Horizon 2020
European Union funding
for Research & Innovation

D5.7 – First report on provided testbed components for running services and pilots

Grant Agreement	676547
Project Acronym	CoeGSS
Project Title	Centre of Excellence for Global Systems Science
Topic	EINFRA-5-2015
Project website	http://www.coegss-project.eu
Start Date of project	October 1, 2015
Duration	36 months
Deliverable due date	31.03.2017
Actual date of submission	18.04.2017
Dissemination level	Public
Nature	Report
Version	1.22
Work Package	WP5 (Centre Operation)
Lead beneficiary	PSNC
Responsible scientist/administrator	Norbert Meyer (PSNC)
Contributor(s)	Michael Gienger (HLRS), Sebastian Petruczynik (PSNC), Radosław Januszewski (PSNC), Damian Kaliszan (PSNC), Sergiy Gogolenko (HLRS), Steffen Fuerst (GCF), Michał Pałka (Chalmers), Enrico Ubaldi (ISI)
Internal reviewer	Leonardo Camiciotti (Top-IX), Steffen Fuerst (GCF)

Keywords	GSS, Benchmarks, HPC, scalability, efficiency, exascale
Total number of pages:	75

The CoeGSS ("Centre of Excellence for Global Systems Science") project is funded by the European Union. For more information on the project please see the website http://www.coegss-project.eu/

Version History

	Name	Partner	Date
From	Norbert Meyer	PSNC	28.02.2017
Version 0.3	Damian Kaliszan	PSNC	24.03.2017
Version 0.4	Norbert Meyer	PSNC	26.03.2017
Version 0.4	Norbert Meyer	PSNC	26.03.2017
Version 0.9	Norbert Meyer	PSNC	28.03.2017
Reviewed by	Leonardo Camiciotti Steffen Fuerst	TOP-IX GCF	29.03.2017 31.03.2017
Updated Version 1.22	Norbert Meyer	PSNC	14.04.2017
Approved by	ECM		18.04.2017

1. Abstract

The work package WP5 CoeGSS (Centre Operation) and task 5.2 (Preparing the Future) has the following aims:

- To setup the centre`s technological baseline
- To operate and maintain the centre's infrastructure
- To evolve, operate and maintain the CoeGSS Portal
- To enable access to leading edge technology.

Task 5.2 is especially aiming to show the vision of future ICT technologies (both hardware and software) which may help to perform GSS applications better, make it easier for users to run applications and build them from the existing or new components. Finally, the GSS applications, which will require an enormous capacity of HPC systems and higher capacity data infrastructures (20-100 PFlops and PBs capacity), will be able to check whether the trends of future hardware development will allow acceleration of the computing and help GSS to solve the defined problems. Deliverable D5.7 (First report on provided testbed components for running services and pilots) is checking in an experimental way which HPC architecture and processor are best for the GSS community. The experimental way is defined by a set of exemplary applications called benchmarks.

The GSS provides evidence about global systems, for example about the network structure of the world economy, energy, water and food supply systems, the global financial system, the global city system, and the scientific community.

It is a new area of research that is trying to find an answer about rules which can be used in the global world, the global market or the global financial system.

The CoeGSS project has defined three exemplary challenges as pilot studies: Health Habits, Green Growth, Global Urbanisation. In addition, the set of benchmarks in D5.7 was extended with additional examples:

- Iterative proportional fitting (IPF)
- Data rastering – a preprocessing process converting all vectorial representations of georeferenced data into raster files to be later used as simulation input.
- Weather Research and Forecasting (WRF) model.

The rational behind this decision was to define a set of tests which may reflect the real GSS applications and requirements, i.e using applications that did not scale easily, these ones operating at the preprocessing stage and large models that would ultimately require significant computing power.

D5.7 is the first report in a series presenting results of benchmarks with an aim to find the most suitable HPC architecture and the best processor which allows to run GSS applications effectively.

2. Table of Contents

3. Introduction

The vision of **GSS (Global System Science)** is to provide scientific evidence to support policy-making, public action and civil society to collectively engage in societal actions.

The policy makers suffer from an intrinsic difficulty when addressing challenges like climate change, financial crises, governance of pandemics, or energy sufficiency: the impact and unintended consequences of public action are increasingly hard to anticipate. Such challenges are global and connect policies across different sectors. When faced with such highly interconnected challenges, societies still tend to address subsystems and so fail to achieve systemic change.

GSS can drive change by

- Helping develop an integrated policy perspective on global challenges; and
- Developing a research agenda that will tackle the fundamental research challenges.

A case of point in the area of urban dynamics and climate change where a combination of data from various sources (smart grids, mobility data, sensor data, socio-economic data, etc.) with dynamical modelling will pave the way to new policy suggestions. Other policy areas include economic modelling after the financial crisis. New concepts and tools – for instance to analyse the network of actors in financial markets – will be developed in collaboration between researchers in GSS and policy bodies.

In computing, a **benchmark** is the act of running a computer program, a set of programs, or other operations, in order to assess the relative **performance** of an object, normally by carrying out a number of standard tests and trials against it. The term 'benchmark' is also mostly utilized for the purposes of elaborately designed benchmarking programs themselves.

Benchmarking is usually associated with assessing performance characteristics of computer hardware, for example, the floating point operation performance of a CPU, but there are circumstances when the technique is also applicable to software. Software benchmarks are, for example, run against compilers or database management systems.

Benchmarks provide a method of comparing the performance of various subsystems across different chip/system architectures.

Test suites are a type of system intended to assess the correctness of software (1).

The known benchmarks include:

- Industry standard (audited and verifiable)
  - Business Applications Performance Corporation (BAPCo) (2)
  - Standard Performance Evaluation Corporation (SPEC), in particular their SPECint and SPECfp (3)
  - Transaction Processing Performance Council (TPC) (4)
- Open source benchmarks
  - Bonnie++: filesystem and hard drive benchmark
  - DEISA Benchmark Suite: scientific HPC applications benchmark (5), (6)
  - HINT: designed to measure overall CPU and memory performance
  - LINPACK benchmarks, traditionally used to measure FLOPS

- LAPACK
- NAS parallel benchmarks
- NBench: synthetic benchmark suite measuring performance of integer arithmetic, memory operations, and floating-point arithmetic
- Parsec: a benchmark for parallel shared memory systems
- Splash2: a benchmark for parallel systems.
    - STREAM: a benchmark for measuring memory bandwidth of a system.

Many benchmarks focus entirely on the speed of computational performance, neglecting other important features of a computer system, such as:

- Qualities of service, aside from raw performance. Examples of unmeasured qualities of service include security, availability, reliability, execution integrity, serviceability, scalability (especially the ability to quickly and non-disruptively add or reallocate capacity), etc.
- In general, benchmarks do not measure Total cost of ownership, e.g. Transaction Processing Performance Council Benchmark specifications partially address this concern by specifying that a price/performance metric must be reported in addition to a raw performance metric, using a simplified TCO formula.
- Facilities burden (space, power, and cooling).

Deliverable D5.7 is the first report that will make it possible to verify the use of new HPC architectures for running GSS applications. However it focuses here on computational performance and the end user references and not the parameters important from the service provider point of view, like the mentioned above. The next deliverable D5.8, to be released in month33, is a second stage of this analysis with improved benchmarks and new computing architectures. Therefore, it was decided to select a list of exemplary applications from the CoeGSS project and several others from the broader research area of the entire GSS environment.

So far, there has been a lack of these lists which we could call ‘GSS benchmark’. In addition there is no extensive activity in the use of HPC architectures for this type of applications (7).

The section 4 describes codes used by GSS communities. In fact, GSS is a new research area, which has not used extensively HPC platforms for doing any analysis so far. The CoeGSS is therefore a challenging approach trying to work out methods, procedures and finally a programming framework that would allow users to use HPC in an innovative and effective way. Because GSS has not been extensively used high performance computing there are a lot of open questions, e.g. set of good predefined applications which could represent the entire community. The section 4 gives a first approach of benchmarks representing the project end users:

- Green Growth pilot using the Pandora library
- Application for data rastering

and in general the whole GSS community

- Iterative proportional fitting
- Data rastering process as preprocessing computing
- The Weather Research and Forecasting.

Each subsection was divided into two parts:

- including a description and meaning of the application which was nominated for the benchmark
- and a detailed guide how to use the test in different programming environments.

Section 5 describes various trends of high performance computing represented by new CPU architectures and computing nodes developed by various hardware vendors. The hardware platforms were chosen among the newest ones from PSNC and HLRS. It was also possible to get remote access to some pre-production clusters, e.g. ARM based or KNL.

Finally, section 5 describes the following architectures:

- Intel Xeon E5-2697 v3 100-node cluster as the reference architecture (Haswell)
- Intel Xeon E5-2682 v4 50-node cluster (Broadwell)
- Intel Xeon Phi 7250 2-node cluster
- ARM 2-node cluster
- IBM Power8 8247-22L single node.

Intel Haswell is used as the reference architecture and starting point of all tests.

Chapter 6 presents results of benchmarks for all the above-mentioned hardware platforms. Output parameters measured for all tests are as follows:

- **%e** Elapsed real time (in seconds; not in tcsh) (8)
- **%M** Maximum resident set size of the process during its lifetime, in Kbytes
- **%I** Number of file system inputs generated by the process
- **%O** Number of file system outputs generated by the process.

Each of the benchmarks is performed as a function of the number of threads. It is to test also the ability to balance and the scalability on each of the architectures. The description of the tests themselves should be considered as a starting point not only for the D5.7 report but also for the next one to be conducted in month M33 - D5.8 (Second report on testbed components for running services and pilots).

Therefore, in D5.8, we can expect further development of both benchmarks as well as an increase in the number of tested architectures with an emphasis on scalability for a really large analysis.

Section 7 concludes the report.

