for future development.

3.2.1 Apache Spark

Spark is the most prominent example of the software for data management analytics. It is well known for its computation speed, especially in cases when data can be entirely loaded to the memory. It is also equipped with libraries which facilitate machine learning (MLib) and distributed graph processing (GraphX).

For a general overview of Apache Spark, please reference:

- https://www.tutorialspoint.com/apache_spark/apache_spark_introduction.htm

For HiDALGO, we believe Resilient Distributed Datasets (RDD) will be particularly relevant:

- https://www.tutorialspoint.com/apache_spark/apache_spark_rdd.htm

3.2.2 Hadoop

Hadoop is a framework for distributed processing of large datasets across clusters of computers. It shows its strengths when we deal with amount of data much bigger than available memory size, through capabilities of the MapReduce module enabling computation among multiple worker nodes.

For an introduction to Hadoop, please reference:

- https://www.tutorialspoint.com/hadoop/hadoop_introduction.htm

In particular, we foresee MapReduce and HDFS as being potentially of interest for HiDALGO:

- https://www.tutorialspoint.com/hadoop/hadoop_mapreduce.htm
- https://www.tutorialspoint.com/hadoop/hadoop_hdfs_overview.htm

3.2.3 Flink, Dask, and R

Apache Flink is a framework and distributed processing engine for stateful computations over unbounded and bounded data streams. Compared with other big data frameworks, Flink excels at processing huge bounded and unbounded data sets.

Dask is a flexible library for parallel computing in Python. It is ideal for distributed computing both data and computation wise. For computation it features optimised dynamic task scheduling similar to Airflow, Luigi, Celery, or Make, but optimized for interactive computational workloads.

R is a language and environment for statistical computing and graphics. It provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques, and is highly extensible.

For more details regarding Flink, Dask and R, please refer to the relevant online documentation:

- https://flink.apache.org/
- https://dask.org/
- https://www.r-project.org/

3.2.4 ECMWF software

ECMWF provides software packages specifically developed to handle data and observations coding and decoding, visualization, data manipulation and archiving and managing of workflows which are very much related to HiDALGO use cases. For analyzing meteorological data related to the Migration and Urban Air Pollution pilots ECMWF offers a range of software packages, the details of which can be found here:

- https://www.ecmwf.int/en/computing/software

Software for data manipulation and visualisation include: ecCodes (encodes and decodes GRIB and BUFR messages), cfgrib (a higher level Python 3 interface to read GRIB data into xarray),

Magics (meteorological plotting software that can be either accessed directly through its Python or Fortran interfaces) and Metview (meteorological workstation application designed to be a complete working environment for meteorologists).

3.3 HPDA benchmarking

In this chapter we present information about the infrastructure used for testing as well as software we used for benchmarking (HiBench) along with detailed information on investigated procedures serving as reference for methods potentially implemented in use case data analytics solutions. Next, first findings are given offering different point of views: procedures, data scale, and framework comparison. They will be a reference point for the further benchmarking planned to be made as the work continuation on more advanced infrastructure.

3.3.1 Benchmarking infrastructure

This infrastructure is intended for fast prototyping reasons in order to provide environment for developing and early testing of elaborated solutions. Before researcher would get an access to "bold" data analytics infrastructure at PSNC or HLRS application developed procedures can be tested at this VM setup.

```
HiDALGO04 – Data analytics
Configuration details:
No of cores: 32
RAM: 32GB
Disk: 1040GB
IP Addresses: 150.254.165.237
Operating system: Ubuntu Server 16.04 LTS
```

This infrastructure was used for benchmarking Spark, Hadoop and R applications. For Dask performance estimation a Hazel Hen server (D5.1) at HLRS was utilized. All mentioned software had to be earlier installed for the project and testing purposes.

3.3.2 Benchmarking software

3.3.2.1 Spark & Hadoop

HiBench is a representative benchmark for Hadoop and spark developed by Intel. It consists a data generator that generates test data of different sizes for workloads. It utilises the Hadoop

Ecosystem which includes software stacks and frameworks like MapReduce, Hive, Nutch and Flink. HiBench benchmarks enable measurement of the following: execution time, resource utilization and throughput which are appended to an output file for analysis. Resource usage for each executed workload can be presented using web platform.

Source code is available for Linux operating systems. Hadoop and Spark are configured using configuration files [2] [3].

Currently HiBench (available in version 7.0) has workloads categorised into following areas: micro, sql, ml (machine learning), graph, websearch, and streaming.

Below we synthetized material on types of tests that were in the area of our interest and selected for further investigation. Information on all remaining tests is accessible on the website [4].

Micro Benchmarks:

1) **Sort** (sort) - This workload sorts its text input data, which is generated using RandomTextWriter.
2) **TeraSort** (terasort) - TeraSort is a standard benchmark created by Jim Gray. Its input data is generated by Hadoop TeraGen example program.
3) **WordCount** (wordcount) - This workload counts the occurrence of each word in the input data, which are generated using RandomTextWriter. It is representative of another typical class of real world MapReduce jobs - extracting a small amount of interesting data from large data set.

SQL:

1) **Scan** (scan), **Join** (join), **Aggregate** (aggregation) - These workloads are developed based on SIGMOD 09 paper "A Comparison of Approaches to Large-Scale Data Analysis" and HIVE-396. It contains Hive queries (Aggregation and Join) performing the typical OLAP queries described in the paper. Its input is also automatically generated Web data with hyperlinks following the Zipfian distribution.

Machine Learning:

1) **Bayesian Classification** (Bayes) - Naive Bayes is a simple multiclass classification algorithm with the assumption of independence between every pair of features. This workload is implemented in spark.mllib and uses the automatically generated

documents whose words follow the zipfian distribution. The dict used for text generation is also from the default linux file /usr/share/dict/linux.words.ords.

2) **Gradient Boosting Trees** (GBT) - Gradient-boosted trees (GBT) is a popular regression method using ensembles of decision trees. This workload is implemented in spark.mllib and the input data set is generated by GradientBoostingTreeDataGenerator.
3) **Random Forest** (RF) - Random forests (RF) are ensembles of decision trees. Random forests are one of the most successful machine learning models for classification and regression. They combine many decision trees in order to reduce the risk of overfitting. This workload is implemented in spark.mllib and the input data set is generated by RandomForestDataGenerator.

For **micro-benchmarks**, HiBench uses a configurable data generator, which mostly used Hadoop's RandomTextWriter to write binary text directly into HDFS. For **Web Search** and **Bayesian Classification**, HiBench uses Wikipedia page-to-page link database and Wikipedia dump file respectively. Hi-Bench typically generates unstructured datasets targeting the Hadoop file system. It must be noted that that HiBench uses only base systems of Hadoop and Spark. It is not possible to test Big Data storage systems (e.g. MongoDB, HBase, ArangoDB) using this benchmark.

3.3.2.2 Dask

In contrast to a `Spark` community, which developed a broad choice of well-designed open source benchmark suites for `Spark` (`spark-bench` from IBM CODAIT [5], `HiBench` from Intel [6], `Spark-Perf` from Databricks [7], `BigDataBench` from Chinese Academy of Sciences [8], etc.), `Dask` users and developers spent relatively little effort on addressing the issue of measuring `Dask` performance with benchmarks. There are only a few open source benchmark suites for data-driven applications that contain `Dask` components [9], [10], as well as several scientific articles and blog-posts about benchmarking `Dask` [11], [12] available on-line.

In this deliverable, we focus on the micro-benchmark [13] proposed by M. Rocklin – a core developer of `Dask`. This benchmark is designed to cover the following performance related aspects [12]:

- different computational and communication patterns (e.g., embarrassingly parallel, fully sequential, bulk communication, tree reductions);

- varying task duration;
- varying computational resources (it measures weak scaling);
- varying APIs (task scheduling, multidimensional arrays, and dataframes).

It allows users to measure the `Dask` performance for a variety of different workloads under increasing scales of both problem and cluster size. The benchmark comprises of three subunits:

- general task scheduling - covering for task scheduling and automatic generation of computation graphs in `Dask` with `dask.future` and `dask.delayed` [14].
- multi-dimensional arrays - benchmarking parallel array operations for arrays that don't fit into memory [14], [15].
- and dataframes - measuring performance of `dask.dataframe.DataFrame` – a parallel version of data structure which closely replicates the `pandas.DataFrame` [14], [16].

Since, in practice, many of the algorithms take advantage of `dask.bag.Bag` can be re-implemented with the same or better efficiency and less programming effort applying `dask.dataframe.DataFrame`, Rocklin's benchmark intentionally excludes tests for `dask.bag.Bag` – an important `Dask` data structure that allows to code MapReduce-style algorithms. In addition, since none of the HiDALGO use cases intend to benefit from `Dask`'s general task scheduling feature, we also excluded discussion of the results for the corresponding subunit of Rocklin's benchmark from the next subsection.

3.3.2.3 R

It is very hard to find an authoritative benchmark for R application. It results from the difficulty of giving an indisputable answer that specific build of R is performing better on a given architecture. However, there are existing benchmarks that are used quite often. One of them was developed by Philippe Grosjean [17] (based on work done by Stephan Steinhaus and others) applicable for multiple scientific packages. They can be found at:

- **R-benchmark-24.R** (https://mac.r-project.org/benchmarks/R-benchmark-24.R) - R benchmark 2.4, a modification of R benchmark 2.3 to work with current R and Matrix package
- **R-benchmark-25.R** (https://mac.r-project.org/benchmarks/R-benchmark-25.R) - R benchmark 2.5, same as above but scaled to more realistic times on current hardware.

3.3.3 Benchmarking findings

3.3.3.1 Spark & Hadoop

In order to get general overview about virtual machine capabilities we have performed several tests for different data sizes, represented as profiles: TINY and SMALL. Both profiles are applicable for developing phase in the project but for complex use cases much bigger profiles must be used (plan for next steps). Moreover, we modified two parameters related to number of used processes (cores) and parallelization factor, which as a matter of fact is number of MPI processes used by each process to execute chunk of the job assigned to it.

Profile TINY

		No. of processes x MPI processes				
		4x8	4x16	2x16	8x4	8x4 8G
Benchmark	Input data size (bytes)	Duration (s)	Duration (s)	Duration (s)	Duration (s)	Duration (s)
HadoopSleep	0	28,661	40,725	41,708	21,652	21,568
ScalaSparkSleep	0	9,156	9,096	9,2	9,136	9,167
HadoopSort	36778	25,962	35,742	36,644	23,556	23,599
ScalaSparkSort	36778	9,054	9,388	9,424	9,019	8,896
HadoopTerasort	3200000	26,629	37,581	38,806	23,656	23,637
ScalaSparkTerasort	3200000	9,023	9,275	9,672	8,937	9,151
HadoopWordcount	37418	25,468	37,53	38,536	23,476	23,47
ScalaSparkWordcount	37418	9,722	9,664	9,947	9,477	9,692
HadoopDfsioe-read	16869908	57,934	53,993	59,891	57,68	59,383
HadoopDfsioe-write	16992182	52,951	57,75	58,704	54,701	52,361
HadoopAggregation	37968	35,898	52,424	54,08	32,397	31,978
ScalaSparkAggregation	37968	21,007	21,666	21,87	21,165	21,213

		No. of processes x MPI processes				
HadoopJoin	201445	64,813	88,952	90,538	60,9	59,793
ScalaSparkJoin	201445	29,374	29,699	29,98	29,373	29,619
HadoopScan	205619	44,039	54,657	54,578	40,991	40,135
ScalaSparkScan	207092	19,673	19,773	20,1	19,482	19,307
HadoopPagerank	10705	53,932	78,302	78,618	48,856	48,899
ScalaSparkPagerank	10705	9,011	9,388	9,481	8,878	8,866
HadoopBayes	92776200	0,102	0,105	0,105	0,105	0,104
ScalaSparkBayes	92776200	21,141	21,19	21,043	22,907	23,306
LogisticRegression	809032	12,526	13,071	12,887	10,87	11,259
PCA	89032	20,997	22,372	22,068	16,596	16,822
GradientBoostingTree	12640	15,359	17,545	17,348	15,701	15,291
RandomForest	12640	9,701	10,232	10,525	9,455	9,496
SVD	805600	14,944	14,866	16,314	14,644	14,684
LinearRegression	4003013616	15,091	14,173	14,872	15,9	14,849
LDA	21898388	28,722	33,567	28,778	41,549	34,538
SVM	8065024	16,152	16,783	17,215	14,04	14,01
ScalaSparkNWeight	4354177	12,97	14,216	13,888	12,235	12,685

Table 1. The results of performance tests of the HiBench benchmark for the TINY model

Profile SMALL

		No. of processes x MPI processes				
		4x8	4x16	2x16	8x4	8x4 8G
Benchmark	Input data size (bytes)	Duration (s)	Duration (s)	Duration (s)	Duration (s)	Duration (s)
HadoopSleep	0	33,195	45,351	44,268	28,961	23,258

		No. of processes x MPI processes				
ScalaSparkSleep	0	37,991	37,652	38,217	37,316	36,789
HadoopSort	3287130	28,183	40,185	40,153	28,011	24,89
ScalaSparkSort	3287130	10,391	11,21	11,081	10,916	10,392
HadoopTerasort	320000000	32,485	47,695	47,592	33,495	32,33
ScalaSparkTerasort	320000000	14,923	15,431	15,716	15,895	16,475
HadoopWordcount	328496958	42,139	51,327	51,388	40,182	37,901
ScalaSparkWordcount	328496958	12,262	12,605	13,21	12,823	13,153
HadoopDfsioe-read	337405022	77,362	72,839	77,117	78,368	74,33
HadoopDfsioe-write	339334850	74,254	81,441	84,279	82,236	74,034
HadoopAggregation	3729493	42,404	56,669	55,791	41,217	36,588
ScalaSparkAggregation	3729433	25,274	25,66	25,443	24,433	24,259
HadoopJoin	19201694	77,023	100,659	104,824	74,346	64,397
ScalaSparkJoin	19201694	34,622	34,899	33,296	33,859	32,635
HadoopScan	20104750	53,352	59,052	59,881	51,859	45,48
ScalaSparkScan	20105279	23,36	23,819	23,569	22,978	22,444
HadoopPagerank	1811191	194,318	304,903	303,614	190,639	161,198
ScalaSparkPagerank	1811191	13,053	14,473	14,245	13,762	13,696
HadoopBayes	111385907	0,123	0,127	0,112	0,122	0,106
ScalaSparkBayes	111385907	25,609	24,935	26,206	24,283	26,262
LogisticRegression	80062008	19,311	19,345	19,77	17,567	15,642
PCA	8062008	43,987	43,666	43,507	42,984	41,787
GradientBoostingTree	408432	32,741	37,266	36,449	31,711	31,58
RandomForest	408432	13,122	13,209	13,031	12,23	11,506
SVD	16034800	55,19	50,983	51,874	54,686	53,806
LinearRegression	16006020600	36,54	34,86	34,313	33,82	32,677

		No. of processes x MPI processes				
LDA	96576628	116,932	80,939	95,516	92,239	141,379
SVM	800602600	22,523	22,953	22,119	22,598	22,589
ScalaSparkNWeight	39299788	38,8	37,5	36,686	37,738	37,063

Table 2. The results of performance tests of the HiBench benchmark for the SMALL model.