3.1 Glossary of Acronyms and Terms

Acronym	Definition
ARM	Advanced RISC Machine
Cannonlake	Intel's codename for the 10-nanometer die shrink of the Skylake microarchitecture, expected to be released in the second half of 2017
CPU	Central Processing Unit
DoA	Description of Action
EC	European Commission
FLOPS	Floating Point Operations per Second
GB	Gigabyte
Gbps	Gigabit per second
GDDR5	Graphic Double Data Rate v5
GPU	Graphics Processing Unit
HLRS	High Performance Computing Center Stuttgart
HPC	High Performance Computing
IPF	Iterative proportional fitting
Knight Landing	Knights Landing (KNL)
KNL	Knights Landing (KNL): 2nd Generation Intel® Xeon Phi™ Processor
MPI	Message Passing Interface
NVIDIA	American technology company based in Santa Clara, California. NVIDIA designs graphics processing units (GPUs)
OpenPOWER	The name of a range of servers in the System p line from IBM. They featured IBM's POWER5 CPUs and run only 64-bit versions of Linux
PSNC	Poznan Supercomputing and Networking Center
Purley	New Xeon Skylake Platform
RAM	Random Access Memory
SDRAM	Synchronous Dynamic Random Access Memory
Skylake	Intel’s new generation CPU
SLURM	Simple Linux Utility for Resource Management , Workload Manager
TFLOP/s	TeraFLOPS per second
WP	Work Package
WRF	The Weather Research and Forecasting

4. Description of benchmarks used for GSS representation

4.1 Green Growth using Pandora library

Working towards an agent-based model for studying the car-centred global system in view of a green growth transition, the green growth pilot first implemented an innovation-diffusion model for electric cars with a global scope and a fine-scale spatial data resolution. As described in D4.4 (9), there exists a deterministic and a stochastic implementation of this model. For benchmarking the stochastic implementation was used.

In this preliminary version of the green growth pilot, we consider only two classes of cars: “green"” cars, which for now correspond to battery electric vehicles due to data availability, and “brown"” ones. Cars are distributed on a gridded global map, that is, in the first instance “agents” are simply cells in the grid, which automatically specifies a neighbourhood network between them, and their characteristics are a number of brown and one of green cars. Each cell at each time step computes the number of new cars to be added from the total numbers exogenous given and a percentage of cars being scrapped.

The model dynamics account for two effects:

- Innovation: as electric cars are already on the market, from time to time somebody will buy such a car for a variety of reasons. However, in order to buy an electric car people need a certain income. Other factors being equal, the higher their income, the higher the share of electric cars bought by innovators.
- Imitation: the more electric cars are present in a given neighbourhood, the higher the probability that a consumer chooses a green car. This represents observations by this consumer and takes the number of green cars already present in this neighbourhood as an indicator of the existence of an electric-car-friendly infrastructure.

Data from several sources have been collected, pre-processed, and integrated into a gridded global map:

- The UN-adjusted population count grids consist of estimates of the number of persons per 30 arc-second (˜1 km) grid cell and adjusted to match the United Nations country totals (Socioeconomic Data and Applications Center (10).
- The indicator “cars per 1000 people” (OICA, 2015) is the most important one to evaluate a country’s contribution to the global fleet of cars (11).
- GDP data per country is provided by the World Bank (2015) and has been available for most countries in recent years (12).

Apart from the spatial input data, GDP per capita, population count and cars per 1000 people, the models require two additional parameters:

- η determines the share of “innovators” that buy green cars independently of the observed neighbourhood. To reflect the purchasing power of different countries and the higher costs of green cars, the share of innovators is modified based on the country’s GDP per capita.

- κ determines resistance to adapt for the "imitators".

The calculation for the number of new green cars at a single time step depends on the share of existing green cars s_{exists} in an extended Moore-Neighbourhood with a Chebyshev distance of r. The influence $w_{x,y}$ of a cell in the neighbourhood of the cell at location $\hat{x}, \hat{y}$ is reduced by its Euclidean distance, with $w_{\hat{x},\hat{y}} = 1$ and $w_{x,y} = 0$ for cells outside the Moore-Neighbourhood. Using those influence weights the "visible" share of green cars $s^{vis}_{\hat{x},\hat{y}}$ and the value $s^{new}_{\hat{x},\hat{y}}$ is calculated by:

$$s^{\text{vis}}_{\hat{x},\hat{y}} = \frac{\sum s^{\text{exist}}_{x,y} \cdot w_{x,y}}{\sum w_{x,y}}$$

$$s^{\text{new}}_{\hat{x},\hat{y}} = \eta G + (1 - \eta G)(s^{\text{vis}}_{\hat{x},\hat{y}})^{\kappa}$$

$$\text{with: } G = \frac{\text{GDP per Capita}_{\text{country of location } \hat{x},\hat{y}}}{\text{GDP per Capita}_{\text{world average}}}$$

The total number of new cars per cell $n_{x,y}$ is exogenously given.

In the deterministic implementation used for the benchmarking our problem is that the calculated number of new green cars is a real number instead of an integer. The solution used was to add the fractional portion $r_{x,y}$ to the value calculated in the next step:

$$n^{\text{green}}_{x,y} = \lfloor n_{x,y} \cdot \min(s^{\text{new}}_{x,y}, 1) + r_{x,y} \rfloor$$

The basic dynamic HPC model has been implemented using the HPC-ABM framework Pandora (13). For parallelization Pandora uses a spatial partition schema, with evenly divided rectangular sections of the world. Information in the border of sections is communicated to the adjacent processes via the MPI. The communication between agents is limited to the size of the border.

Using data (2005-2015) and scenarios (2016-2025) for population density on a global map and for cars per 1000 inhabitants per country, scenarios computed on the basis of a model from the literature(14), model simulations deliver maps as the one shown in Fig. 1.

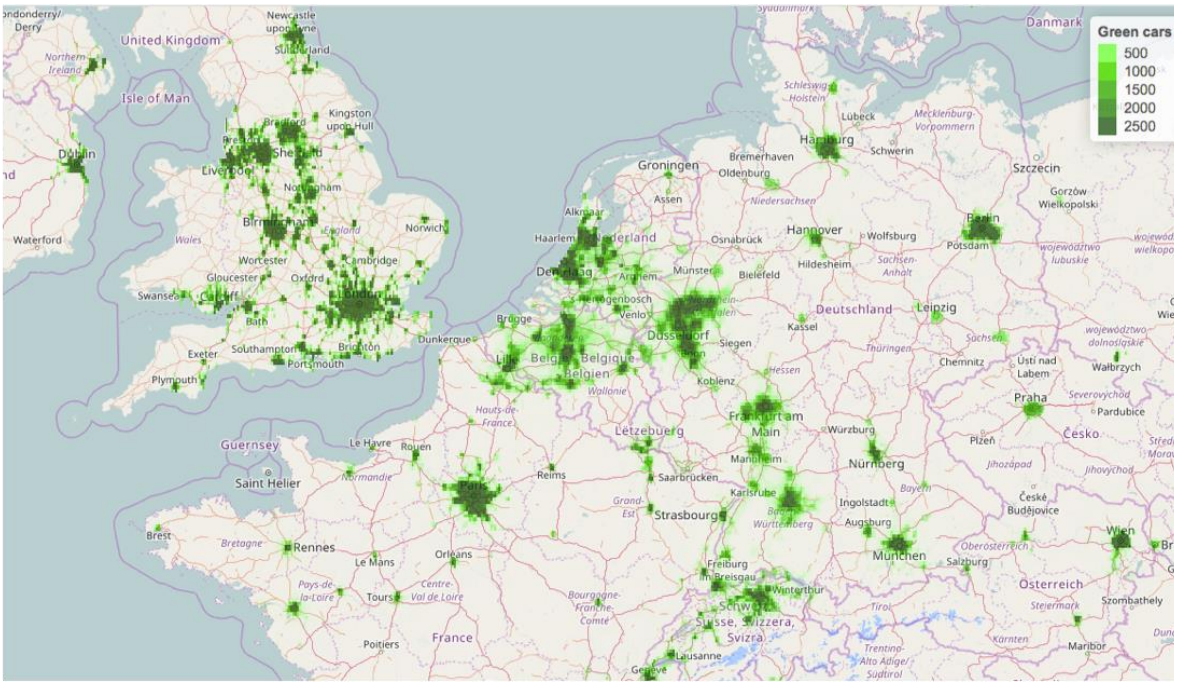

Fig. 1 Simulation output of the preliminary green growth pilot model: green cars per 5 x 5 km cell in 2025.

As one example, this map suggests that regional niches can be helpful in fostering electric mobility. Further model developments shall include complexifying the model to take into account more and more components of the car-centered global system (15).

Use case description

The main goal of the benchmarking process was to test the Pandora library against various available architectures and check its behaviour and possible scalability. The other issue was to check if scientists benefit from applying the novelty hardware.