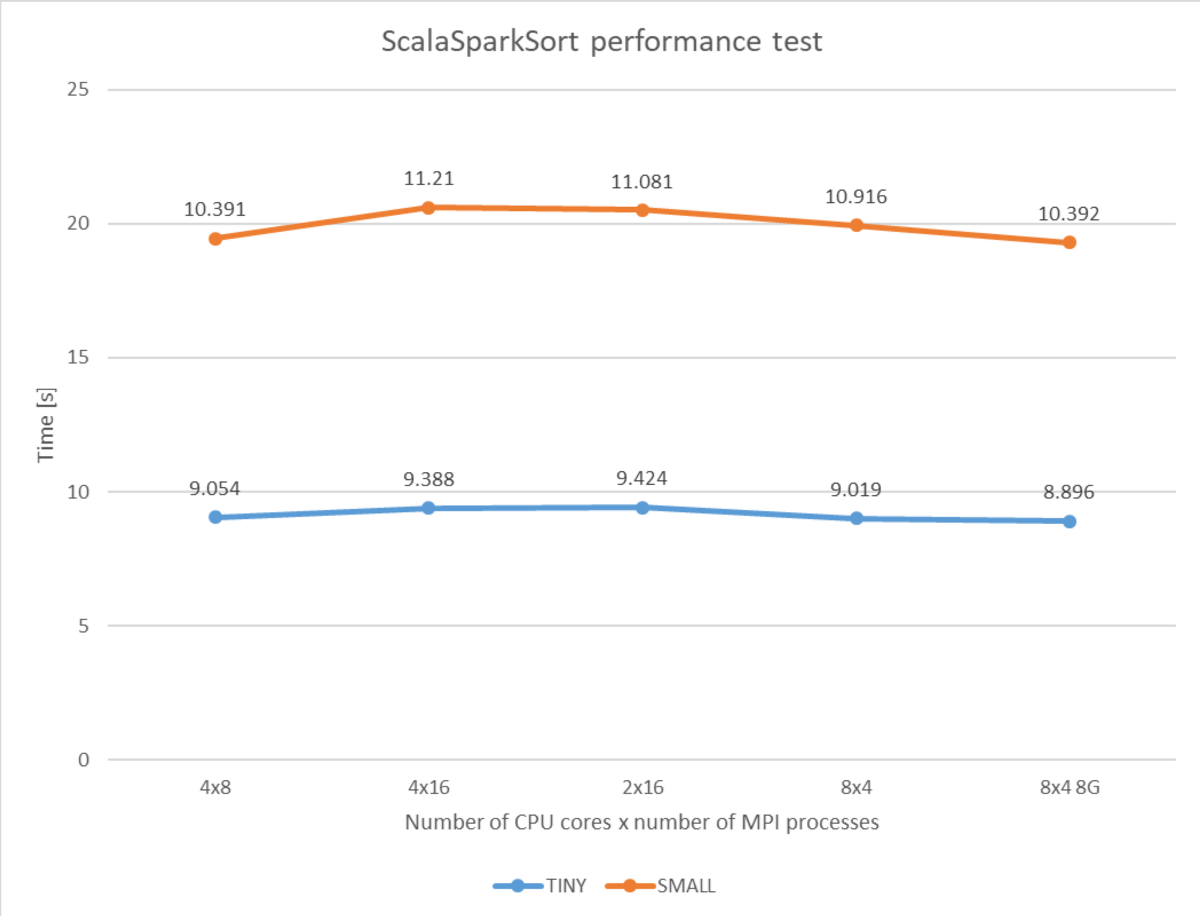


Figure 1. ScalaSparkSort performance test for TINY and SMALL model sizes

From the results we can conclude that, in overall, greater benefits can be obtained by using more processes than increasing number of MPI processes. Moreover, enlarging the amount of available memory (last column) for processing we can gain, in most cases, additional reduction of processing time. It is especially notable in cases where load is significantly large (reduction of memory paging process).

MicroBenchmark

Below is a summary of results for Spark and Hadoop tests using the MicroBenchmark test.

input data [MB]	time [s]	throughput [MB/s]
Spark Sort		
31.33	9.872	3.17
313.28	12.605	24.85
3'132.73	33.999	92.14
Spark Terasort		
30.52	9.58	3.19
305.18	14.92	20.45
3'051.76	75.22	40.57
Spark Wordcount		
31.33	10.36	3.02
313.28	12.26	25.55
3'132.75	19.90	157.39
Hadoop Sort		
31.33	24.72	1.27
313.28	25.92	12.08
3'132.73	70.44	44.47
Hadoop Terasort		
30.52	23.97	1.27
305.18	30.07	10.15
3'051.76	93.66	32.58
Hadoop Wordcount		

input data [MB]	time [s]	throughput [MB/s]
31.33	24.70	1.27
313.28	36.81	8.51
3'132.75	166.00	18.87

Table 3. HiBench MicroBenchmark test results.

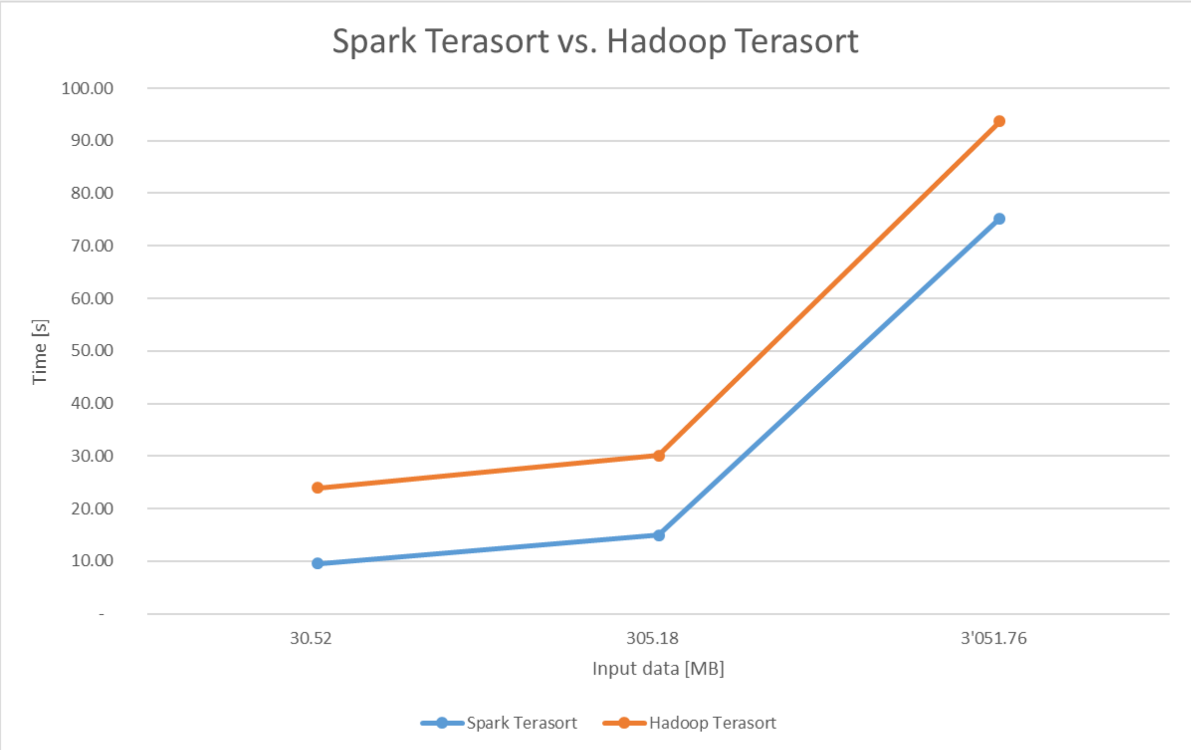


Figure 2. Comparison of Spark and Hadoop for Terasort test

In the above chart (Figure 2) we can observe that at some point expanding 10 times of input data size causes significant (exponential) increasing of execution time.

SQL tests

input data [MB]	time [s]	throughput [MB/s]
Spark Aggregation		
1.92	21.69	0.09
19.20	23.81	0.81
191.87	34.81	5.51
1'908.29	160.15	11.92
18'080.21	1'189.94	15.19
Spark Join		
1.83	30.20	0.06
18.27	30.73	0.59
183.06	36.74	4.98
1'830.44	48.93	37.41
18'304.85	123.47	148.25
Spark Scan		
1.93	19.63	0.10
19.14	20.46	0.94
191.67	26.55	7.22
1'916.58	44.43	43.14
19'166.38	157.32	121.83
Hadoop Aggregation		
1.92	42.84	0.04
19.20	47.09	0.41
191.87	72.66	2.64
1'908.29	529.28	3.61
18'080.21	5'400.21	3.35
Hadoop Join		

input data [MB]	time [s]	throughput [MB/s]
1.83	64.32	0.03
18.27	75.36	0.24
183.06	289.28	0.63
1'830.44	161.85	11.31
18'304.85	901.97	20.29
Hadoop Scan		
1.93	43.14	0.04
19.14	47.74	0.40
191.67	28.77	6.66
1'916.58	72.77	26.34
19'166.38	569.00	33.68

Table 4. HiBench SQL test results

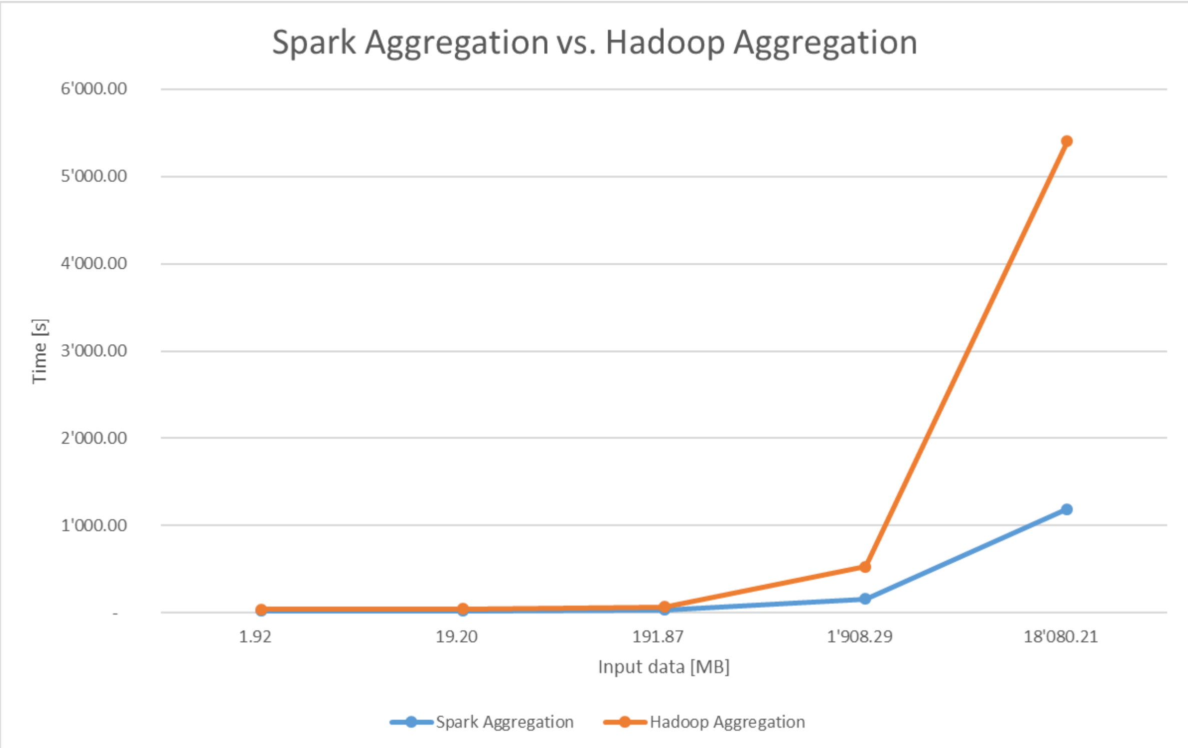


Figure 3. Comparison of Spark and Hadoop for SQL Aggregation test

In Figure 3 we can observe that in almost all cases Spark performs much better (faster) than Hadoop. Moreover, we can observe that at some point expanding 10 times of input data size causes significant (exponential) increasing of execution time.

Machine learning

Spark performance measurement for three tests: Bayes, Gradient Boosting Tree and Random Forest.

input data [MB]	time [s]	throughput [MB/s]
Bayes		
88.59	22.98	3.85
896.80	51.06	17.56
8'969.12	747.05	12.01
Gradient Boosting Tree		
87.80	591.67	0.15
877.44	16'273.48	0.05
8'774.38	92'678.00	0.09
Random Forest		
87.80	29.26	3.00
877.44	159.43	5.50
8'773.86	3'817.76	2.30

Table 5. HiBench Machine Learning test results

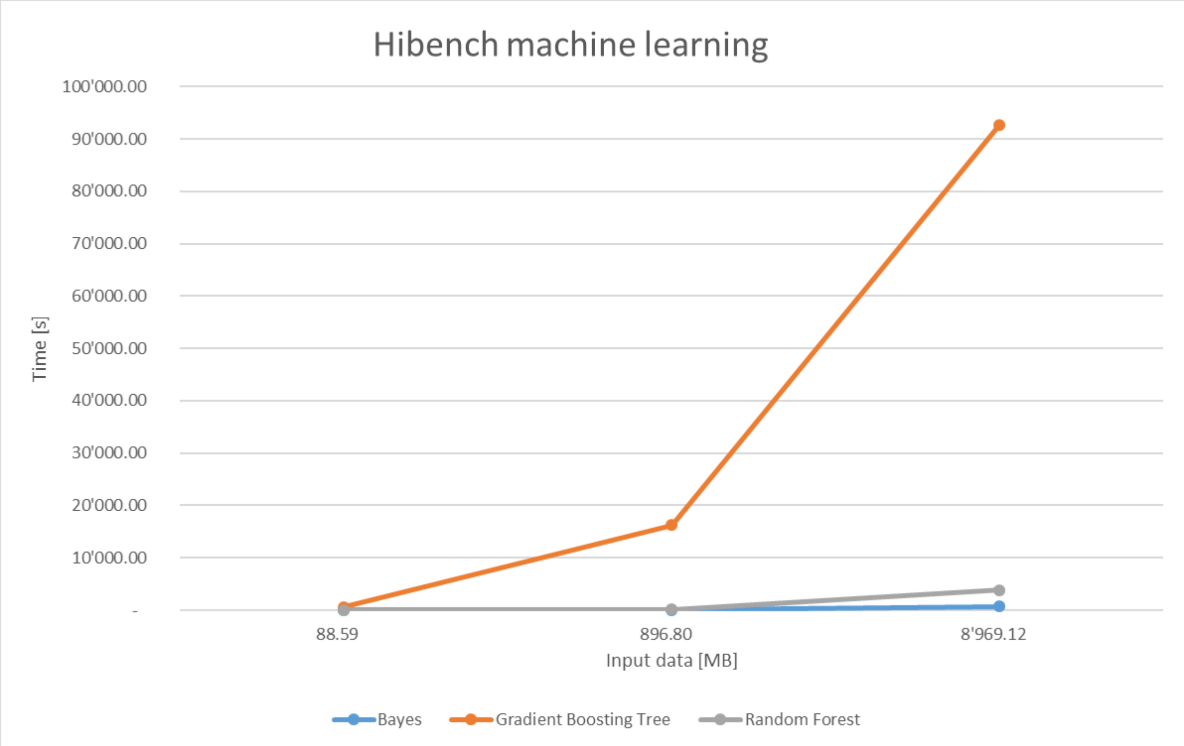


Figure 4. Illustration of Spark execution time of different machine learning models vs. input data size

The graph (Figure 4) illustrates execution time for all three selected machine learning models. I can be observed that Gradient Boosting Tree is much more sensitive to input data size when compared to other models.