The first step was to compile the library itself using SCons (16) as the main building tool. It is open-source, Python-based and next generation substitution of *make* utility. Version 2.4.1 of the tool was used because the newest version is not compatible with the *build* input file (SConstruct). For the build process Pandora requires three additional, external libraries: HDF5, GDAL and BOOST. For the task purposes version 1.8.18 of HDF was used and configured in the following way:

```
CC=mpicc ./configure --prefix=/usr/local/hdf5 --enable-parallel
```

In the case of GDAL we used 1.11.5 for similar reasons to SCons. GDAL is a Geospatial Data Abstraction Library and presents a single raster abstract data model and single vector abstract data model to the calling application for all supported formats. It also comes with a variety of

useful command line utilities for data translation and processing. The library was configured in the following manner:

```
CC=mpiicc CXX=mpicxx ./configure --prefix=/usr/local/gdal -with-sse=yes --with-hdf5=/usr/local/hdf5
```

Finally, Pandora installation required a BOOST C++ library which was built and configured as follows:

```
./bjam --with-mpi
./bootstrap.sh --prefix=/usr/local/boost
./b2 --prefix=/usr/local/boost install
```

Depending on the hardware and system capabilities we had the following MPI versions on-board:

- Intel(R) MPI Library for Linux* OS, Version 2017 Update 1 Build 20161016 (id: 16418) (2-node KNLs 7250)
- Open MPI 1.10.2 (ARM)
- impi 5.0.3.048 (Eagle)

Input data for benchmark has been delivered as two maps (Europe and World) with two different resolutions `840x680` and `8640x3432`. To evenly distribute the map chunks (rasters) across mpi processes the helper python function (calculator) was created to print all available possibilities of map division for the given architecture and the number of processes.

```
architectures = [32]
map_sizes = [[840, 680], [8640, 3432]]
num_procs = []
if __name__ == '__main__':
  for node in range(1, 101):
    for arch in architectures:
      num_procs.extend([arch*node])
  for proc in num_procs:
    print("# Computing all possible combinations for "
       "{0} procs:".format(proc))
    for size in map_sizes:
      width = size[0]
      height = size[1]
      print("## Map size [{0}x{1}]: nodesPerRow, "
         "nodesPerColumn, [rasterX x rasterY]".format(
           width, height)
         )
      for nodesPerRow in range(2, proc):
        if (proc % nodesPerRow == 0):
          nodesPerColumn = proc / nodesPerRow
          rasterX = int(width/nodesPerRow)
          rasterY = int(height/nodesPerColumn)
          if (
              rasterX % 2 == 0 and
              rasterY % 2 == 0 and
              rasterX >= 8 and
              rasterY >= 8):
            bb = False
            for i in num_procs:
              if int(nodesPerColumn)*nodesPerRow == i:
                bb = True
```

```
            if bb:
                print(
                  "{0}, {1}, [{2}x{3}]".format(
                    nodesPerRow, int(nodesPerColumn),
                    rasterX, rasterY))
```

What the calculator actually does is to check the following conditions:

- number of processes `mod` nodesPerRow is 0,
- rasterX = `int`(width/nodesPerRow)
  rasterY = `int`(height/nodesPerColumn)
- rasterX `mod` 2 is 0, rasterY `mod` 2 is 0, rasterX > 8, rasterY > 8

As for different architectures (and thus number of mpi processes) we obtain different map division, so it may happen that only part of the map will be calculated.

`int(width/nodesPerRow)*nodesPerRow x int(height/nodesPerColumn)*nodesPerColumn`

points

4.2 IPF

Iterative proportional fitting (IPF) is a procedure that reconstructs a contingency matrix based on the known marginals in an unbiased way. IPF can be applied to multidimensional matrices, but the multidimensional case is a simple generalization of the two-dimensional case on which we base our presentation.

IPF fits the matrix coefficients so that the sums of elements of every row and every column are equal to the given respective marginals. The initial contingency matrix values are either all , or are based on some known correlation data (such as a micro sample). In general, IPF solves an underspecified problem, and its result is the maximum likelihood estimation assuming that the elements of the matrix are drawn from the Poisson distribution given by the initial contingency matrix.

In the two-dimensional case, the procedure starts with a matrix A[] and two vectors R[] (row marginals) and C[] (column marginals), and performs a number of iterations. A single iteration is performed as follows. First, the sums of elements in every row are computed. Then, every row is scaled by the division of the corresponding element from R[] and the sum of elements from that row. Thus:

$$A'[i, j] = A[i, j] * (R[i] / A[i, \_])$$

where A[i, _] denotes the sum of all elements from row i. Then the same procedure is repeated for the columns.

$$A''[i, j] = A[i, j] * (C[j] / A[\_, j])$$

If there are not too many zeros, the procedure will converge, and will typically do it after several iterations.

IPF applied to a matrix that has more dimensions performs the same iteration procedure executing an updating step for every dimension in turn.

IPF is often performed to fit a contingency matrix to known marginals as one step of synthetic population generation. If a lower-dimension contingency matrix is known, their values can be used as marginal for IPF, which then performs fitting that matches the correlations defined by this contingency matrix.

The IPF procedure is often followed by **sampling**, which uses the values from the reconstructed contingency matrix as weights. It may also be followed by the Iterative Proportional Updating (IPU) procedure, which uses reconstructed contingency matrices for individuals and households, and a micro sample of households containing individuals to perform individual-household assignment.

Use case description

The benchmarked IPF implementation performs the IPF procedure for a 2-dimensional matrix.

The implementation is programmed in C and uses the MPI and PBLAS APIs. MPI and PBLAS are common HPC APIs whose implementations are available on many platforms. On typical Linux systems they can be provided by the OpenMPI, OpenBLAS and Scalapack open-source packages. On HPC systems they are typically provided by proprietary libraries, such as the Cray MPI library, the Cray libsci library or the Intel MKL library.

The implementation uses MPI to launch computing processes on different computational nodes, and perform message-based communication between them. The PBLAS API, which builds on top of MPI, provides operations on distributed vectors and matrices, such as computing a sum of a distributed vector (pdasum) or scaling a distributed vector (pdscal), both of which are used by the implementation.

The PBLAS operations assume that the distributed vectors and matrices are laid out according to the one- or two-dimensional block-cyclic distribution. The distribution is parametrised by the block size, which affects the performance of the operations performed on the vectors and matrices.

The implementation performs iterations until the maximum-norm error between the target marginals and the marginal computed from the matrix falls below a threshold value.

For testing purposes IPF has been compiled against ScaLAPACK and OpenBLAS. On all testbeds they were installed in the latest available versions: 2.0.2 and 0.2.19, respectively. The installation process requires straightforward steps:

`python setup.py --prefix=/path/to/install/scalapack -downall` for ScaLAPACK

`make TARGET= PREFIX=path/to/install/openblas install` for OpenBLAS

When built, the IPF application can be now compiled :

```
mpicc -o ipf ipf.c -L/home/damian/scalapack/lib -lscalapack -
L/home/damian/openblas/lib -lopenblas –lgfortran
```

and run in the following way:

```
./ipf PROCROWS PROCCOLS ROWBLOCK COLBLOCK TESTITERS
```

The number of MPI processes must be equal to `PROCROWS*PROCCOLS`. `ROWBLOCK` and `COLBLOCK` are set independently and should be equal to power of 2.

4.3 Data rastering application

One of the pilots' requirements is the data pre-processing task. Part of this procedure consists of adding spatial information to data available in a tabular form (e.g., binding the value of each row of a csv file to a particular region or local authority in a country).

While some input data are defined at a global or country level (e.g., the price of gasoline for cars or the price of a packet of cigarettes in a country) other inputs have a finer resolution. For example, the population density in the world has been mapped in 1kmx1km cells by the SEDAC project (17)and can be used as a reliable proxy to estimate the number of people living in a certain area. More specifically, the SEDAC data define a global grid of squared cells of about 1km width and height, giving for each of them its position in latitude and longitude coordinates and a value representing the number of people living in that particular area. This representation of data is called a raster and it can be thought as a pixel-map covering the surface under investigation giving a value for each pixel (area). Another possible representation of geographical entities is the vectorial format, in which shapes (boundaries) are saved into shapefiles or geojson files. The latter are a collection of shapes (usually the borders of regions or local authorities) each containing a vector of one or more keys-values tuples recording different indicators. For example, a shapefile may contain information about the average income, smoking prevalence, or population count for each region inside a country (18).

Depending on the particular application one representation may be more suitable than the other, thus requiring an automated and optimized procedure to convert data from one format to the other.

Indeed, each representation has its own pros and cons. Specifically, the vectorial representation allows for a fine description of each region borders and usually allows for a quick lookup of the features of a single region as each record in the file has its own specific unique identifying code.

On the other hand, the raster file is more cumbersome to handle because of its squared regular lattice of points: each cell is identified only by its row/column position.and one does not know a-priori the fraction of a given cell that falls within a specific region (the cell may cross the border between two or more confining regions). This work has to be manually done in a pre-

processing part of the analysis procedure. The strength of the raster files comes when considering their easy implementation in parallel simulations, as they provide a natural way to split the work of different threads by leveraging on the regular lattice they feature. Indeed, the Pandora simulation framework takes advantage of a regular lattice division of the computing load. That is why we first have to convert all the vectorial representations of georeferenced data into raster files to be later used as simulation input.

Given the need to run multiple simulations with different scenarios and, possibly, change some input rasters from one simulation replica to the other, as much as possible of the rasterization process should be done in an automated and general way.

In our implementation, we cover the steps of the following procedure:

- match csv files with shape-file records so as to create a vectorial map containing the data we need for simulation;
- convert the generated shapefile to a raster whose reference system complies with the reference raster of the SEDAC population count;

The conversion from vectorial geo-layer (shapefile or geoJson) to rasters comes with some technicalities regarding the value to assign to cells laying on the boundaries of the shapes: by now we apply the simplest approach possible, i.e. we insert in the raster the average value of all the regions intersecting with the cell under consideration.

Following this pre-processing procedure the next development would account for post-processing of the simulation results and be based on various possible interpolation functions.

Use case description

Selected rastering application is based on two IPython scripts which can be opened inside the Jupyter notebook web application and then launched. The apps environment allows sharing scripts between users, data visualization if appropriate graphic module is loaded. Jupyter can also be used to export files to regular Python scripts. For benchmarking reasons this path was selected.

The virtual environment for Python 2.7 was created and activated using `virtualenv` command. The additional modules were also installed using pip.

```
virtualenv -p python jupyter2

source jupyter2/bin/activate

pip install numpy pandas pyshp matplotlib xlrd jupyter
```

The `agents` version is based on the Pandora library but it is still in the preliminary version and could have been tested in a single process mode. The calculator used in GG application for map division reflecting the number of MPI processes cannot be applied here.

The benchmarking is using not only the rastering application of the Health habit, but also the model itself.

4.4 Weather Research and Forecasting model

The Weather Research and Forecasting (WRF) model is an example of quite well scalable application. Therefore it was added to the set of tests increasing the variety of requirements of the "GSS benchmark".