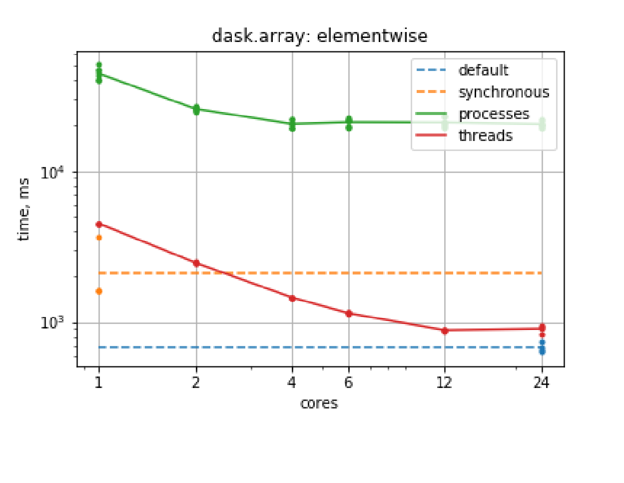
3.3.3.2 Dask

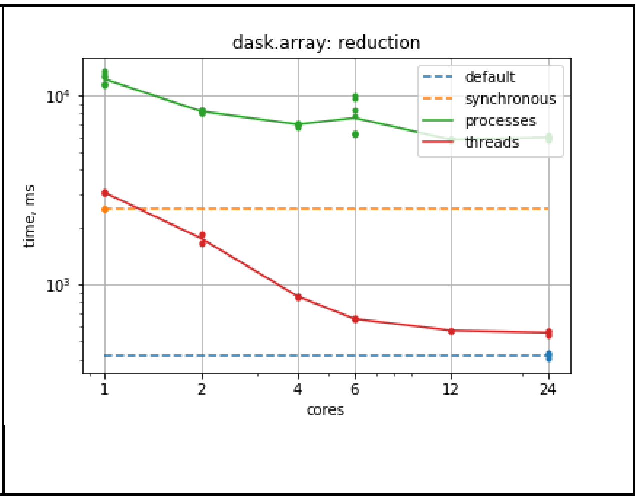
This section contains analysis of results for benchmark of the `Dask`'s single-machine scheduler only [18], [19]. The single-machine scheduler is a simple scheduler with little overhead for launching `Dask` codes on a single node. It supports three modes: processes mode for execution in multiple processes, threads mode for multi-threaded execution, and single-threaded more for running computations in a single thread. By default, `Dask` sets up single-machine scheduler in a threads mode for arrays and dataframes. `Dask` creates multiple threads and local processes by means of `ThreadPool` and `ThreadPool` classes from `multiprocessing.pool` library. Although the user can substitute custom threading and multiprocessing libraries with the same API.

We use the latter option to control the number of threads and local processes in our experiments.

In order to run Rocklin's benchmark with a single-machine scheduler, we changed benchmark sources accordingly, since the original benchmark was written for distributed scheduler. In particular, we modified the codes for setting up the scheduler and reduced the size of test inputs. In the updated benchmark, we use 2D integer arrays (matrices) of size 24Kx24K comprised of 2Kx2K chunks and dataframes of size 576Gx10 comprised of 576Kx10 chunks filled with integer elements. We did not further tune the benchmark or configuration at all for the experiments. Thus, they represent of what might be expected from `Dask` without setting up this framework properly or thinking about its configuration. All tests were executed on a single node of Hazelhen cluster (see D5.1 for details on Hazelhen cluster compute nodes configuration). We used `Dask` 1.2.2 from the latest Cray Python3 bundle in the wild [20], [21].

Figure 5 and Figure 6 reflect results of the benchmark by illustrating strong scaling properties of `Dask` in different modes of single-machine scheduler. On those plots, dashed blue lines correspond to the default setup of single-machine scheduler, dashed yellow lines correspond to the synchronous (single-threaded) mode, red lines correspond to the threads mode, and green lines correspond to the processes mode. As experiments show, the default setup is permanently better than synchronous mode. In addition, the default setup is almost always demonstrated the best result over all configurations. It indicates that by default `Dask` benefits from parallelization on a single node in the best possible way.





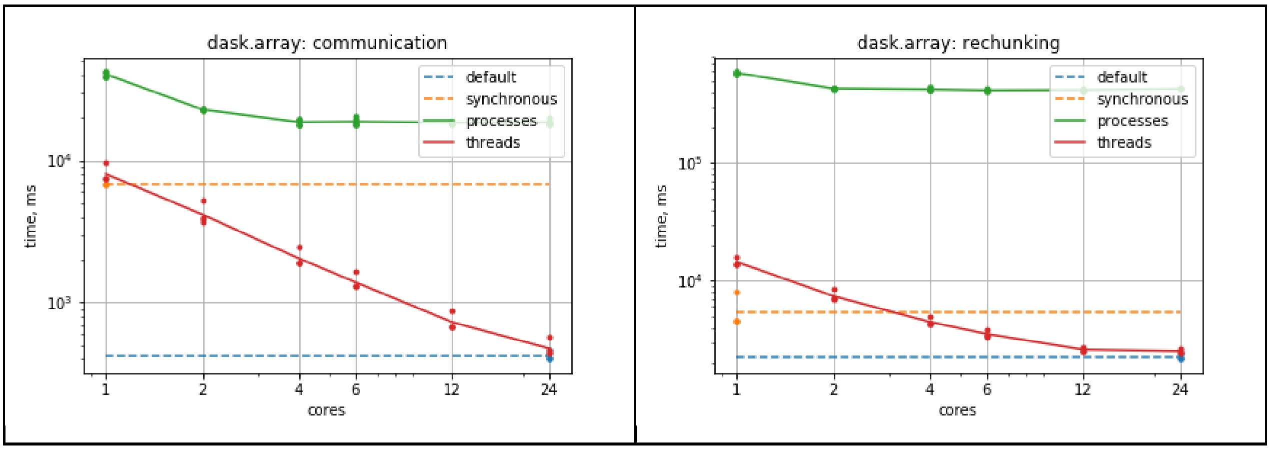


Figure 5. Performance of Dask arrays

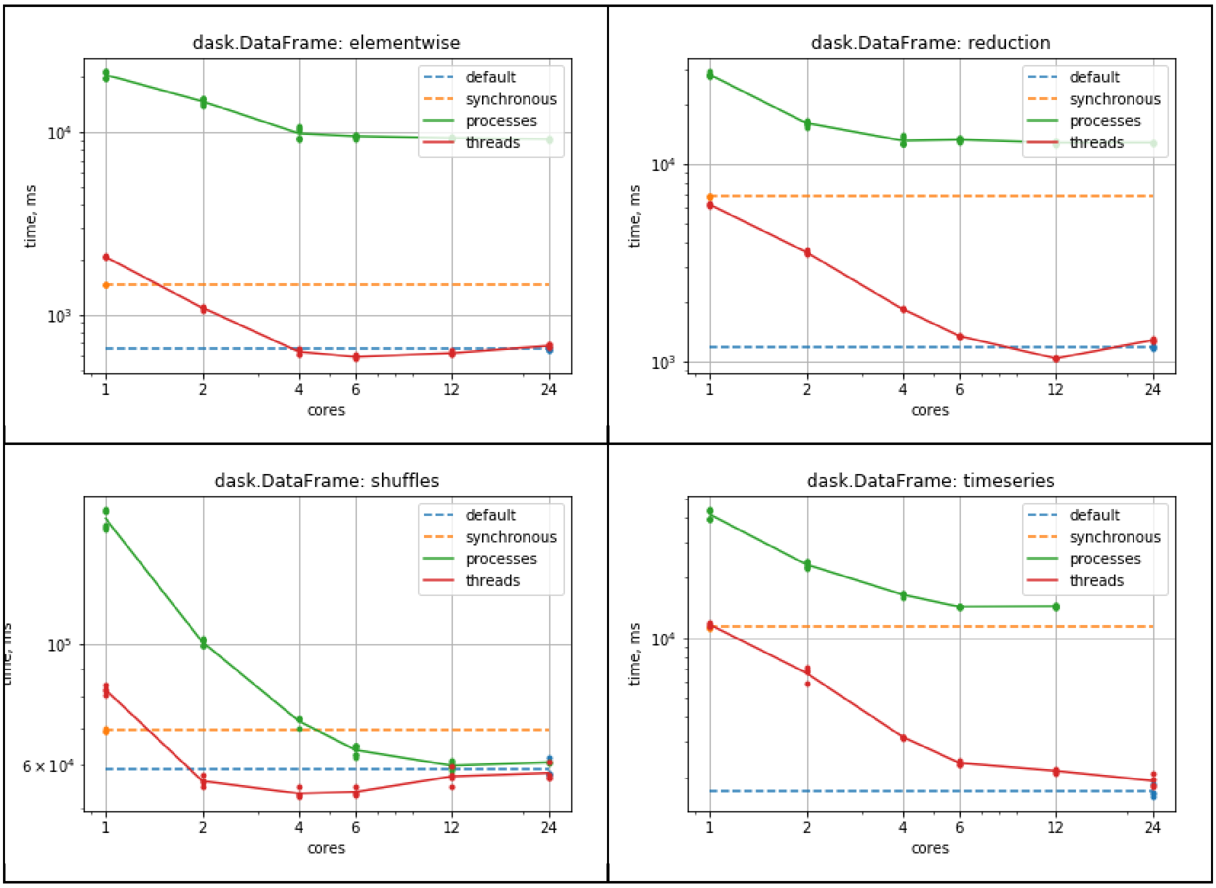


Figure 6. Performance of Dask dataframes

In all experiments, the elapsed time for processes mode is an order of magnitude larger than the elapsed time for other modes. Moreover, although the total elapsed time monotonically reduces with the number of processes, the processes mode does not beat synchronous mode even if it employs all physical cores. This is an expected behaviour, since the processes mode is designed for GIL-bound codes, while `Dask` arrays and dataframes are built on top of in-memory computational systems like `NumPy` and `Pandas` that release the GIL.

The benchmark shows the same pattern for all `Dask` array operations in the threads mode: the elapsed time monotonically reduces with the number of involved physical cores, and converges to the elapsed time in the default `Dask` setup. The behaviour for `Dask` dataframe operations is slightly more complicated. Timeseries operation (rolling aggregation test) scales to all 24 physical cores, while other operations scale up to a smaller number of cores: 6 cores in case of elementwise and shuffling operations and 12 cores in case of reduction operation. In the latter cases, the threads mode with the optimum number of cores performs better than the default `Dask` setup. Note also that the threads mode introduces small overhead in a single-threaded setup, which can be seen from comparison of the plots for the threads mode and the synchronous mode.

In the next deliverable, we will present the results of measuring performance of the distributed scheduler [18]. In addition, we plan to identify and benchmark performance of computational kernels available in `Dask` and needed by HiDALGO use-cases and benchmark.

3.3.3.3 R

In this section we present benchmark findings for four different problems (detailed description in subchapter above):

- Eigenvalues
- Hilbert matrix
- Toeplitz matrix
- Linear regression

Input size	Elements	Time (s)	Throughput (elem/s)
Eigenvalues			
600x600	360'000	0.733	491'132
1897x1897	3'598'609	10.300	349'380
6000x6000	36'000'000	139.706	257'684
Hilbert matrix			
600x600	360'000	0.008	45'000'000
1897x1897	3'598'609	0.337	10'678'365
6000x6000	36'000'000	2.648	13'595'166
Toeplitz matrix			
600x600	360'000	0.796	452'261
1897x1897	3'598'609	7.020	512'622
6000x6000	36'000'000	66.661	540'046
Linear regression			
600x600	360'000	0.188	1'914'894
1897x1897	3'598'609	0.199	18'083'462
6000x6000	36'000'000	2.494	14'434'643

Table 6. R benchmarking results for various reference problems

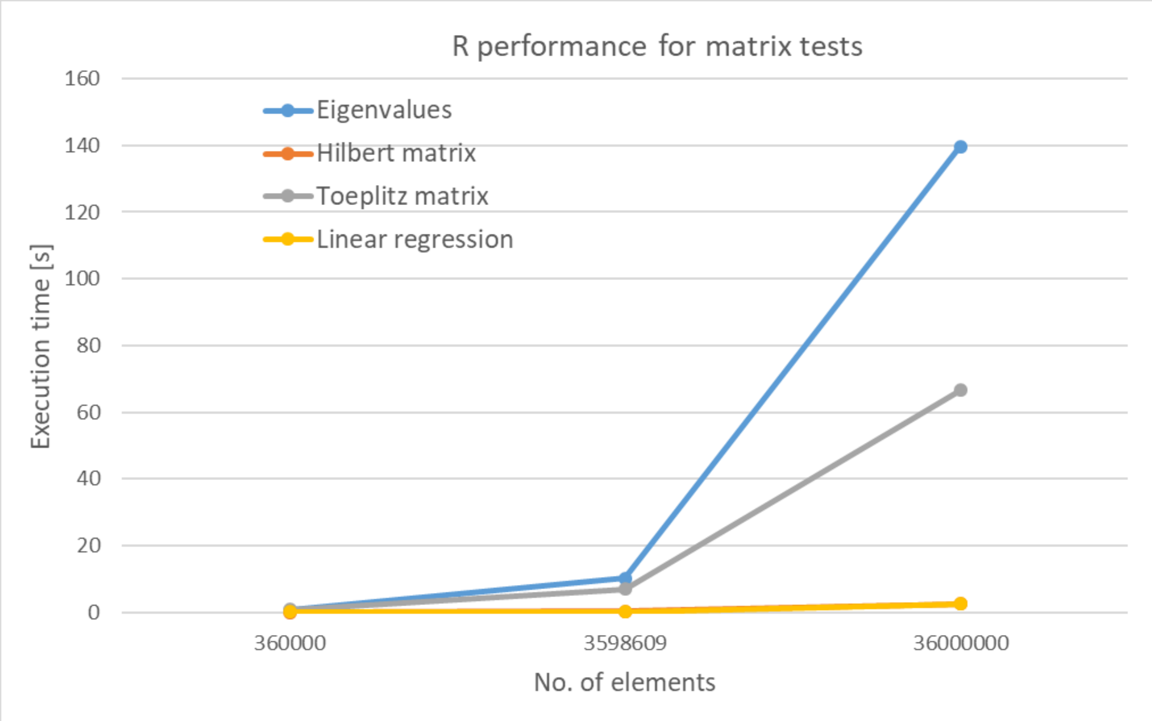


Figure 7. Execution time vs. no. of elements for various matrix tests

From above figure (Figure 7) we can notice that in most cases tenfold increasing of the problem size causes exponential increase of the computational time. Only "Linear regression" shows a linear tendency of the processing time.

4 Data management

Data Management System (DMS) is an indispensable part of the workflow for Global Challenges (GC) systems. It can be assumed that it would be the most heavily utilized module of the system by many applications on various stages of scenario execution. Furthermore, since we are dealing with parallel computing where simultaneous operations are executed on the same storage resource the framework must support distributed and parallel operations.

Moreover, we can expect a tremendous growth of the amount of data as well as their heterogeneity – taking into account different data sources, e.g. data streams from social media. It drives us to two main problems which should be solved in the proposed data management framework: efficient data management and reliable computation.

Taking into consideration high demands of GC system towards capacity, efficiency and reliability selection of the DMS framework must be made very carefully.

In this report we provide overview on requirements addressed by the use cases as well as software contenders. From the wide group of available candidates, we selected two: CKAN and Rucio. This time we investigated the first one, the information about the later one will be reported along with the next deliverable.

4.1 Requirements

The use case applications developed in the project deal with very large data sets and this makes parallelization, scalability and pre-processing of data very important. It is also necessary to be able easily run the model with many sets of parameters to do parameter scans over a multi-dimensional parameter space. It imposes additional efficiency requirements on DMS when data need to be efficiently delivered to processing place.

Moreover, important requirements which are addressed are concerned with the ability to individually tailor data into to the three different pilots, including aggregation of data from different data sources as well as storing results from simulations and data analysis methods to be applied.

One of the overriding goals of the HiDALGO project is effective convergence of the HPC and HPDA systems. It imposes additional requirements on the data management system related to the cooperation with the infrastructure of both systems allowing for effective access and collection of data sets required by pilot applications.

At the end we cannot forget about the possibility to cooperate with other parts of the HiDALGO system like portal is. It must consider that all operations executed by the user are made from the portal level so DMS must be properly interfaced at this level as well. It complies operations like data exploration, preliminary visualization, harvesting other sources and storage structuration, which conform the workflow in organization.

From the requirements above the following aspects were identified:

- Relational databases are not suitable due to the size of the datasets. Instead, new Big Data management systems for unstructured data, such as CKAN, or Rucio, have to be used.
- The data that will be collected from many different sources must be stored and organized in a consistent way, enabling the HiDALGO applications (including the pilots) to easily select and adapt it to their specific configurations.
- Handling HPC and HPDA convergence by effective cooperation with both systems
- The dynamic part of the use case scenario needs to be calibrated based on relatively sparse and aggregated data. This requires flexibility that deal with unstructured and incomplete data sets.
- Functionality is needed to explore the massive multi-dimensional results of use case simulations such as those performed by the pilots.
- Interfacing with the portal level.

4.2 Software

There are two software solutions we are going to investigate in this project in order to choose the one that better meets the requirements addressed by use cases:

- CKAN (Comprehensive Knowledge Archive Network) [22]
- Rucio [23]

Along with this deliverable we are going to focus on the first one (CKAN) while in the next reports Rucio will be elaborated.

4.2.1 CKAN - Introduction

The purpose of CKAN is to register datasets, facilitate the search of datasets, and finally provide access to these datasets. Besides providing these core functionalities related with data storing, CKAN also allows grouping of datasets, creating organizations, metadata management, and relationship management of data, it can handle different data formats, can harvest external data sources and provides a shared pool of data where all can benefit from the publicly available collection of data of other users which covers most if not all of the pilots' requirements.

In order to fully meet the requirements, CKAN is capable to offer extra functionality by implementation dedicated extensions like:

- Datapusher: Whenever a structured resource is added to a dataset, this extension downloads the file (CSV or XLS), parses the data and pushes the data to Datastore.
- Datastore: Provides an ad-hoc database for structured CKAN resources which allows automatic data previews on the resource page. Adds search filter and update operations for the data without having to download, edit and upload the whole resource.
- LDAP: Allows LDAP authentication for CKAN.
- Disqus: Allows users to publish comments about datasets.
- DCAT: This extension provides plugins that allow CKAN to expose and consume metadata from other catalogues using RDF documents serialized using DCAT.
- Harvester: Provides a command line interface and adds a web user interface to CKAN for managing harvesting other CKAN instance datasets. By using either of these interfaces, a harvesting source can be set and the extension creates processes which download all the publicly available datasets from the source and adds them to the CKAN instance.
- DREL: This extension is being implemented in the context of CoeGSS [24]. The purpose of the extension is to create and manage relationships between datasets such as: parent-child, dependency and derivation.
- Theme: This extension is needed in order to modify the default CKAN look-and-feel to be in line with the HiDALGO visual style. CKAN encourages creating extensions for visual changes for the sake of maintainability and stability.

4.2.1.1 CKAN Use Cases

In general, there are two main user roles in CKAN: system administrator and regular user. The CKAN system administrator is always allowed to perform any functionality provided by CKAN. A regular user has to be a part of an organization or a group in order to be able to gain more rights (excluding the user's own profile management) than non-authenticated visitors. A non-authenticated visitor can view publicly available data on CKAN, search and view datasets and access the visualization tool. Any user registered to the HiDALGO portal is assigned the regular user role and will have all the rights of a regular user – plus the rights that come with the organization and/or group memberships. The latter are detailed in the paragraphs below.

If a user who is already registered to the HiDALGO portal accesses CKAN, he/she will be able to view all the publicly available datasets and all groups. In order to be able to create or edit content, the user has to be assigned to an organization. Therefore, a mechanism where the user can reach organization administrators to request an invitation or reach a system administrator to request a new organization to be created will have to be implemented.

Once the organization assignment is done, the user will be able to create, edit or delete datasets belonging to that organization. Any user assigned to a group will be able to take the actions stated in Subsection 4.2.1.3, depending on the group role assigned.

Besides handling datasets, the users will also be able to manage relationship links between the datasets belonging to their organization.

4.2.1.2 Organization membership

In CKAN, each dataset has to belong to an organization and a dataset can only be owned by a single organization. Organizations control which user can see, create and update these datasets. A user can have one of these three roles in an organization: admin, editor, and member. An **organization admin** can:

- view the organization's private datasets;
- add new datasets to the organization;
- edit or delete any of the organization's datasets;
- make datasets public or private;
- add users to the organization, and choose whether to make the new user a member, editor or admin;
- change the role of any user in the organization, including other admin users;

- remove members, editors or other admins from the organization;
- edit the organization itself (for example: change the organization's title, description or image);
- delete the organization.

An **organization editor** can:

- view the organization's private datasets;
- add new datasets to the organization;
- edit or delete any of the organization's datasets.

An **organization member** can**:**

- view the organization's private datasets.

4.2.1.3 Group membership

Groups in CKAN work like controlled tags and allow categorizing datasets. A dataset can belong to arbitrary number of groups. There are two different roles in a group: admin and member. A **group admin** can:

- add datasets to the group or remove existing datasets;
- add or remove group members, and choose whether to make the new user a member or an admin.

A **group member** can:

- add datasets to the group or remove existing datasets.

4.2.2 CKAN - Performance tests

4.2.2.1 Input file size

After a successful CSV file upload, the CKAN datapusher plugin converts input data (comma-separated values) to a database representation – one input CSV file – one table in PostgreSQL database. We made 10 repetitions for each individual input files.

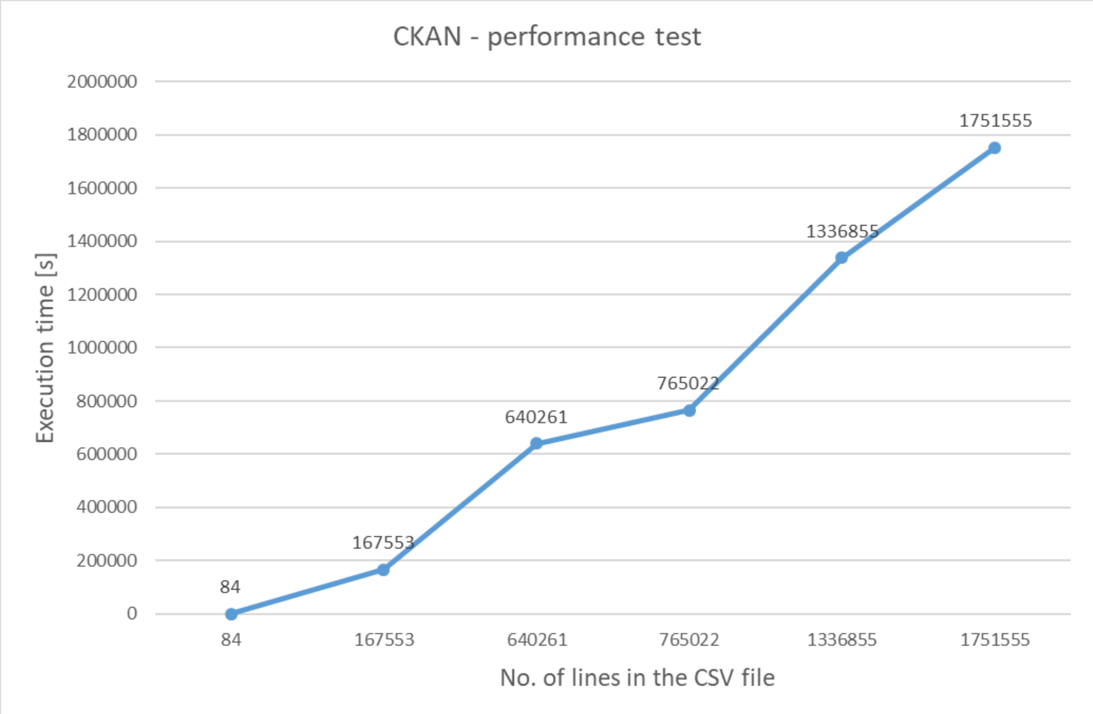


Figure 8. CKAN performance test – data import processing time.

Above figure illustrates how CKAN performs depending on input file size, or more precisely number of lines need to be processed. Each line is composed of some number of properties assigned to agent, which is elementary processing unit. We can observe that processing time is linear.

4.2.2.2 Database parameters

CKAN base on the PostgreSQL database. We analysed potential PostgreSQL performance parameters and selected two of them:

- **work_mem** - amount of memory to be used by internal sort operations and hash tables before writing to temporary disk files,
- **shared_buffers** - amount of memory the database server uses for shared memory buffers.

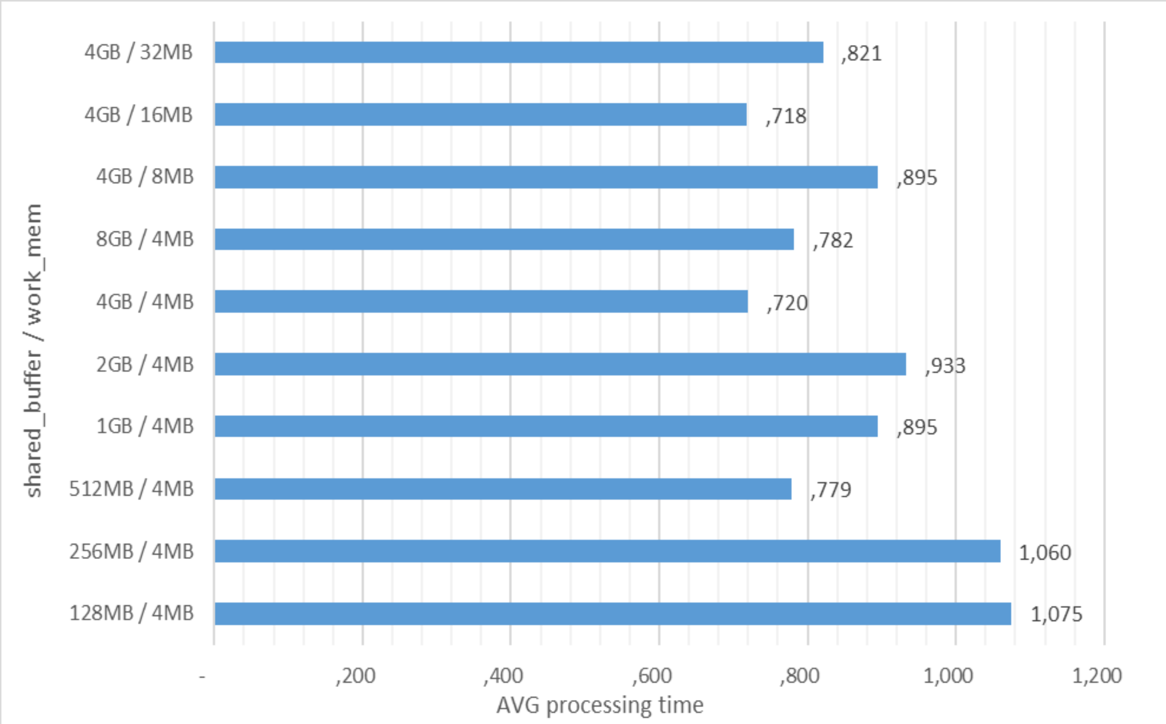


Figure 9. Average processing time depending on configuration parameters shared_buffers/work_mem

The best test results were obtained by configuring the parameters shared_buffers = 4GB and work_mem = 16MB. Compared to the default parameters values (shared_buffers = 128MB and work_mem = 4MB), the average processing time of the datapusher plugin has decreased by about 6 minutes. The values of these parameters have been set as production for deployed system.

5 Visualization

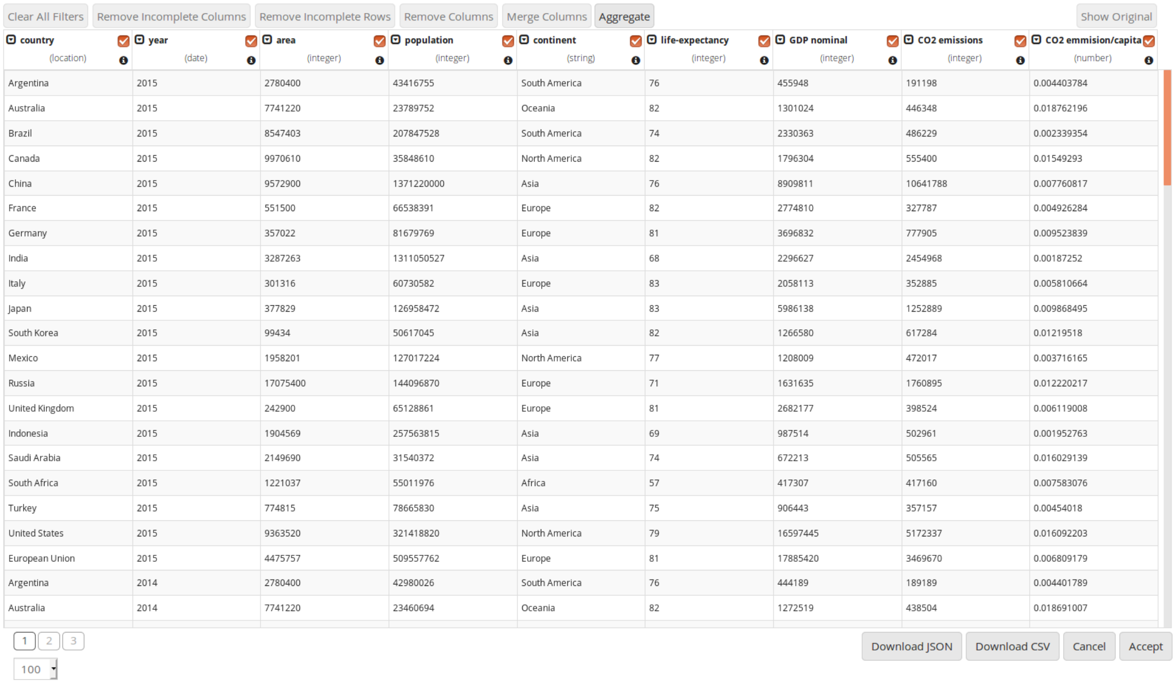
In this section four software tools for data visualization are described, three out of them are benchmarked. These are Visualizer - a web-based tool for visualizing tabular data, TSA (Time Series Analysis) - a tool for analysing large-scale time series data, a tool for signal search and finally, a web graph enabling users to visually investigate network data. The last approach is described but not benchmarked as it is designed to initially show most relevant data and enables users to further explore the data depending on their demands.