Hurricane Katrina formed as Tropical Depression Twelve over the southeastern Bahamas on August 23, 2005, as the result of an interaction of a tropical wave and the remains of Tropical Depression Ten. It strengthened into Tropical Storm Katrina on the morning of August 24. The tropical storm moved towards Florida, and became a hurricane only two hours before making landfall between Hallandale Beach and Aventura on the morning of August 25. The storm weakened over land, but it regained hurricane status about one hour after entering the Gulf of Mexico, and it continued strengthening over open waters. On August 27, the storm reached Category 3 intensity on the Saffir-Simpson hurricane wind scale, becoming the third major hurricane of the season. An eyewall replacement cycle disrupted the intensification, but caused the storm to nearly double in size. The storm rapidly intensified after entering the Gulf, growing from a Category 3 hurricane to a Category 5 hurricane in just nine hours. This rapid growth was due to the storm's movement over the "unusually warm" waters of the Loop Current.

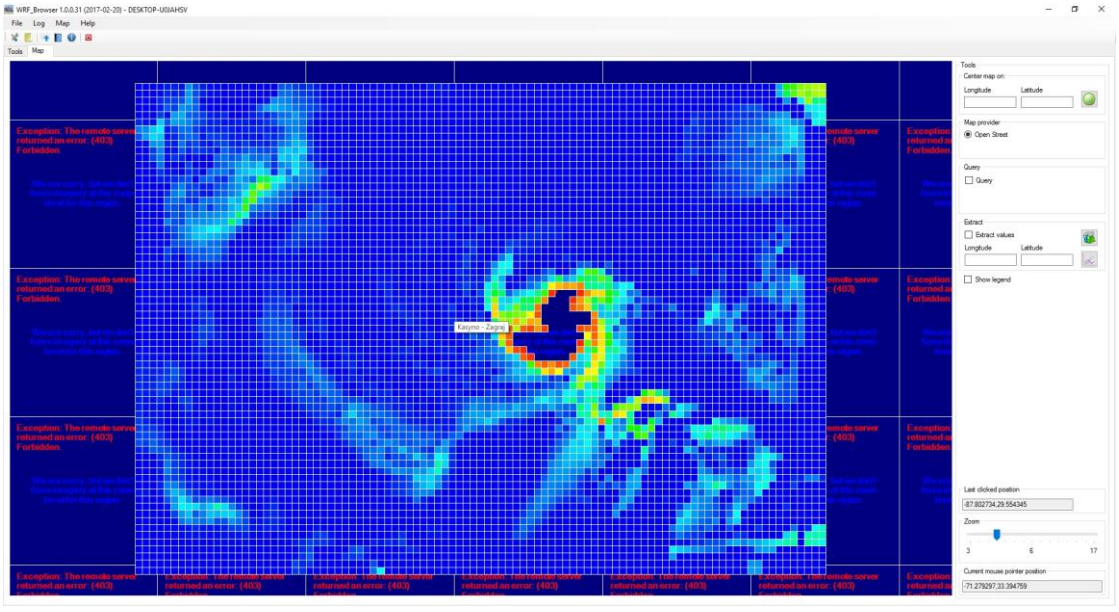

Fig. 2 The result of WRF simulation for Katrina hurricane

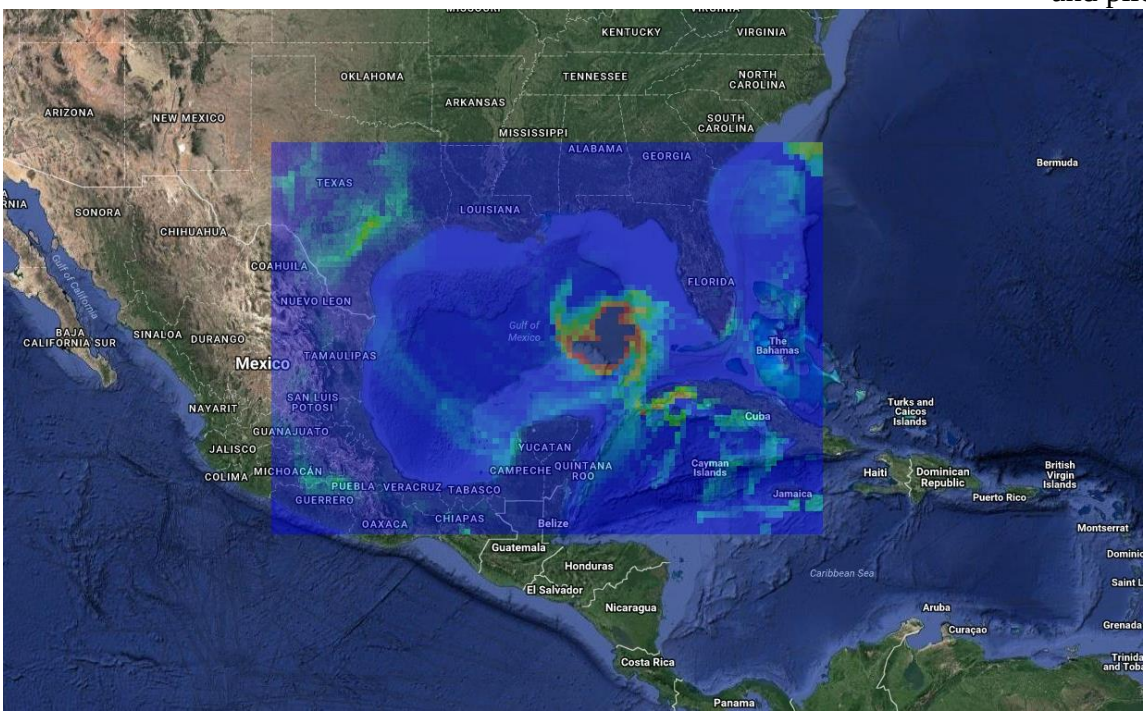

Fig. 3 The result of WRF simulation for Katrina hurricane put on Google Earth map

Katrina attained Category 5 status on the morning of August 28 and reached its peak strength at 1800 UTC that day, with maximum sustained winds of 175 mph (280 km/h) and a minimum central pressure of 902 mbar (26.6 inHg). The pressure measurement made Katrina the fourth most intense Atlantic hurricane on record at the time, only to be surpassed by Hurricanes Rita and Wilma later in the season; it was also the strongest hurricane ever recorded in the Gulf of Mexico at the time. However, this record was later broken by Hurricane Rita. The hurricane subsequently weakened, and Katrina made its second landfall at 1110 UTC on August 29 as a Category 3 hurricane with sustained winds of 125 mph (200 km/h) near Buras-Triumph, Louisiana. At landfall, hurricane-force winds extended outward 120 miles (190 km) from the center and the storm's central pressure was 920 mbar (27 inHg). After moving over southeastern Louisiana and Breton Sound, it made its third landfall near the Louisiana–Mississippi border with 120 mph (190 km/h) sustained winds, still at Category 3 intensity. Katrina maintained strength well into Mississippi, finally losing hurricane strength more than 150 miles (240 km) inland near Meridian, Mississippi. It was downgraded to a tropical depression near Clarksville, Tennessee, but its remnants were last distinguishable in the eastern Great Lakes region on August 31, when it was absorbed by a frontal boundary. The resulting extratropical storm moved rapidly to the northeast and affected eastern Canada (19).

Hurricane WRF software

The Hurricane Weather Research and Forecasting (HWRF) model is a specialized version of the Weather Research and Forecasting (WRF) model and is used to forecast the track and intensity of tropical cyclones. The model was developed by the National Oceanic and Atmospheric Administration (NOAA), the U.S. Naval Research Laboratory, the University of Rhode Island, and Florida State University.

This HWRF modeling system, based on the NMM (non-hydrostatic mesoscale model) core of the WRF (Weather Research and Forecasting) model, is a high resolution coupled air-sea-land prediction model with a movable nested grid and with the applicability to the prediction problems of hurricane track, intensity, structure, and rainfall. The model uses a two-way interactive movable nested grid that follows the forecasted path of tropical cyclone.

Use case description

Hurricane WRF use case..

1. Download software
2. Compilation of dependencies
3. Compilation of weather model
4. Download Hurricane Katrina dataset
5. Run

1. Download software libraries:
   - Jasper Library
   - HDF5 Library
   - netCDF
   - zlib
   - HWRF model: http://www.dtcenter.org/HurrWRF/users/downloads/index.php

2. Compilation of dependencies:
   - Jasper Library: (https://www.ece.uvic.ca/~frodo/jasper/)
     ./configure --prefix=$PREFIX
     make
     make install
   - HDF5:
     ./configure --prefix=$PREFIX --enable-parallel
     make check install
   - zlib:
     ./configure --prefix=$PREFIX
     make
     make install
   - netCDF:
     ./configure --prefix=$PREFIX --with-zlib
     make check install

3. Compilation of HWRF weather model (http://www.dtcenter.org/HurrWRF/users/docs/users_guide/HWRF_v3.8a_UG.pdf):
Download all 9 archives:
   • HWRF_v3.8a_WRFV3.tar.gz
   • HWRF_v3.8a_WPSV3.tar.gz
   • HWRF_v3.8a_UPP.tar.gz

- HWRF_v3.8a_GSI.tar.gz
- HWRF_v3.8a_hwrf-utilities.tar.gz
- HWRF_v3.8a_gfdl-vortextracker.tar.gz
- HWRF_v3.8a_ncep-coupler.tar.gz
- HWRF_v3.8a_pomtc.tar.gz
- HWRF_v3.8a_hwrfrun.tar.gz

The tar files can be unpacked by use of the GNU gunzip command:

gunzip *.tar.gz

and the tar files extracted by running:

tar -xvf

individually on each of the tar files.
The User should first unpack hwrfrun, which will create a directory hwrfrun/ in ${SCRATCH}. Then within ${SCRATCH}/hwrfrun/sorc/ directory, unpack the remaining tar files. Once unpacked, there should be eight source directories in sorc/.