5.1 Visualizer

5.1.1 Description

Visualizer [25], [26] is a web-based data visualization tool for analysing any tabular data and can be accessed online using https://visualizer.know-center.tugraz.at. Visualizer supports CSV and JSON format. All data processing is performed on the client, which allows users to investigate any sensitive data. Visualizer provides two different views, a Table View, see Figure 10, and a Dashboard View, see Figure 11. The Table View enables users to investigate their raw data, provides automatic datatype detection and allows simple data manipulation operations, e.g. filtering data fields or replacing values. The Dashboard View allows users to select multiple data field on the left side and create possible visualizations on the right side. Visualizations within the Dashboard View are coordinated with each other, meaning that interactions in one visualization highlight the corresponding fields in all other visualizations. Currently, data can only be loaded from the directory or by using an online source but it can be extended to support additional sources e.g. REST API.

When generating a dashboard, the corresponding URL is constantly updating, adding all dashboard configuration parameters. This allows users to share the generated dashboard with other users. Additionally, this URL can be used to integrate the dashboard into other web applications using I-frames. Furthermore, Visualizer can be extended by users uploading their visualisations using a pre-defined API.



country (location)	year (date)	area (integer)	population (integer)	continent (string)	life-expectancy (integer)	GDP nominal (integer)	CO2 emissions (integer)	CO2 emmision/capita (number)
Argentina	2015	2780400	43416755	South America	76	455948	191198	0.004403784
Australia	2015	7741220	23789752	Oceania	82	1301024	446348	0.018762196
Brazil	2015	8547403	207847528	South America	74	2330363	486229	0.002339354
Canada	2015	9970610	35848610	North America	82	1796304	555400	0.01549293
China	2015	9572900	1371220000	Asia	76	8909811	10641788	0.007760817
France	2015	551500	66538391	Europe	82	2774810	327787	0.004926284
Germany	2015	357022	81679769	Europe	81	3696832	777905	0.009523839
India	2015	3287263	1311050527	Asia	68	2296627	2454968	0.00187252
Italy	2015	301316	60730582	Europe	83	2058113	352885	0.005810664
Japan	2015	377829	126958472	Asia	83	5986138	1252889	0.009868495
South Korea	2015	99434	50617045	Asia	82	1266580	617284	0.01219518
Mexico	2015	1958201	127017224	North America	77	1208009	472017	0.003716165
Russia	2015	17075400	144096870	Europe	71	1631635	1760895	0.012220217
United Kingdom	2015	242900	65128861	Europe	81	2682177	398524	0.006119008
Indonesia	2015	1904569	257563815	Asia	69	987514	502961	0.001952763
Saudi Arabia	2015	2149690	31540372	Asia	74	672213	505565	0.016029139
South Africa	2015	1221037	55011976	Africa	57	417307	417160	0.007583076
Turkey	2015	774815	78665830	Asia	75	906443	357157	0.00454018
United States	2015	9363520	321418820	North America	79	16597445	5172337	0.016092203
European Union	2015	4475757	509557762	Europe	81	17885420	3469670	0.006809179
Argentina	2014	2780400	42980026	South America	76	444189	189189	0.004401789
Australia	2014	7741220	23460694	Oceania	82	1272519	438504	0.018691007

Figure 10: Visualizer's Table View allowing users to perform simple operations.

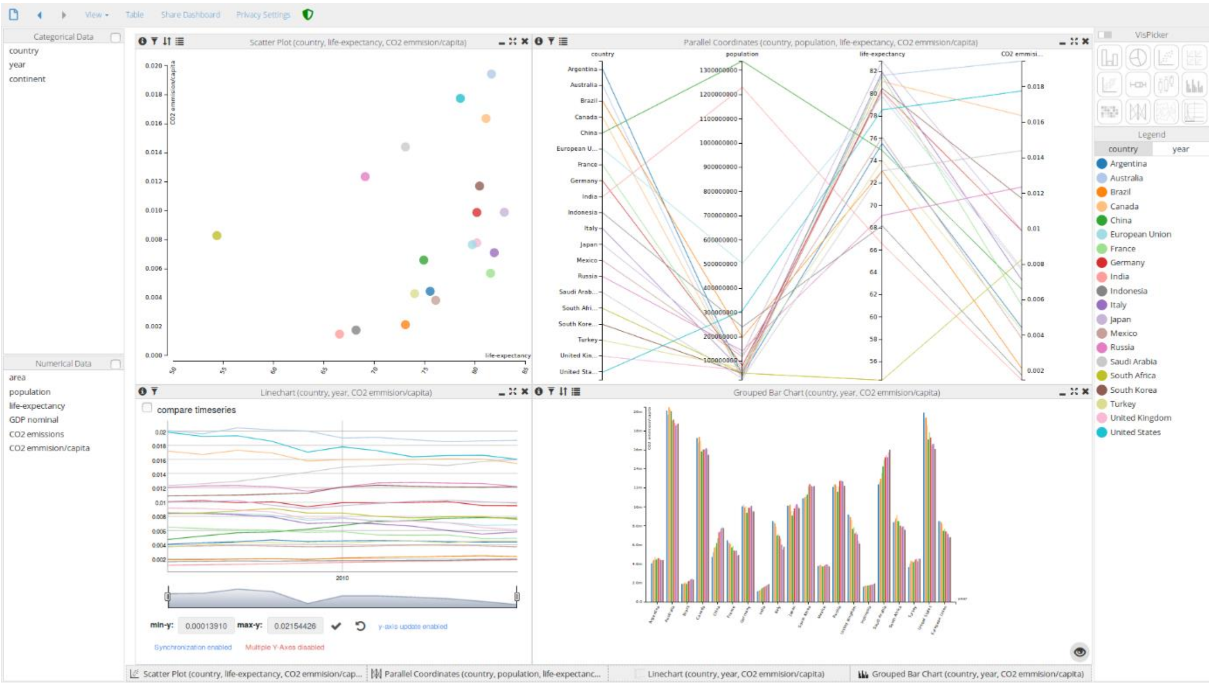


Figure 11: Visualizer's Dashboard View showing four coordinated visualizations.

5.1.2 Testing Environment

In order to benchmark the dashboard, the following notebook configuration was used:

Notebook: Lenovo ThinkPad T470P

CPU: 8 Intel ® Core ™ i7-8550U CPU @ 1,80 GHz 1.99 GHz

RAM: 16GB

System: 64 Bit operating system, x64-based processor

Operating System: Windows 10

Version: Windows 10 Pro

Browser: Chrome, latest version

5.1.3 Testing Procedure

For this dashboard two different features were benchmarked, which are described in the following. In order to identify outliers, each test was executed six times. The charts displayed below show the average values of those measurements. The dataset, which was used for testing contained 12 data columns and 986 data rows. It can be accessed online using the link http://samplecsvs.s3.amazonaws.com/Sacramentorealestatetransactions.csv.

For benchmarking, the available data rows were multiplied.

5.1.3.1 Loading data in the Table View

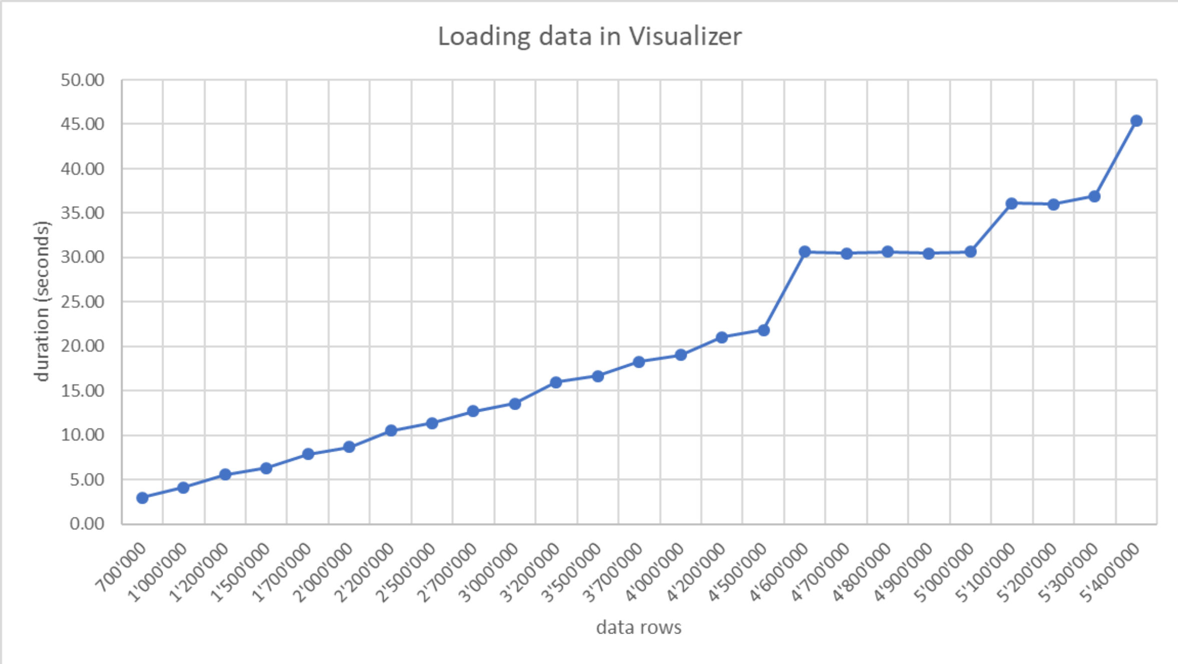


Figure 12: Average time in seconds for loading CSV files into Visualizer depending on the number of data rows.

Figure 12 shows the loading time in relation to the number of data rows, whereby the x-axis shows the number of data rows and the y-axis shows the corresponding time required for loading the data in seconds. It shows that loading time increases linearly except for 4,6 million data rows where it suddenly increases dramatically. This is also true for 5,1 million data rows. The dashboard was not able to load more than 5,4 million rows; therefore, no further measurements are displayed.

5.1.3.2 Opening an existing dashboard

For loading a dashboard, besides data loading, additional operations are required. These include data aggregation and visualization generation. For benchmarking, a dashboard containing three visualizations, a bar chart, a bubble chart and parallel coordinates, are generated. The resulting dashboard is shown in Figure 13.

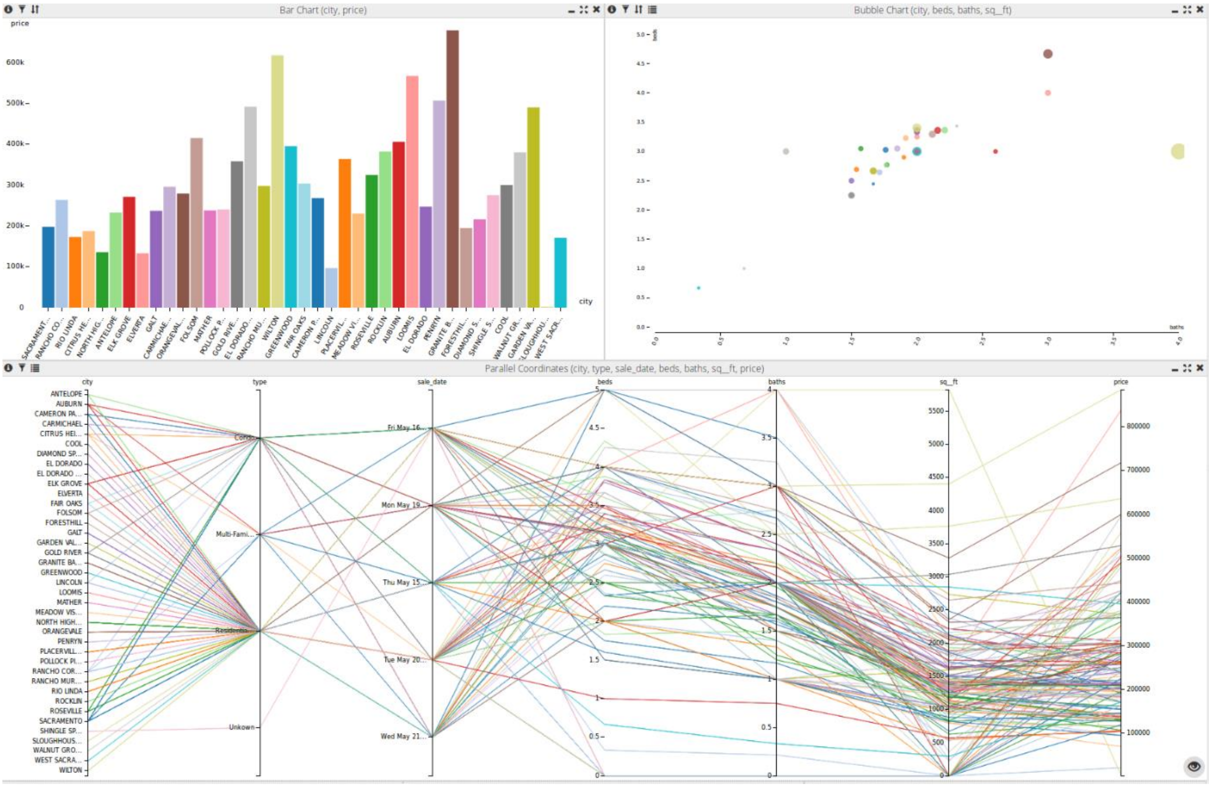


Figure 13. Dashboard used for benchmarking dashboard generation.

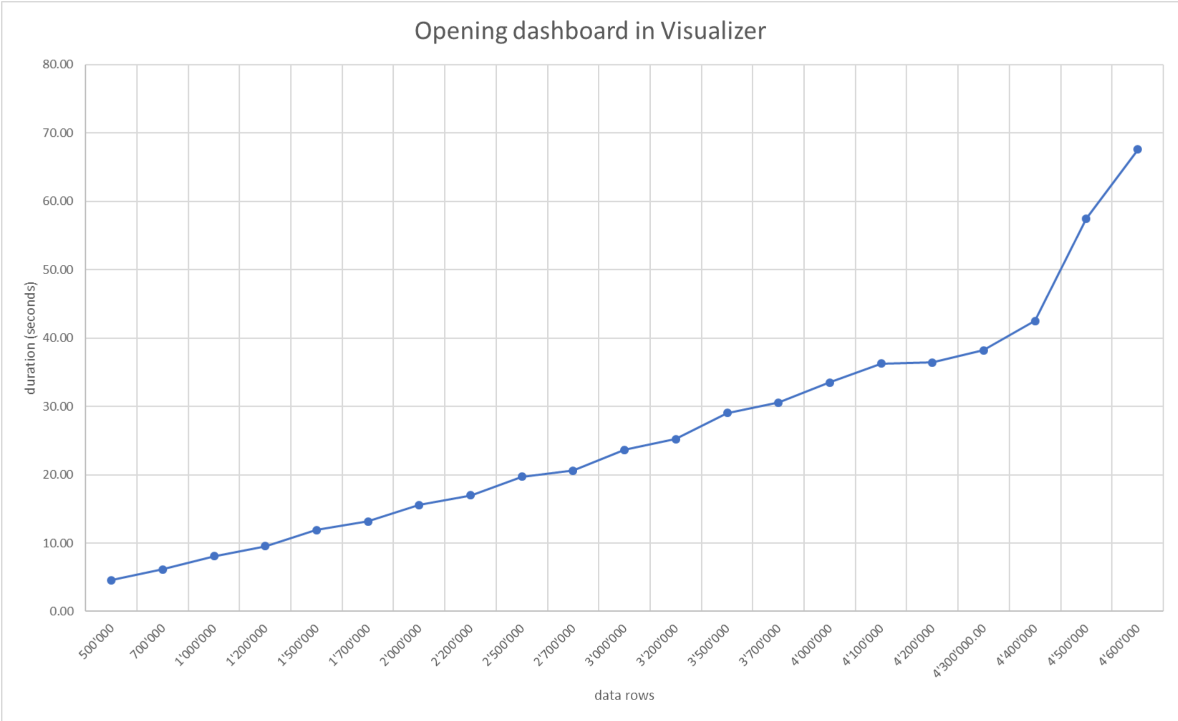


Figure 14: Average duration in seconds for loading a pre-defined dashboard in Visualizer depending on the number of data rows.