- WRFV3 – Weather Research and Forecasting model
- WPSV3 – WRF Preprocessing System
- UPP – Unified Post-Processor
- GSI – Gridpoint Statistical Interpolation
- hwrf-utilities – Vortex initialization, utilities, tools, and supplemental libraries
- gfdl-vortextracker – Vortex tracker
- ncep-coupler – Ocean/atmosphere coupler
- pomtc – Tropical cyclone version of MPIPOM

Building WRF

Export following variables:
export HWRF=1
export WRF_NMM_CORE=1
export WRF_NMM_NEST=1
export JASPERLIB=~/jasper/lib/
export JASPERINC=~/jasper/include/
export NETCDF=~/NetCDF
export WRFIO_NCD_LARGE_FILE_SUPPORT=1
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:~/NetCDF/lib/
Load following modules:
module load icc/17.0.1 ifort/17.0.1 impi/2017.1.132 mkl/2017.1.132

./configure -> choose your architecture (20. (dmpar) for Intel Xeon)
./compile nmm_real 2>& 1 | tee build.log

A successful compilation produces two executables listed below in the directory main/ :

- real_nmm.exe - WRF initialization
- wrf.exe WRF - model integration

If a recompilation is necessary, a clean to remove all object files should be completed:

./clean

A complete clean is strongly recommended if the compilation failed, the configuration file has been changed, or the configuration file is changed. To conduct a complete clean that removes all built files in all directories, as well as the configure.cpl, use the "-a" option:

./clean -a

Building WPS

Set up the build environment for WPS by setting the WRF_DIR environment variable:

export WRF_DIR=${SCRATCH}/hwrfrun/sorc/WRFV3/

Change to the WPS directory and issue the configure command:

cd /hwrfrun/sorc/WPSV3
./configure -> choose your architecture (19. (dmpar) for Intel Xeon)
./compile 2>&1 | tee wps.log

After issuing the compile command, a successful compilation of WPS produces the three symbolic links: geogrid.exe, ungrib.exe, and metgrid.exe in the WPSV3/ directory, and several symbolic links in the util/ directory:

- avg_tsfc.exe
- calc_ecmwf_p.exe
- g1print.exe
- g2print.exe
- height_ukmo.exe
- int2nc.exe
- mod_levs.exe
- rd_intermediate.exe

4. Download and unpack under hwrfrun/sorc/WRFV3/DATA/ data for Hurricane Katrina:
http://www2.mmm.ucar.edu/wrf/TUTORIAL_DATA/Katrina.tar.gz
Download complete geographical input data:
http://www2.mmm.ucar.edu/wrf/users/download/get_sources_wps_geog.html and topo_2m:
http://www2.mmm.ucar.edu/wrf/src/wps_files/topo_2m.tar.bz2

Unpack this data under: hwrfrun/sorc/WPS_GEOG

5. Run
Set up WPS:

- Go to hwrfrun/sorc/WPSV3

- mkdir wpsprd
- Create new file called namelist.wps and paste following lines:

```
&share
 wrf_core = 'NMM',
 max_dom = 1,
 start_date = '2005-08-28_00:00:00','2005-08-28_00:00:00',
 end_date = '2005-08-30_00:00:00','2005-08-28_00:00:00',
 interval_seconds = 10800,
 io_form_geogrid = 2,
/

&geogrid
 parent_id     =  1,  1,
 parent_grid_ratio =  1,  3,
 i_parent_start  =  1, 17,
 j_parent_start  =  1, 31,
 e_we       = 48, 58,
 e_sn       = 92, 100,
 geog_data_res = 'default','default',
 dx = 0.193384,
 dy = 0.191231,
 map_proj = 'rotated_ll',
 ref_lat  = 42.00,
 ref_lon  = -71.00,
! truelat1 = 0,
! truelat2 = 0,
! stand_lon = -89.0,
 geog_data_path = '/home/admins/seba/HurricaneWRF/hwrfrun/sorc/WPS_GEOG'
 opt_geogrid_tbl_path = '/home/admins/seba/HurricaneWRF/hwrfrun/sorc/WPSV3/geogrid/'
/

&ungrib
 out_format = 'WPS',
 prefix = 'FILE',
/

&metgrid
 fg_name = 'FILE'
 io_form_metgrid = 2,
 opt_metgrid_tbl_path = '/home/admins/seba/HurricaneWRF/hwrfrun/sorc/WPSV3/metgrid/'
/
```

Look at the paths and modify them (important!)

- link binary files to current directory (hwrfrun/sorc/WPSV3/wpsprd/):
  ln -sf hwrfrun/sorc/WPSV3/*.exe .

Run geogrid.exe:
Before running the first step of WPS, make sure you are using the correct GEOGRID.TBL for the NMM:

ln -sf hwrfrun/sorc/WPSV3/geogrid/GEOGRID.TBL.NMM hwrfrun/sorc/WPSV3/geogrid/GEOGRID.TBL

Run geogrid.exe:

mpirun -np 4 ./geogrid.exe

After running geogrid.exe, a success message should appear on the screen and at the bottom of the geogrid.log file. The output file for the domain, called:

geo_nmm.d01.nc

Run ungrib.exe:
For the ungrib program to run, a Vtable must be supplied and the GRIB files must be linked to the file names that are expected by ungrib. The WPS is supplied with Vtable files for many sources of meteorological data. The Vtable corresponding to the input data you are using must be linked in using the following command:

ln -sf hwrfrun/sorc/WPSV3/ungrib/Variable_Tables/Vtable.GFS Vtable

The ungrib program will try to read GRIB files named GRIBFILE.AAA, GRIBFILE.AAB, ..., GRIBFILE.ZZZ . To simplify the work of linking the GRIB files to these filenames, a shell script, link_grib.csh, is provided. The link_grib.csh script takes as a command-line argument a list of the GRIB files to be linked. Issue the command:

hwrfrun/sorc/WPSV3/link_grib.csh hwrfrun/sorc/WRFV3/DATA/Katrina/*

After all the links have been completed, ungrib can be run by typing the following command:

./ungrib.exe >& ungrib.log

Since the ungrib program may produce a significant volume of standard output, it is recommended that the standard output be redirected to a log file, as shown in the command above. If ungrib completed successfully, a success message should appear at the bottom of ungrib.log and intermediate files should appear in the current working directory with file names:

FILE:yyyy-mm-dd_00
FILE:yyyy-mm-dd_03
FILE:yyyy-mm-dd_06
.
. where "FILE" is the prefix specified in the ungrib namelist record.

Run metgrid.exe:
The final step in the WPS process is to run metgrid, which interpolates the meteorological data extracted by ungrib to the simulation grid defined by geogrid.

Before running metgrid, make sure you are using the correct METGRID.TBL for the NMM:

ln -sf hwrfrun/sorc/WPSV3/metgrid/METGRID.TBL.NMM hwrfrun/sorc/WPSV3/metgrid/METGRID.TBL

Run metgrid by issuing the following command:

mpirun -np 4 ./metgrid.exe

If metgrid has run to completion, a success message "Successful completion of metgrid." will appear on the screen, as well as at the bottom of the metgrid.log file. The following output files should also have been created in the working directory:

met_nmm.d01.yyyy-mm-dd_00: 00:00
met_nmm.d01.yyyy-mm-dd_03: 00:00
met_nmm.d01.yyyy-mm-dd_06: 00:00

Set up WRF NMM.
Change to the main working directory for Katrina:

cd hwrfrun/sorc/WRFV3/DATA/Katrina/

Create a working directory for running HWRF:

mkdir wrfprd
cd wrfprd

Link the following to the working directory:

ln -sf hwrfrun/sorc/WRFV3/run/*_DATA .
ln -sf hwrfrun/sorc/WRFV3/run/ETAMP* .
ln -sf hwrfrun/sorc/WRFV3/run/*.TBL .
ln -sf hwrfrun/sorc/WRFV3/run/tr* .
ln -sf hwrfrun/sorc/WRFV3/run/*.txt .
ln -sf hwrfrun/sorc/WRFV3/run/*.tbl .
ln -sf hwrfrun/sorc/WRFV3/run/*.formatted .
ln -sf hwrfrun/sorc/WRFV3/main/*.exe .
ln -sf hwrfrun/sorc/WPSV3/wpsprd/met_nmm.d01* . (WPS output files)

Create file called namelist.input and paste following lines:

```
&time_control
run_days            = 2,
run_hours           = 0,
run_minutes         = 0,
run_seconds         = 0,
start_year          = 2005,   2005,
```

```
start_month           = 08,   08,
start_day             = 28,   28,
start_hour            = 00,   00,
start_minute          = 00,   00,
start_second          = 00,   00,
tstart                = 00,
end_year              = 2005, 2005,
end_month             = 08,   08,
end_day               = 30,   30,
end_hour              = 00,   00,
end_minute            = 00,   00,
end_second            = 00,   00,
interval_seconds      = 10800,
history_interval      = 60,   60,
frames_per_outfile    = 1,    1,
restart               = .false.,
restart_interval      = 5400,
reset_simulation_start = F,
io_form_input         = 2
io_form_history       = 2
io_form_restart       = 2
io_form_boundary      = 2
io_form_auxinput1     = 2
debug_level           = 300
/

&domains
time_step             = 60,
time_step_fract_num   = 0,
time_step_fract_den   = 1,
max_dom               = 1,
e_we                  = 48,   58,
e_sn                  = 92,   100,
e_vert                = 38,   38,
num_metgrid_levels    = 27,
dx                    = 0.193384, 0.064461,
dy                    = 0.191231, 0.063744,
p_top_requested       = 5000.
ptsgm                 = 42000.,
grid_id               = 1,    2,
parent_id             = 0,    1,
i_parent_start        = 1,    17,
j_parent_start        = 1,    31,
parent_grid_ratio     = 1,    3,
parent_time_step_ratio = 1,   3,
/

! &physics
```

```
! mp_physics            = 5,    5,
! ra_lw_physics          = 1,    1,
! ra_sw_physics          = 1,    1,
! nrads                  = 50,   150,
! nradl                  = 50,   150,
! sf_sfclay_physics      = 2,    2,
! sf_surface_physics     = 2,    2,
! bl_pbl_physics         = 2,    2,
! nphs                   = 2,    6,
! cu_physics             = 2,    2,
! ncnvc                  = 2,    6,
! num_soil_layers        = 4,
! /

&physics
 mp_physics             = 3,   3,
 ra_lw_physics          = 1,   1,
 ra_sw_physics          = 1,   1,
 radt                   = 30,  30,
 sf_sfclay_physics      = 1,   1,
 sf_surface_physics     = 2,   2,
 bl_pbl_physics         = 1,   1,
 bldt                   = 0,   0,
 cu_physics             = 1,   1,
 cudt                   = 5,   5,
 isfflx                 = 1,
 ifsnow                 = 0,
 icloud                 = 1,
 surface_input_source   = 1,
 num_soil_layers        = 4,
 sf_urban_physics       = 0,   0,
 /

! &dynamics
! coac                   = 1.6,
! codamp                 = 6.4,
! slophc                 = 0.0064,
! euler_adv              = .true.,
! idtadt                 = 2,
! idtadc                 = 1
! /

 &bdy_control
 spec_bdy_width         = 1,
 specified              = .true.,
 nested                 = .false.
 /
```

```
&fdda
/

&grib2
/

&namelist_quilt
nio_tasks_per_group = 0,
nio_groups = 1
/
```

Run real_nmm.exe:
To run real_nmm.exe, type:

mpirun -np 4 ./real_nmm.exe

This command uses 4 CPUs to run real_nmm.exe.
The standard output and error for an MPI run are written to the following files:

```
rsl.out.0000  rsl.error.0000
rsl.out.0001  rsl.error.0001
rsl.out.0002  rsl.error.0002
rsl.out.0003  rsl.error.0003
```

One pair of output files is generated for each running processor.
To determine whether the run is successful, type:

tail rsl.out.0000

and look for: "SUCCESS COMPLETE REAL_NMM INIT "

Make a directory to move the rsl.* files in order to save the output and create new files when running the wrf.exe.

mkdir rsl_real
mv rsl.* rsl_real

If the "real_nmm.exe" ran successfully, the following files should be found in the working-directory:
wrfinput_d01 (Initial conditions, single time level data)
wrfbdy_d01 (Boundary condition data for multiple time steps)

Run WRF:
To run wrf.exe, type:
mpirun -np 6 ./wrf.exe

To determine whether the run is successful, type:

tail rsl.out.0000

and look for: "SUCCESS COMPLETE WRF"
If "wrf.exe" is successful, the following files should be found in the working directory:

wrfout_d01_yyyy-mm-dd_00:00:00
wrfout_d01_yyyy-mm-dd_01:00:00
wrfout_d01_yyyy-mm-dd_02:00:00
.
.
The times written to an output file can be checked by typing:

ncdump -v Times wrfout_d01_yyyy-mm-dd_00:00:00

5. Future Advanced HPC Architectures

5.1 Reference architecture Xeon E5-2600v3 (Haswell)

Architecture description

Cores	18
L1 cache	64 KB per core
L2 cache	256 KB per core
L3 cache	2-40 MB (shared)
Created	2013
Architecture	Haswell x86
Extensions	x86-64, Intel 64 SSE, SSE2, SSE3, SSSE3, SSE4, SSE4.1, SSE4.2 AVX, AVX2, TXT, and TSX (disabled via microcode, except for Haswell-EX) VT-x, VT-d
Socket(s)	LGA 2011-v3
Energy	Power consumption 52W (Xeon 2608Lv3) - 145W(Xeon 2699v3)

For the first time, **a single CPU is capable of more than half a TeraFLOPS** (500 GFLOPS). This is made possible through the use of AVX2 with FMA3 instructions.
Important changes available in E5-2600v3 “Haswell-EP” include:

- Up to 18 processor cores per socket (with options for 4-, 6-, 8-, 10-, 12-, 14- and 16-cores)
- Support for Quad-channel ECC DDR4 memory speeds up to 2133MHz
- Direct PCI-Express (generation 3.0) connections between each CPU and peripheral devices such as network adapters, GPUs and coprocessors (40 PCI-E lanes per socket)
- Advanced Vector Extensions (AVX 2.0):
  - effectively double the throughput of integer and floating-point operations with math units expanded from 128-bits to 256-bits
  - introduce Fused Multiply Add (FMA3) instructions which allow a multiply and an accumulate instruction to be completed in a single cycle (effectively doubling the FLOPS/clock from 8 to 16 for each core of a CPU)
  - add support for additional instructions, including Gather and vector shift
  - F16C 16-bit Floating-Point conversion instructions accelerate data conversion between 16-bit and 32-bit floating point formats
- Turbo Boost technology improves performance under peak loads by increasing processor clock speeds. With version 2.0, (introduced in “Sandy Bridge”) clock speeds are boosted more frequently, to higher speeds and for longer periods of time. With “Haswell”, top clock speeds depend upon the type of instructions (AVX vs. Non-AVX).
- Dual Quick Path Interconnect (QPI) links between processor sockets improve communication speeds for multi-threaded applications
- Improved energy efficiency with Per Core P-States and independent uncore frequency control

- Intel Data Direct I/O Technology increases performance and reduces latency by allowing Intel ethernet controllers and adapters to talk directly with the processor cache
- Advanced Encryption Standard New Instructions (AES-NI) accelerate encryption and decryption for fast, affordable data protection and security
- 32-bit & 64-bit Intel Virtualization Technology (VT/VT-x) for Directed I/O (VT-d) and Connectivity (VT-c) deliver faster performance for core virtualization processes and provide built-in hardware support for I/O virtualization.
- Intel APIC Virtualization (APICv) provides increased virtualization performance
- Hyper-Threading technology allows two threads to "share" a processor core for improved resource usage. Although useful for some workloads, it is not recommended for HPC applications.

Cluster (node) configuration

Number of nodes	1178
Cpu per board	2x Intel Xeon E5-2697v3
RAM	64GB - 589 nodes 128GB – 530 nodes 256GB – 59 nodes
Interconnect description	InfiniBand FDR (56Gb/s)
I/O and disks	Lustre
OS version	Scientific Linux CERN 6.7 (Carbon)
Programming environment, compilers, libs etc	C++ Compiler 17.0 Update 1 Fortran Compiler 17.0 Update 1 Math Kernel Library 2017 Update 1 for C/C++ Math Kernel Library 2017 Update 1 for Fortran MPI Library 2017 Update 1 Integrated Performance Primitives 2017 Update 1 Threading Building Blocks 2017 Update 2 Data Analytics Acceleration Library 2017 Update 1 Debugger for Heterogeneous Compute 2017 Update Debugger for Intel(R) MIC Architecture 2017 Update 1 GNU* GDB 7.10 VTune(TM) Amplifier XE 2017 Update 1 Inspector 2017 Update 1 Advisor 2017 Update 1 Trace Analyzer and Collector 2017 Update 1 Distribution for Python* 2.7 Update 1 Distribution for Python* 3.5 Update 1 Scons 2.4.1 OpenBLAS 0.2.19 ScaLAPACK 2.0.2 GDAL 2.1.3 GDAL 1.11.5

5.2 Xeon E5-2600v4 (Broadwell)

Architecture description

Cores	32
L1 cache	64 KB per core
L2 cache	256 KB per core
L3 cache	2-6 MB (shared)
Created	2016
Architecture	x86
Extensions	x86-64, Intel 64 SSE, SSE2, SSE3, SSSE3, SSE4, SSE4.1, SSE4.2 AVX, AVX2, TXT, and TSX VT-x, VT-d MMX
Socket(s)	LGA 2011-v3
Energy	Power consumption 120W

Comparing to Haswell, E5-2600v4 "Broadwell-EP" provides the following:

- Up to 22 processor cores per socket (with options for 4-, 6-, 8-, 10-, 12-, 14- and 16-cores)
- Support for Quad-channel ECC DDR4 memory speeds up to 2400MHz
- Faster Floating Point Instruction performance
- Improved parallelism in scheduling micro-operations
- Improved performance for large data sets
- 14nm technology.

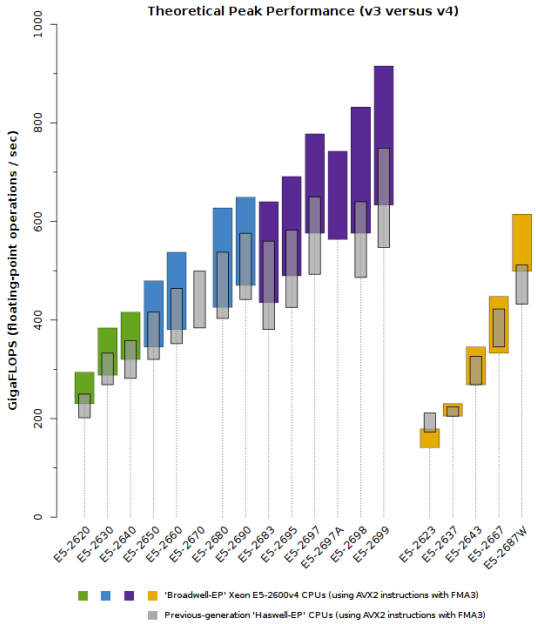

Fig. 4 Theoretical of peak performance Xeon E5 v3 vs. v4

Cluster (node) configuration

Number of nodes	50
CPUs per board	2x Intel Xeon E5-2682v4
RAM	256 GB
Interconnect description	InfiniBand FDR (56Gb/s)
I/O and disks	Lustre
OS version	Scientific Linux CERN 6.7 (Carbon)
Programming environment, compilers, libs etc	C++ Compiler 17.0 Update 1 Fortran Compiler 17.0 Update 1 Math Kernel Library 2017 Update 1 for C/C++ Math Kernel Library 2017 Update 1 for Fortran MPI Library 2017 Update 1 Integrated Performance Primitives 2017 Update 1 Threading Building Blocks 2017 Update 2 Data Analytics Acceleration Library 2017 Update 1 Debugger for Heterogeneous Compute 2017 Update Debugger for Intel(R) MIC Architecture 2017 Update 1 GNU* GDB 7.10 VTune(TM) Amplifier XE 2017 Update 1 Inspector 2017 Update 1 Advisor 2017 Update 1 Trace Analyzer and Collector 2017 Update 1 Distribution for Python* 2.7 Update 1 Distribution for Python* 3.5 Update 1 Scons 2.4.1 OpenBLAS 0.2.19 ScaLAPACK 2.0.2 GDAL 2.1.3 GDAL 1.11.5 Boost 1.63 HDF5 1.8.18 Python 2.7.12

5.3 Xeon Phi™ 7250

Architecture description

Cores	68
L1 cache	32 kB
L2 cache	34 MB
L3 cache	
Created	2016
Architecture	Intel Xeon Phi (Knights Landing)

Extensions	Multithreading
Socket(s)	LGA-3647
Energy	Power consumption 215 W

The Xeon Phi architecture, which has been developed and manufactured by Intel, is a many-core architecture that has been introduced for the high-end server market. Initially, in contrast to the well-adopted Intel Xeon processors, Xeon Phi has been developed as a co-processor or accelerator, which makes use of the available Peripheral Component Interconnect Express (PCIe) bus. However, the newest Xeon Phi, Knights Landing, additionally operates in standalone mode, so that the performance bottlenecks of the PCIe bus can be overcome. In order to keep the description and comparisons reasonable, only the newest model of the Xeon Phi development Knights Landing is focused within this subsection.