Compared to loading a dataset to the Table View, a similar pattern can be observed when loading a pre-configured dashboard to Visualizer. Performance decreases dramatically for more than 4.4 million data rows; it was not able to load the pre-configured dashboard containing more than 4.6 million data rows.

5.2 Time Series Analysis

5.2.1 Description

This tool allows users to investigate large-scale sensor data using the provided dashboard shown in Figure 15 and can be accessed online by using the link http://tsa.know-center.tugraz.at. Users can select available sensors, a time range and visualisations. The dashboard allows analysing data using three charts: combined line charts, single line charts and correlation charts (only for two sensors). Currently, it does not support monitoring live data, but could be extended, if required. Users can annotate their data, if they find a pattern of interest, which can be later used for suggesting new annotations. Depending on the selected time range, different data resolutions are loaded, see Figure 16. Furthermore, the tool allows users to filter for user defined value ranges, only showing those values, which are within the selected range.

Figure 15: Time series analysis dashboard for investigating and annotating large-scale sensor data.

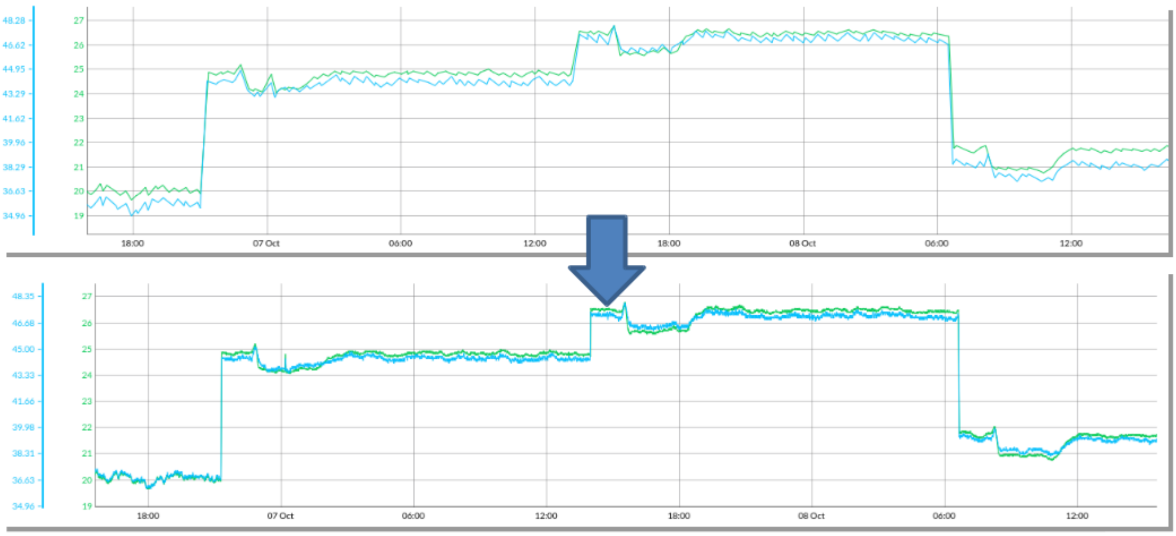


Figure 16: Zooming in, loads data in higher resolution.

5.2.2 Testing Environment

In order to benchmark all functionalities, the following server configuration was used:

CPU: 24 Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz

RAM: 264GB

System: 64 Bit operating system, x64-based processor

Operating System: Ubuntu 16.04.3 LTS

5.2.3 Testing Procedure

The following features were benchmarked for this tool:

- Data Conversion
- Data Downsampling
- Data Indexing
- Date Loading
- Data Filtering

5.2.3.1 Data Conversion

The original CSV data used for benchmarking has a size of 2,8GB split in 640 files each containing eight data columns with about 60000 data rows. The dataset can be accessed using the link http://archive.ics.uci.edu/ml/machine-learning-databases/00361/.

In order to convert data from CSV to the specific binary data format used for this tool (RTS2 - Regular Time Series File Format 2), data conversion was triggered five times. In average it took 117 seconds to convert the data. Measurements ranged from 108 seconds to 123 seconds. The resulting files have a total size of 1,9GB.

5.2.3.2 Data Downsampling

In order to support users in loading long time ranges within a short time period, data was downsampled in advance to provide multiple resolutions. Each downsampling operation was executed five times.

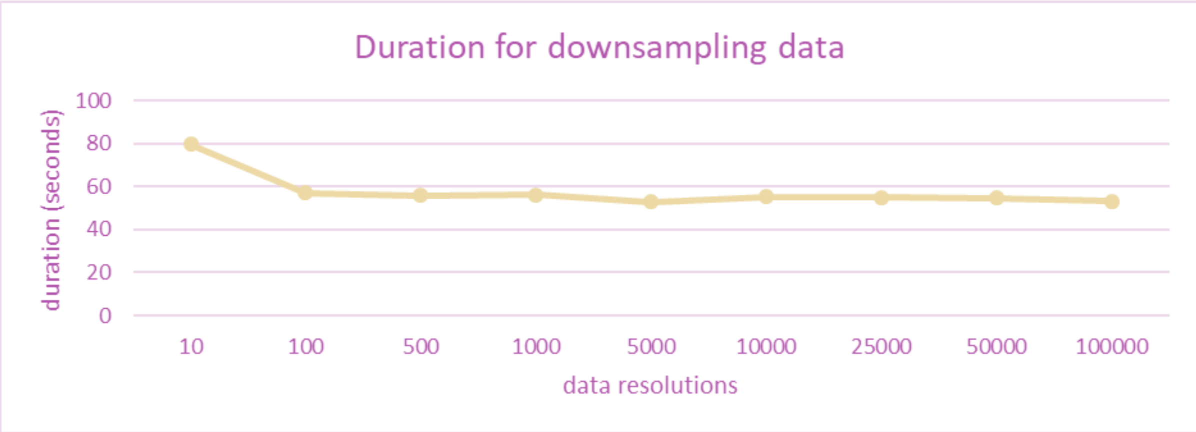


Figure 17: Average duration for sampling down the original data using different resolutions.

Figure 17 shows the average duration for sampling down the original data by a factor 10, 100, 500, 1.000, 5.000, 10.000, 25.000, 50.000 and 100.000. It shows that for reducing the data by a factor of 10, the average duration is comparable higher, whereas there are only small differences for all other downsampling calculations.

The algorithm used for downsampling is called Largest-Triangle-Three-Bucket algorithm [27]. The characteristics of this algorithm is, while reducing the number of datapoints, it preserves the signal shape. Figure 18 shows the data size for different downsampling resolutions. Each sensor is downsampled by the described factor in the x-axis, keeping those measurements which preserve the shape best. This can lead to the fact that different sensors have different time points, therefore the resulting size cannot be derived from the original size and the downsampling factor.

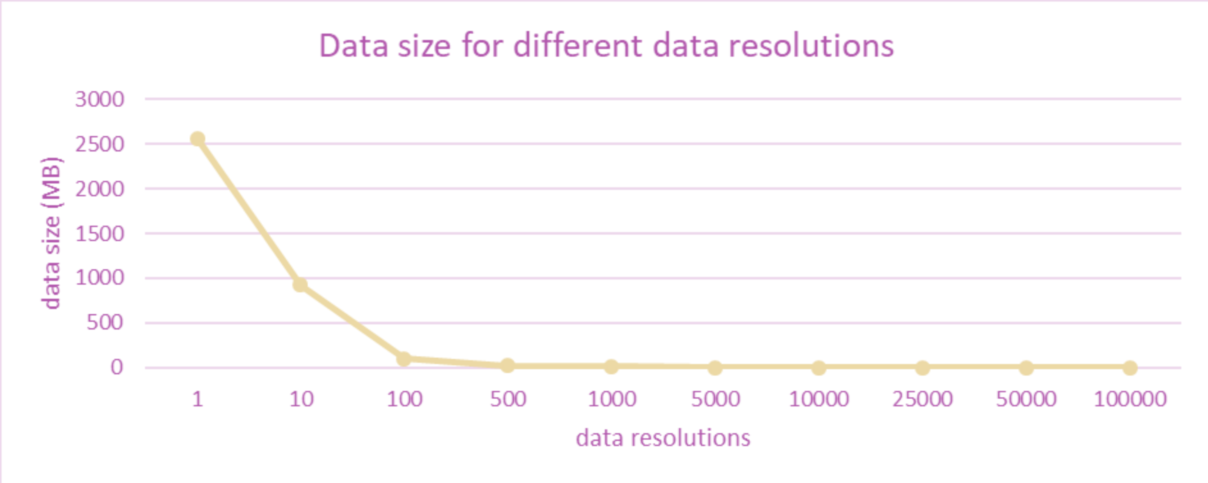


Figure 18: Data size for different downsampling resolutions.

5.2.3.3 Data Indexing

Data needs to be indexed in a search engine enabling users to filter for selected value ranges. In order to benchmark data indexing, the previously generated RTS2 files with a total size of 1,9GB were indexed.

data size (GB)	1,9
used RAM (GB)	100
indexing time (hours)	3:25:18
parallel threads	12
index size (GB)	17

Table 7: Performance for data indexing and index size for a 1,9 GB dataset used for value filtering.

Table 7 shows the performance for data indexing. It shows that data indexing requires close to 3,5 hours and the resulting index has a size of 17 GB.

5.2.3.4 Data Loading

In order to identify outliers, each configuration was executed three times.

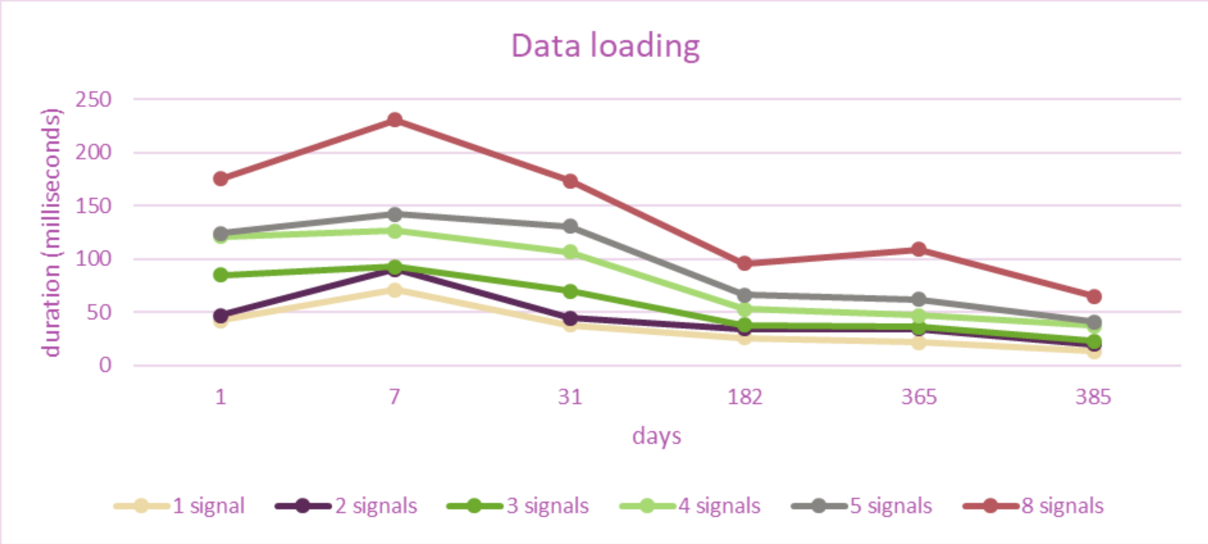


Figure 19: Average duration for loading data from the server depending on the number of signals; x-axis: selected time range in days; y-axis: time for loading data in milliseconds.

Figure 19 shows the average time in milliseconds for loading data from the server for a duration of 1, 7, 31, 182, 365 and 385 days depending on the number of selected sensors. The results show, that the more sensors are loaded from the server, the longer it takes. Whereas loading data for a longer time period does not necessarily take longer as the server does not retrieve the original data but a suitable resolution of the data.

5.2.3.5 Data Filtering

In order to identify outliers, each testing configuration was executed three times.

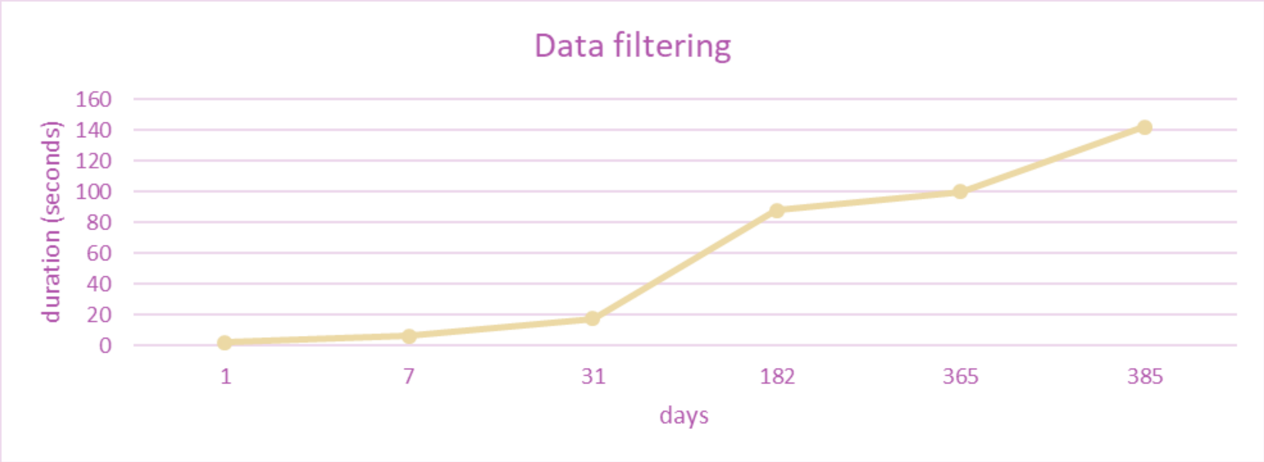


Figure 20: Average duration for filtering value ranges in one sensor depending on the selected time range; x-axis: time range in days; y-axis: filtering time in seconds.

Figure 20 shows the average time the server requires for filtering values in one sensor. It shows that filtering in few days requires only few seconds. This is significantly increasing when selecting a longer time range of e.g. 182 days. Thus, in order to support filtering for longer time ranges, improvements for filtering are required enabling users to analyse their data without longer delays.

5.3 Signal Search

5.3.1 Description

This tool enables users to investigate sensor measurements, select patterns of interest and search for similar results. Currently this tool is not publicly available, but it will be deployed for testing. In order to execute a search, different configurations can be applied leading to faster search results but may also decrease recall. The provided dashboard, see Figure 21, enables users to investigate search results in detail. Each search hit is highlighted in the line chart for the resulting sensors using different opacity levels depending on their relevance. Next to each line chart, an information box is displayed enabling users to navigate through all hits and investigate them in detail.