The Intel Knights Landing processor is manufactured in a 14 nm process and as the first Intel processor, supports the innovative on-package memory technology that increases significantly the memory bandwidth to the in maximum 72 processor cores. Those processor cores are based on the Intel Atom architecture, codename Airmont, offer four threads per core and run with a clock rate of maximally 1.500 MHz for the top model. Besides the on-package Multi-Channel Dynamic Random Access Memory (MCDRAM), another 384 GB of standard DDR4 Dynamic Random Access Memory (DRAM) are supported. Fig. 5 complements the information above by presenting the schematic architecture and the interconnections between the processor cores, the memory as well as Intel's brand new networking architecture Omni-Path.

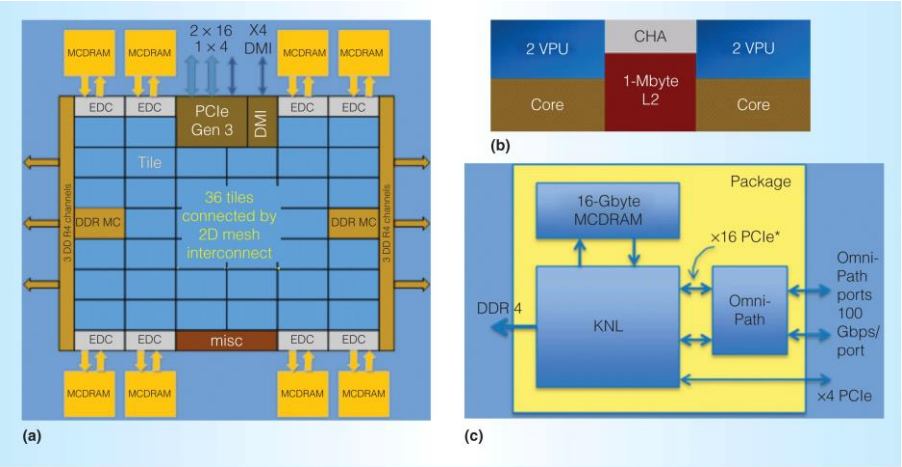

Fig. 5 Intel KNL schematic architecture (20)

For a better understanding of the Xeon Phi development, The Tab. 1compares a couple of key performance indicators for different systems. It becomes obvious that the Xeon Phi bridges the gap between traditional Intel Xeons and accelerated computing with GPUs. Nevertheless, all compared processor types do have their right to exist: although accelerated computing can provide a lot of performance for massively parallel applications, the PCIe bus with its 12.5 GB/s of performance limits the direct host memory access drastically. In addition, low clock rates are

counterproductive for sequential parts in the application kernels, so that traditional Intel Xeons can exploit their power for this key performance indicator.

Tab. 1 Comparision of Xeon Phi 7290, E5 2699v4 and Tesla P100

Processors / Key Performance Indicators	Intel Knights Landing Xeon Phi 7290	Intel Xeon Broadwell Xeon E5-2699v4	NVIDIA GPU Tesla P100
Processor cores	72 (288 threads)	22 (44 threads)	3.584
Base clock	1.500 MHz	2.200 MHz	1.320 MHz
Turbo clock	1.700 MHz	3.600 MHz	1.480 MHz
Memory size	16 GB / 384 GB	1.54 TB	16 GB
Memory bandwidth	400 GB/s / 115.2 GB/s	76.8 GB/s	730 GB/s
GFLOPs	3.500 GFLOPs	630 GFLOPs	4.700 GFLOPs

Cluster (node) configuration

Number of nodes	2
CPUs per board	2x Intel Xeon Phi 7250
RAM	256 GB
Interconnect description	OmniPath
I/O and disks	
OS version	Ubuntu 16.10
Programming environment, compilers, libs etc	GCC 4.8.5 GCC 5.4.0 Scons 2.4.1 OpenBLAS 0.2.19 ScaLAPACK 2.0.2 GDAL 2.1.3 GDAL 1.11.5 Boost 1.63 HDF5 1.8.18 Python 2.7.12

5.4 ARM

Architecture description

Cores	32
L1 cache	80 kB
L2 cache	2 MB
L3 cache	n/a
Created	2015
Architecture	Cortex A-57 (ARMv8-A 64-bit)
Extensions	fp asimd aes pmull sha1 sha2 crc32 DSP and NEON SIMD extensions are mandatory per core VFPv4 Floating Point Unit onboard (per core) Hardware virtualization support Thumb-2 instruction set encoding reduces the size of 32-bit programs with little impact on performance. TrustZone security extensions
Socket(s)	N/A
Energy	Power consumption 55 W

The development of ARM-based server processors is still a very hot topic within the HPC community; however, the speed of development has slowed down a little. Therefore, most of the statements of D5.5 are still valid so that this subsection discusses the similarities and changes for the different vendors. Besides the actual chip manufacturers, which buy an ARM chip design and evolve it, it needs to be stated that ARM has been acquired recently for $31 billion by the Japanese company Softbank (21), (22). Thus, ARM is no longer a European company and the developments within this Japanese group need to be monitored carefully.

The brief assessment of the relevant vendors is presented below:

- **Broadcom**
  In October 2013, Broadcom announced to develop and produce a server-class ARMv8 many-core processor for the enterprise and HPC market. After several years of development, Broadcom abruptly announced to stop the developments for ARM-based server chips and focus on their key business: networks again. Consequently, Broadcom developments will not be part of any technology evaluation anymore.
- **AMD**
  AMD offers an ARM-based processor and published a roadmap for future developments (23). According the AMD website (), the main purpose of this processor is to support classical data centre workloads. With respect to this statement and its specifications, the chip will not be of interest for future High Performance Computing centres focusing
- **AppliedMicro**
  In January 2017, it has been announced that MACOM () successfully completed the acquisition of AppliedMicro (24). Beforehand it was communicated that especially the networking branches of AppliedMicro were of interest for MACOM and, furthermore, that

the ARM server development sector would be sold (25). When writing this deliverable, no further information is available so that the AppliedMicro developments will be monitored until a clear statement is available.

- **Cavium**
  The ThunderX processor manufactured by Cavium is currently the only serious ARM-based enterprise server processor in the market. Although ThunderX2 has been announced recently (26), the specifications of the ThunderX processor in D5.5 still hold. A specification of the ThunderX2 chip will be provided after an official release is available (expected during 2017).
- **Qualcomm**
  Qualcomm announced a 48 cores ARM-based server-class processor for the second half of 2017 (27). Although this development is not relevant for this deliverable, Cavium will get a competitor in the ARM HPC market. CoeGSS considers this development very interesting and efforts will be undertaken to provide access to such a system.

Considering all the information above it becomes obvious that the ThunderX processor from Cavium is the only candidate to perform tests and co-design activities on.

Cluster (node) configuration

Number of nodes	2
CPUs per board	2
RAM	
Interconnect description	InfiniBand FDR (56Gb/s)
I/O and disks	SSD
OS version	Ubuntu 16.04.1 LTS
Programming environment, compilers, libs etc	OpenMPI 1.10.2 Scons 2.4.1 OpenBLAS 0.2.19 ScaLAPACK 2.0.2 GDAL 2.1.3 GDAL 1.11.5 Boost 1.63 HDF5 1.8.18 Python 2.7.12

5.5 Power8

Architecture description

Cores	10
L1	Instr:32KB per core; Data: 64KB per core
L2 cache	512KB per core
L3 cache	8MB per core
L4 cache	16MB per DIMM
Created	2015
Architecture	Power Architecture (Power ISA v.2.07)
Extensions	fp asimd aes pmull sha1 sha2 crc32
Socket(s)	
Energy	Power consumption

Within this subsection of the deliverable, the Power architecture is described and the recent developments are updated. First of all, Power relates to a couple of products, reaching from the POWER high performance microprocessors, up to the PowerPC and the Cell processors. This section focuses on the developments of the High Performance Computing chips, the POWER8 processors.

IBM commits to the development roadmap as shown in Fig. 6Besides the roadmap, this figure shows in addition that for the beginning of 2017, IBM POWER8+ is the targeted processor technology, which will finally be replaced by POWER9 (expected at the end of 2017). POWER8+ supports the NVLINK technology, which provides higher bandwidth between host and GPU as the x86 PCIe bus. However, this technology does not replace PCIe, it complements it. NVLINK is only used to transfer data, actual instructions are still transferred via PCIe. Hence, if the functionalities of modern NVIDIA GPUs cannot be used, there is no reason for changing from a POWER8 to a POWER8+ system.