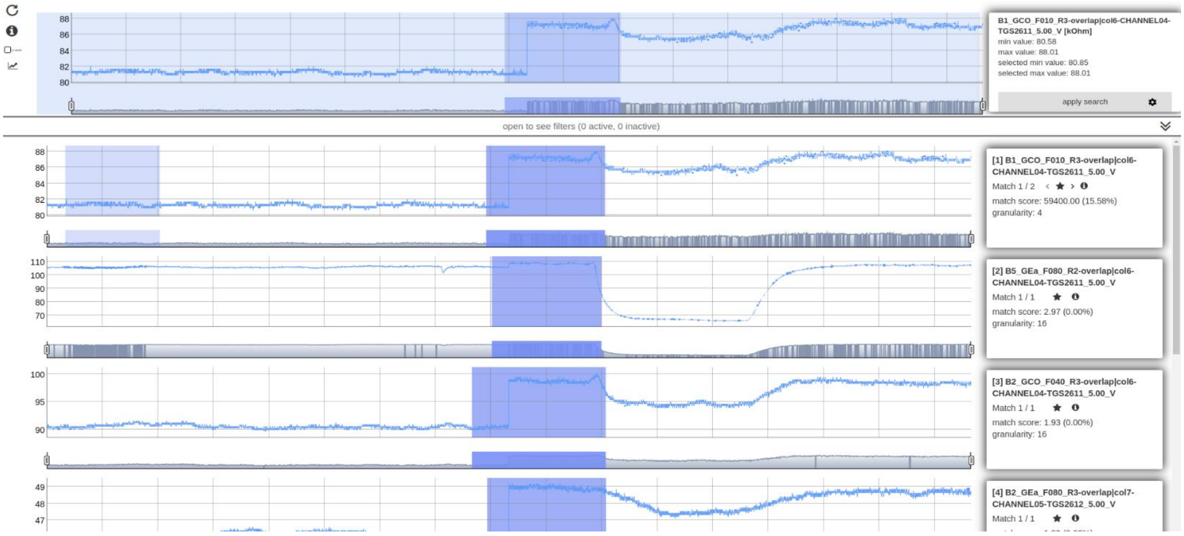


Figure 21: Dashboard for signal search and search result analysis.

5.3.2 Testing Environment

In order to benchmark all functionalities, the following server configuration was used:

CPU: 24 Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz

RAM: 264GB

System: 64 Bit operating system, x64-based processor

Operating System: Ubuntu 16.04.3 LTS

5.3.3 Testing Procedure

For this tool, two features were benchmarked:

- Data Indexing
- Signal Search

In order to benchmark this tool, the same dataset as for the previous tool was used.

5.3.3.1 Data Indexing

In order to make sensor data searchable, data needs to be indexed. Therefore, numeric data needs to be transformed to a letter representation. Thus, the value range of each sensor is spit up into different resolutions. Each measurement can then be assigned to one letter representation, which is further reduced by performing run length encoding.

In order to make these letters better searchable, they are duplicated by always summarizing a certain number of letters. For benchmarking, these number are 3 and 4. Table 8 shows the corresponding indexing time and index size. It shows that compared to 3-grams; 4-grams require more than five times longer. Also, the index size is slightly larger. Thus, investigating data indexing, indexing time for 4-grams is comparable bad, while they are supposed to have a better search performance. Therefore, finding a trade of between data indexing time and search time is required.

number of index fields	32	32
used RAM (GB)	20	100
parallel threads	8	8
n-grams	3	4
indexing time (minutes)	39	206
final index size GB	2,26	2,35

Table 8: Performance for data indexing and final index size for a 424,3MB dataset using different n-gram sizes.

5.3.3.2 Signal Search

For searching, sensor data was investigated and 34 different search queries were selected. Size in terms of datapoints ranged from 72 to 6568. Table 9 shows that search time for 4-grams is shorter than for 3-grams.

number of index fields	32	32
number of queries	34	34
used RAM	20 GB	20 GB
n-grams	3	4
search time (minutes)	59	50

Table 9: Performance for searching for 34 different queries using different n-gram sizes.

Summarizing results for both, data indexing and signal search: while larger n-gram sizes require much more time during data indexing, which is only performed once, search time is reduced significantly. Therefore, it is essential to find a good trade-off between the number of index fields, which in general produce a higher recall, the n-gram size and the search time. The more fields are indexed and the smaller the n-gram size, the longer searching requires.

5.4 Web Graph

5.4.1 Description

This visualisation [28] [29] allows users to investigate network data by initially visualizing only a relevant subset, while it enables users to navigate along interesting nodes and further explore the network. Therefore, this approach is not benchmarked but only described here.

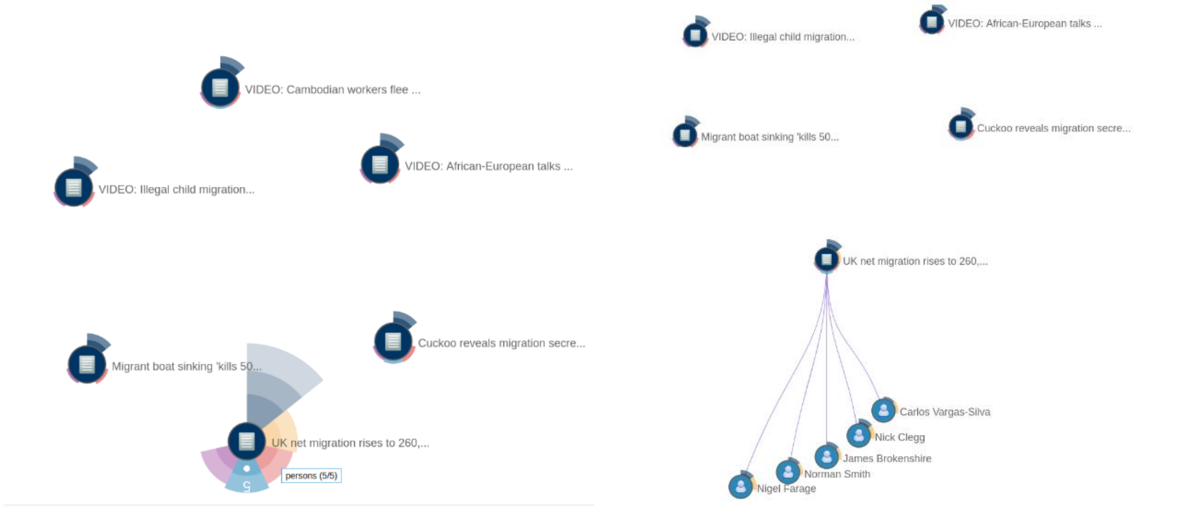


Figure 22: Initially an interesting subset of the data is displayed; each node contains connection information in its surroundings; mouse-over enlarges this information and enables users to open connected nodes.

The example displayed in Figure 22 shows the six most relevant search results for the query "migration and language: English" for news articles crawled between 2013 and 2016 from selected sources. The graph can be accessed by using http://search-demo.know-center.tugraz.at/search/WebgraphFrameFullHD.html, selecting "news" in the dropdown menu and providing a user query. It contains two types of nodes. On the one hand there are the news articles, which were indexed by the search engine, on the other hand there are extracted entities from those news articles. These can be persons, organisations, locations or others.

Each node contains connection information around the node icon, mouse-over on one of the nodes enlarges this information. This means, users can see how many similar articles are available or for example how many locations were extracted from the currently investigated article without showing all connected nodes. Starting from one article, users can open articles which are similar or extracted entities. This enables users to only investigate those nodes, which are interesting while hiding the complexity of the whole network.

Figure 23 shows different node and connection types within the graph. Each node type can be identified by a specific icon and colour. Additionally, the three different types of connections are shown:

- Two news articles are connected if their content is similar. Thus, this type of connection is called "similar to" and is coloured blue.
- A news article and an entity are connected if the entity was extracted from the news article. Thus, this type of connection is called "has property" and is coloured violet.
- Two entities are connected if they often occur together in documents within a small distance. Thus, this type of connection is called "associated with" and is coloured red.

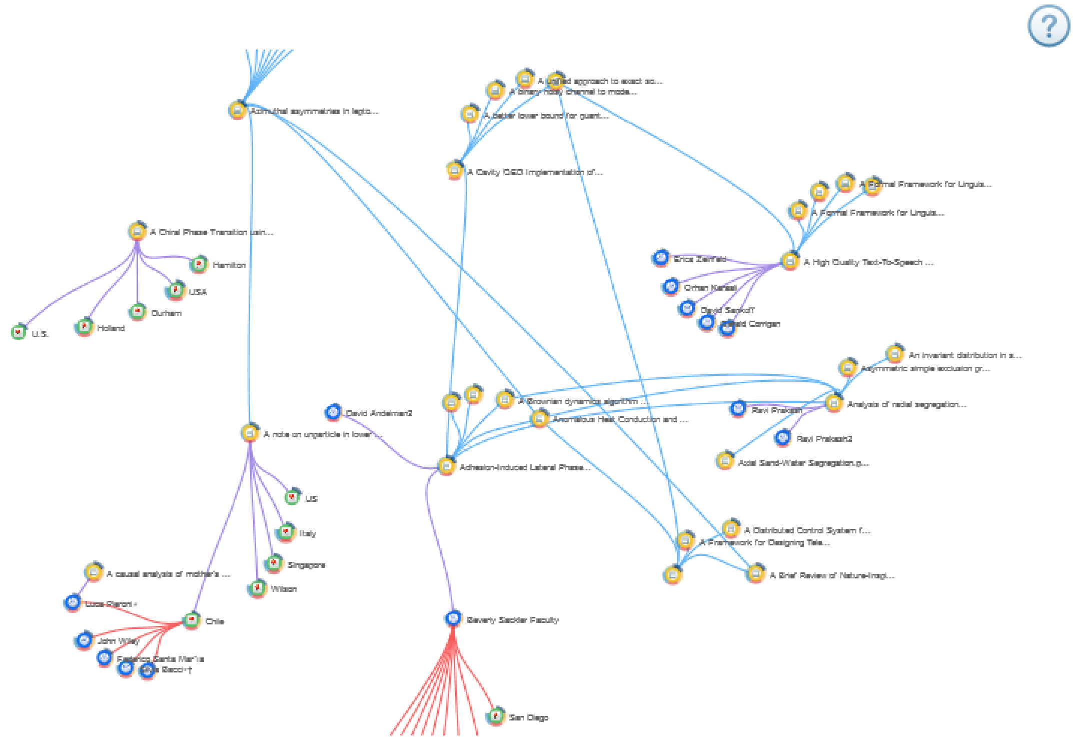


Figure 23. The web graph enables users to explore interesting nodes and connections while ignoring less relevant information and thus, hiding the complexity of the whole network.

6 Coupling technologies

Coupling technologies are used to combine different (existing) applications to make them work together for an overarching purpose. Coupling technologies are a staple in the multiscale and hybrid simulation approaches, and a range of generic technologies have emerged in recent years, each with their unique added values [30].

In this section we present our approach towards coupling technologies within the HiDALGO project from a technical perspective. These technologies are required because each of the three pilot applications will be coupled to at least two external models, and/or data sources, over the course of the project. We proceed first by summarizing the project-wide development choices that the consortium has agreed on (6.1), followed by an overview of the coupling characteristics, in terms of coupled models and required data exchange mechanisms (6.2). Based on the analysis in 6.2, we then conclude this chapter by positioning key technologies in accordance with the coupling requirements and road maps of the pilot applications (6.3).

6.1.1 Project-wide coupling development decisions

To effectively kick off the coupling task within this work package and ensure a consistent development philosophy across the consortium, we first agreed on a range of high-level design choices:

Establish a knowledge base: a key requirement to effectively implement model couplings is a clear awareness of the available coupling technologies, their advantages and drawbacks, and their intended scope of use. To establish this awareness across the consortium, we therefore have chosen to create an online directory for coupling knowledge resources on the internal OwnCloud platform.

Stratify per pilot: Although many coupling technologies are intended to be general purpose, we recognize that the optimal choice of coupling approach is strongly dependent on the way that models within an application interact with each other [30]. Because the pilot applications in HiDALGO are so distinctly different (e.g., in terms of code base size, or intervals of coupling interactions), we will find that different coupling technologies provide an optimal solution for the different applications. We therefore have chosen to stratify the coupling activities per pilot, and to appoint pilot leaders to take charge of the road map and prioritization for new couplings. Likewise, we will establish coupling data formats on a per use case basis.

6.1.2 Overview of planned coupled models and data formats

To clarify the nature of the planned couplings, we present a listing of the models that we plan to couple to each of the pilot applications, as well as the data formats that we are likely to adopt when linking these coupled models to the main application. In addition, we provide estimates for when we expect prototypes of each coupled model to be established, and when we aim to have defined each coupling data format. The listings provided here serve as a general guideline, not a constraint, because modified requirements or technological challenges faced later in the project should be reflected in the development timelines.

6.1.3 Migration

The migration pilot is intended to be coupled to a range of submodels and data sources. These include submodels and data sources for conflict propagation (Flare), weather forecast and telecommunications. Because migration simulations are an emerging domain, and almost no simulation approaches existed prior to 2016 [31], we will initially make prototype versions of these couplings, and harden the integrations in places where we discover that coupled submodels have a clear influence on performance or granularity of study. One important coupling that of a macro-scale migration model with a regional escape model, is not highlighted in this table. This is because that coupling activity is performed as part of the VECMA FET-HPC project. However, we will use (and further enrich if needed) that coupled model also as part of the HiDALGO pilot activities.

Overview of coupling		**Coupling communication characteristics**			**Planning**
Name of coupled submodel or data source	What kind of data is exchanged?	1-way or 2-way coupling?	Submodels run concurrently?	Envisioned coupling mechanisms (e.g. file I/O, DB, TCP or MPI)	Expected date of first working prototype.
Flare (conflict propagation)	A	1-way	No	File I/O	M6

Overview of coupling		Coupling communication characteristics			Planning
ECMWF (weather)	B or C	1-way	No	File I/O with basic processing@ECMWF	M12
ECMWF (weather)	B, C or D	1-way	No	REST API over HTTP, with processing@ECMWF	M24
Moonstar (telecomm.)	E+F	1-way	No	File I/O	M18
Moonstar (telecomm.)	E+F	2-way	Yes	File I/O	M30

Table 10. Migration: Overview of Planned Coupled Models

Overview of coupling			Coupling data format characteristics			Planning
ID	Name of coupling data format	What kind of data is exchanged?	Size per exchange	Expected frequency of exchange	Format type (e.g., CSV, HDF5, YML, bespoke ASCII, bespoke binary).	Expected date of first definition
A	Conflict Data	Conflict state by location and day.	kBs	Once per run.	CSV	M4
B	Weather Data	National precipitation and temperature	1-10 MBs	Once per simulated day	GRIB/YML	M10
C	Weather Data	Regional weather details	1-10MBs	Once per simulated hour	GRIB/YML	M15

Overview of coupling			Coupling data format characteristics			Planning
D	Weather Data	Extreme weather indicators	1-10 MBs	Once per simulated day	GRIB/YML	M15
E	Call Data	Call by time and endpoint countries	10-100 MBs	Once per simulated day	TBD	M20
F	Sim Data	Sim card requests by country and day.	1-10 MBs	Once per simulated day	TBD	M15

Table 11. Migration: Overview of Planned Coupling Data Formats

6.1.4 Social Networks Pilot

The social networks pilot will be coupled to a range of data sources. Snapshots of the social networks Pokec and Orkut are taken from SNAP (Standford Large Network Dataset Collection) [32]. Other social networks graphs derive from the well-known Twitter platform and the closed Hungarian social network IWIW. Concerning data due to social interaction, Tweets are retrieved with Twitter API and telecommunications data from Moonstar is used.

Overview of coupling		Coupling communication characteristics			Planning
Name of coupled submodel or data source	What kind of data is exchanged?	1-way or 2-way coupling?	Submodels run concurrently?	Envisioned coupling mechanisms (e.g. file I/O, DB, TCP or MPI)	Expected date of first working prototype.
SNAP	A	1-way	No	File I/O	M4, M12
Twitter	B, C	1-way	Yes	File I/O or DB	M10
Moonstar (telecomm.)	D	1-way	No	File I/O	M14
IWIW	E	1-way	No	File I/O	M12

Table 12. Social Networks: Overview of Planned Coupled Models

Overview of coupling			Coupling data format characteristics			Planning
ID	Name of coupling data format	What kind of data is exchanged?	Size per exchange	Expected frequency of exchange	Format type (e.g., CSV, HDF5, YML, bespoke ASCII, bespoke binary).	Expected date of first definition
A	SNAP social network	Pokec social network graph, Orkut social network graph	115 MB, approx. 350 MB	Once per run	Compressed text file in Metis format	M4, M12
B	Twitter social network	Social network graph	Approx. 800 MB	Once per run	Compressed text file in Metis format	M10
C	Tweets	Messages	n/a	Once per run	JSON files	M10
D	Moonstar	End points of voice terminations	n/a	Once per run	To be determined.	M14
E	IWIW	Social network graph	Approx. 110 MB	Once per run	Compressed text file in Metis format	M12

Table 13. Social Networks: Overview of Planned Coupling Data Formats

6.1.5 Urban Air Pollution

The Urban Air Pollution pilot will be coupled to three different models (a traffic and emission model, a meteorology model and a dispersion model), and three distinct data sources. In addition, we will establish several couplings to connect the post-processing software infrastructure to the main application.

The already existing and running version of the pilot uses implementation of the respective couplings for almost all cases except those corresponding to the sensor network, which is to be established during the project. In this version of the pilot the meteorological data correspond to a data set composed in the predecessor project of the pilot, the MSO4SC data set for the demonstration city; this data set contains weather data for a certain past period and will be completely replaced by the weather data set to be provided by ECMWF for HiDALGO. In general, all coupling methods of the pilot will be developed during the project life time and most of them will be fully re-implemented according to the tools and data of the HiDALGO infrastructure.

In the tables below dates for the couplings with the HiDALGO modules and data are provided. The planning takes into account the implementation of the sensor network and next milestones of the projects for new HiDALGO tools and data, in particular the milestone corresponding to new implementation of the pilots in M12.

Overview of coupling		Coupling communication characteristics			Planning
Name of coupled submodel or data source	What kind of data is exchanged?	1-way or 2-way coupling?	Submodels run concurrently?	Envisioned coupling mechanisms (e.g. file I/O, DB, TCP or MPI)	Expected date of first working prototype.
traffic measurement	A	1-way	yes	TCP, file I/O, MPI (all 3 mechanism will be used)	M12
traffic database	B	1-way	no	file I/O from DB	M10
traffic and emission model	C	1-way	yes	MPI	M12
meteorology model (data	A	1-way	yes	TCP	M12

Overview of coupling		Coupling communication characteristics			Planning
from the sensor network)					
meteorology database and service	B	1-way	no	file I/O from DB, MPI	M12
dispersion model	C	2-way	yes	MPI	M12
postprocessing – indicators	B, D	1-way	yes, no	MPI	M12
postprocessing – graphics	B, D	1-way	yes	MPI	M12

Table 14. Urban Pollution: Overview of Planned Coupled Models

Overview of coupling		Coupling data format characteristics			Planning
Name of coupling data format	What kind of data is exchanged?	Size per exchange	Expected frequency of exchange	Format type (e.g., CSV, HDF5, YML, bespoke ASCII, bespoke binary).	Expected date of first definition
A	online measured or provided simulation data	10 MB (meteo) or 1 MB (traffic)	6 hours (meteorology data) and 15 minutes (traffic)	NetCDF, CSV	M12
B	measured data in DB	10 MB	(almost) continuously	CSV	M12
C	HPC/HPDA computed and postprocessed	1 MB - 100 GB or even	(almost) continuously	CSV, HDF5	M12

Overview of coupling		Coupling data format characteristics			Planning
	data	more			
D	postprocessing of HPC computed data	100 MB	6 hours – 2 days	vtu (paraview)	M12

Table 15. Urban Pollution: Overview of Planned Coupling Data Formats

6.1.6 Initial findings and technological implications

In this Chapter we presented the coupling roadmap for each of the three applications, which encompass a total of 16 couplings between main pilot solvers and other submodels or data sources. As indicated by the roadmaps of the three pilots, we intend to prototype most of the model couplings already in year 1. Particularly for the social media and urban pollution use case, this means that early implementations of the coupled models will be functional at least to some extent by M12. Because a large number of couplings are expected to be established in the first year for these applications, enabling these functionalities will be a priority in the remainder of Year 1, while we will seek to optimize for performance and flexibility after M12. For the migration pilot, the roadmap is substantially different. A coupling to the Flare conflict evolution model is now available in prototype form, and we will add a new coupled model every sixth month. Because of the emerging nature of the migration modelling domain, this slightly slower roadmap progression gives us more time to validate the couplings against available observational data, and to establish meaningful settings in which these couplings indeed contribute to more accurate forecasts for the application overall.

In terms of technology, we initially need to direct our efforts towards establishing good file-based couplings, and converge on data formats that are flexible and consistent within the context of each pilot. For the migration pilot, we expect to be able to rely on highly efficient file I/O for most of our couplings. However, in the case of the coupling with weather data we will require a working integration with the REST API at ECMWF. In addition, and not in the tables, the integration with smaller scale escape models in the context of VECMA (www.vecma.eu) poses some important technological requirements when scaled up to more detailed location graphs. Within VECMA, we have established this coupling using files, but we will need to rely on TCP-based coupling if we want to scale up this coupling to larger core counts. It is our intention to establish these optimized and scalable TCP couplings in the second and third year of the project, and to assess the benefit of using these TCP couplings in the integration with other migration-related submodels.

In the case of the social media pilot, snapshots of social graphs are used to verify the developed graph clustering mechanism. Static graph processing will be sufficient for this use case. Regarding simulating social interaction, regular data retrieval is desired together with a regular data flow to the pilot application. Thus, we will look for mechanism to integrate real-time data from Twitter's Streaming API into our pilot application. Together with the mobile communication data, it will lead to complex event processing and it will bring hopefully interesting combination effects.

The urban pollution pilot is perhaps the most demanding application in terms of technical coupling requirements. It features very high frequencies of exchange in the couplings, and here the file exchange coupling mechanism will be a temporary solution in many cases, to help prepare the ground for establishing higher-performance coupling approaches. Among the solutions that we are currently considering are TCP- and MPI-based coupling, as well as file-based coupling using an external database.

In terms of tools, we identify a range of software platforms that we can apply for these applications. Although we are still in the process of identifying the most suitable solutions in many cases, we are considering for instance the use of iReS [33] and FabSim [34] in the case of workflow coupling, and MPI directly or the MUSCLE toolkit [35] in the case of network coupling. However, new coupling tools emerge at a very high rate [30], so our final choice of coupling approach is not fixed, and should take into account the latest developments that will be reported (among other places) in D4.2 (M12) and D4.3 (M24).

7 Conclusions

This document provided initial strategies for optimising applications and implementing novel algorithms and methods. In particular, initial strategies for coupling applications in conjunction with WP4 are presented.

In respect of performance investigation (Chapter 2) of simulation tools and due to complexity of the process we were able to provide only short update on performed actions.

As a preparation phase for implementation data analytics we investigated several frameworks (Spark, Hadoop, Flink, Dask and R) and provided valuable benchmarking solutions along with tests results. These results will be used for selection a relevant framework for specific procedures have to be implemented for data analytics process of use case data.

Later, an elaboration on requirements for data managements system is provided. Moreover, first tests findings on performance and system optimization are presented.

In the chapter related to visualization four software tools are introduced and benchmarked. These are capable of visualizing tabular data, analyse large-scale time series data and signal search which facilitate visual analysis of investigated data by users.

Coupling technologies are discussed in the latter section. After final elaboration they will be used for combining different applications to make them work together for an overarching purpose.

7.1.1 General remarks & lessons learned

Deliverable D3.2 presents HiDALGO benchmarking findings. In respect of HPDA methods we learned about potential tools and libraries for development. Thanks to the test findings we know their capabilities and which tool should be chosen for specific implementation. The same is for visualization part. Currently four tools are available, designed for different presentation purposes, the others are coming (e.g. COVISE).

Data management systems are an important part of the project infrastructure. The selection procedure must be well thought out. Thanks to this elaboration we know more about first candidate: CKAN. Rucio as another potential framework will be researched in the next order.

Information about coupling technologies is in the initial phase but state a good starting point for further investigation which at the end could bring number of benefits for the project workflow.

7.1.2 Next steps

We are going to further develop and investigate the performance of presented applications and approaches particularly:

- Profiling of simulation tools on HPC systems
- Develop data analytics methods,
- Assess and improve data analytics performance
- Investigate Rucio as potential framework for data management in the project.
- Research other visualization tools for high demanding data presentation, especially COVISE, developed by HLRS
- Continue work on aspects related to coupling technologies to present consistent perspective which provides added value to the project work

References

[1] "The Intersection of AI, HPC and HPDA," [Online]. Available: https://www.top500.org/features/content/the-intersection-of-ai-hpc-and-hpda-how-next-generation-workflows-will-drive-tomorrows-breakthroughs/.

[2] "Configuration of the Spark," [Online]. Available: https://spark.apache.org/docs/latest/configuration.html.

[3] "Explaining Hadoop Configuration," [Online]. Available: https://www.edureka.co/blog/explaining-hadoop-configuration/.

[4] "HiBench Readme file," [Online]. Available: https://github.com/IMCG/HiBench/blob/master/README.md.

[5] "spark-bench: Benchmark Suite for Apache Spark," [Online]. Available: https://github.com/CODAIT/spark-bench.

[6] "HiBench: The bigdata micro benchmark suite," [Online]. Available: https://github.com/Intel-bigdata/HiBench.

[7] "spark-perf: Spark Performance Tests," [Online]. Available: https://github.com/databricks/spark-perf.

[8] "BigDataBench: A Scalable Big Data and AI Benchmark Suite for IoT, Edge, Datacenter and HPC, Chinese Academy of Sciences & BenchCouncil," [Online]. Available: http://prof.ict.ac.cn .

[9] "Sanzu version 0.7: A Data Science Benchmark," [Online]. Available: http://bigdata.cs.unb.ca/projects/sanzu.

[10] "Dask Benchmarks: A suite of benchmarks for dask-related projects," [Online]. Available: https://github.com/dask/dask-benchmarks.

[11] A. Watson, D. Shree Vittal Babu and S. Ray, "Sanzu: A Data Science Benchmark," *IEEE BigData,* pp. 263-272, 2017.

[12] M. Rocklin, "Dask Benchmarks," [Online]. Available: https://matthewrocklin.com/blog/work/2017/07/03/scaling.

[13] M. Rocklin, "Dask Benchmarks," [Online]. Available: https://gist.github.com/mrocklin/4c198b13e92f881161ef175810c7f6bc#file-scaling-gcs-ipynb.

[14] J. Daniel , "Data Science at Scale with Python and Dask," *Manning Publications,* no. 325 p., 2019.

[15] "Dask Array," [Online]. Available: http://docs.dask.org/en/latest/array.html.

[16] "Dask DataFrame," [Online]. Available: http://docs.dask.org/en/latest/dataframe.html .

[17] "Prof. Philippe Grosjean website," [Online]. Available: http://www.sciviews.org/team/philippe-grosjean/.

[18] "Dask Scheduling," [Online]. Available: http://docs.dask.org/en/latest/scheduling.html .

[19] "Dask Single-Machine Scheduler," [Online]. Available: https://docs.dask.org/en/latest/setup/single-machine.html .

[20] R. Vesse , A. Heye and M. Ringenburg, "Python based Data Science on Cray Platforms," Cray Inc, 2017. [Online]. Available: https://www.ecmwf.int/sites/default/files/elibrary/2017/17838-python-based-data-science-cray-platforms.pdf.

[21] "Cray Urika XC: Analytics & AI Software Suite," Cray Inc, 2018. [Online]. Available: https://www.cray.com/sites/default/files/Cray-Urika-XC-Brochure.pdf.

[22] "CKAN," [Online]. Available: https://ckan.org.

[23] "Rucio," [Online]. Available: https://rucio.cern.ch/.

[24] "CoeGSS," [Online]. Available: http://coegss.eu/.

[25] I. Šimić, "VISUALIZER - An Extensible Dashboard for Personalized Visual Data Exploration," *Master Thesis, Graz University of Technology,* 2018.

[26] M. Rauch , A. Gaal , I. Šimić and V. Sabol , "Evaluation of Visual Decision Support Systems used in Semiconductor Industry," *19th European Advanced Process Control and Manufacturing (apcm) Conference,* 2019.

[27] S. Steinarsson , "Downsampling Time Series for Visual Representations," *Master Thesis, University of Iceland,* 2013.

[28] M. Rauch , R. Wozelka , E. E. Veas and V. Sabol , "Semantic Blossom Graph: A New Approach for Visual Graph Exploration," *18th International Conference on Information Visualisation,* p. 234–240, 2014.

[29] M. Rauch , W. Klieber , R. Wozelka , S. Singh and V. Sabol , "Knowminer Search – A Multi-visualisation Collaborative Approach to Search Result Analysis," *19th International Conference on Information Visualisation,* p. 379–385, 2015.

[30] D. Groen, J. Knap, P. Neumann, D. Suleimenova, L. Veen and K. Leiter, "Mastering the scales: a survey on the benefits of multiscale computing software," *Philosophical Transactions of the Royal Society A,* vol. 377(2142), no. p.20180147, 2019.

[31] D. Groen, "Simulating refugee movements: Where would you go?," *Procedia Computer Science,* no. 80, pp. 2251-2255, 2016.

[32] "SNAP Datasets: Stanford Large Network Dataset Collection," [Online]. Available: https://snap.stanford.edu/data/index.html.

[33] K. Doka, N. Papailiou, D. Tsoumakos, C. Mantas and N. Koziris, "Ires: Intelligent, multi-engine resource scheduler for big data analytics workflows," *ACM SIGMOD International Conference on Management of Data,* pp. 1451-1456, May 2015.

[34] D. Groen, A. Bhati, J. Suter, J. Hetherington, S. Zasada and P. Coveney, "FabSim: facilitating computational research through automation on large-scale and distributed e-infrastructures," *Computer Physics Communications,* vol. 207, pp. 375-385, 2016.

[35] J. Borgdorff, M. Mamonski, B. Bosak, K. Kurowski, M. Belgacem, B. Chopard, D. Groen, P. Coveney and A. Hoekstra, "Distributed multiscale computing with MUSCLE 2, the multiscale coupling library and environment," *Journal of Computational Science,* vol. 5(5), pp. 719-731, 2014.