As already highlighted, in 2017 a new POWER9 chip will be released, which will serve as the basis for the huge installations planned at Oak Ridge and Livermore through the CORAL program (28). Both systems are expected to be amongst the fastest systems of the world when entering the Top500 list () so that for sure the innovative architecture is of interest for CoeGSS as well. Nevertheless, detailed specifications are market-ready products are mandatory before taking any decision.

Taking the above information into account, it can be stated that all statements of D5.5 are still correct and no updates are required. Since the applications of CoeGSS are rather data than compute bound, all co-design activities can be performed on an already available POWER8 system at the High Performance Computing Center Stuttgart.

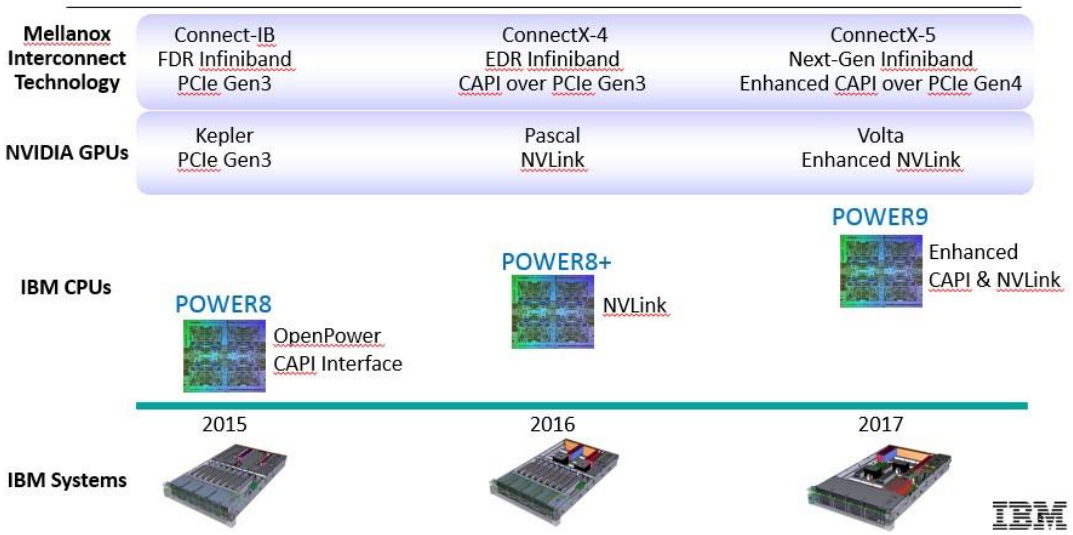

Fig. 6 IBM Power roadmap (29)

Cluster (node) configuration

At the High Performance Computing Center Stuttgart, a single node system equipped with the POWER8 processor is available for supporting the co-design activities. The system is not meant for production, it serves as a testing facility to understand applications in more detail and benchmark the POWER8 hardware as well as the installed software stack.

Number of nodes	1 (Model 8247-22L)
CPUs per board	2
RAM	256 GB
Interconnect description	
I/O and disks	32GBps, 4 PCIe Gen3 slots (one x16 and three x8 PCIe Gen3) 128 GB HDD
OS version	Ubuntu 16.04.2 LTS
Programming environment, compilers, libs etc	GCC 5.4.0 (20160609) OpenMPI 1.10.2 (2016/01/21) GDAL 1.11.3 (2015/09/16) HDF5 (2016/05/10) Jupyter 4.2.0 Osmosis 0.44 Python 2.7.12

6. Tests performed

For applications benchmarking purposes it was decided to use */usr/bin/time* Linux command providing several useful statistics from which the following were used:

- **%e** Elapsed real time (in seconds; not in tcsh),
- **%M** Maximum resident set size of the process during its lifetime, in Kbytes,
- **%I** Number of file system inputs by the process,
- **%O** Number of file system outputs by the process.

It is worth noting that the last four metrics are reported only by the process that is under control of */usr/bin/time*.

A number of MPI processes have been adjusted to the range of cores accessible on tested architectures. Most of the tests have been repeated when random values of %I (mostly equal to 0) were spotted. In many cases %I results equals 0, which is in fact caused by reading input data from system caches instead of the storage system. This behaviour saves time on reading from disk, but in benchmark data should always be read from cache or always read from disk. Random 0 values mean that both cases occur. The solution was based on executing the following line as a root user before running benchmark:

```
sync; echo 3 > /proc/sys/vm/drop_caches
```

6.1 Green Growth using Pandora library

The tests have been conducted on two provided datasets, as described in Section 4.1. One dataset is restricted to Europe, the second cover the whole world.

The following tables and charts present achieved results.

ARM without cache clearing

EU map

Tab. 2 Pandora tests results on 2-nodes ARM system for EU map

	ARM/ EU map			
#MPI procs	%e	%M	%I	%O
1	313,20	829080	0	3699632
16	101,72	117792	16	3864560
32	80,57	77100	75840	3842656
64	74,44	59200	0	1988520

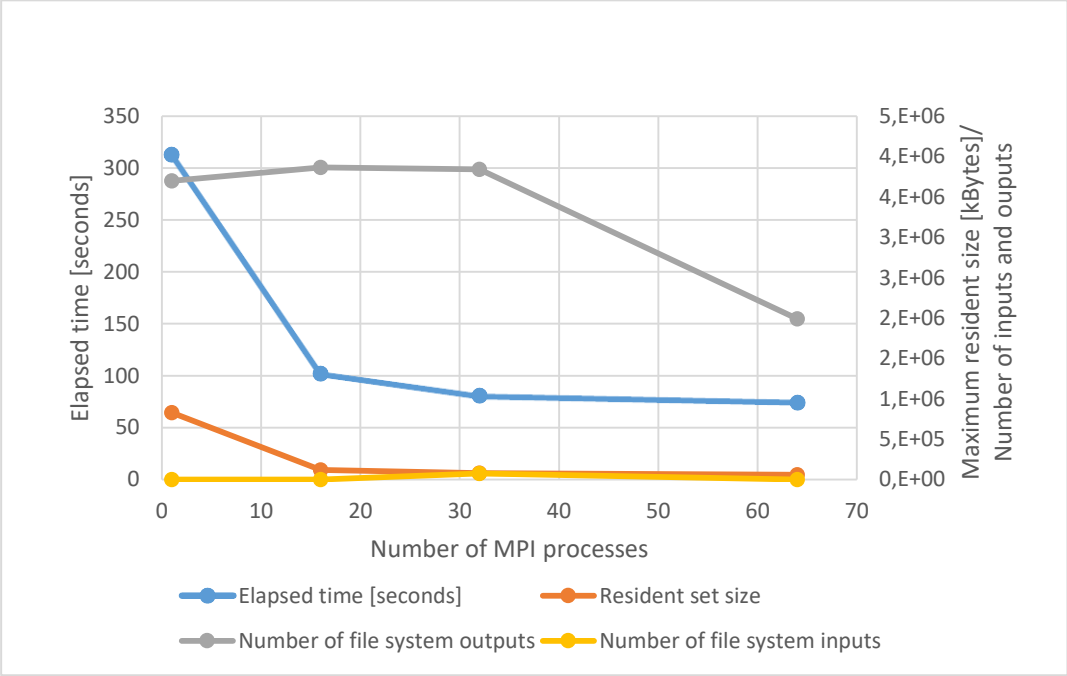

Fig. 7 GG-pilot w/Pandora scalability on 2-nodes ARM system for EU map

World Wide (WW) map

Tab. 3 Pandora tests results on 2-nodes ARM system for WW map

	ARM/ WW map			
#MPI procs	%e	%M	%I	%O
1	18502,65	24421180	1074864	234101376
16	5122,42	3787452	4744	240661680
32	3070,43	1947784	416080	240739920
64	1811,35	1102864	2014328	132071512

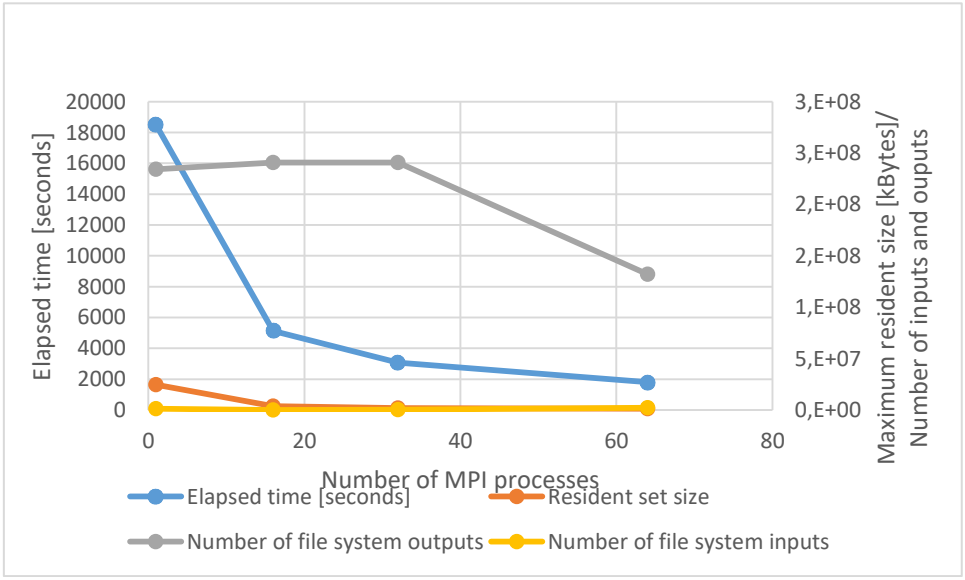

Fig. 8 GG-pilot w/Pandora scalability on 2-nodes ARM system for WW map

ARM with cache clearing

EU map

Tab. 4 GG-pilot w/Pandora scalability on 2-nodes ARM system for EU map and cache clearing

	ARM/ EU map with cache clearing			
#MPI procs	%e	%M	%I	%O
1	293,48	829768	240424	3699528
16	104,12	117300	241008	3804472
32	82,66	77060	240872	3819216
64	85,81	57036	157808	1994856

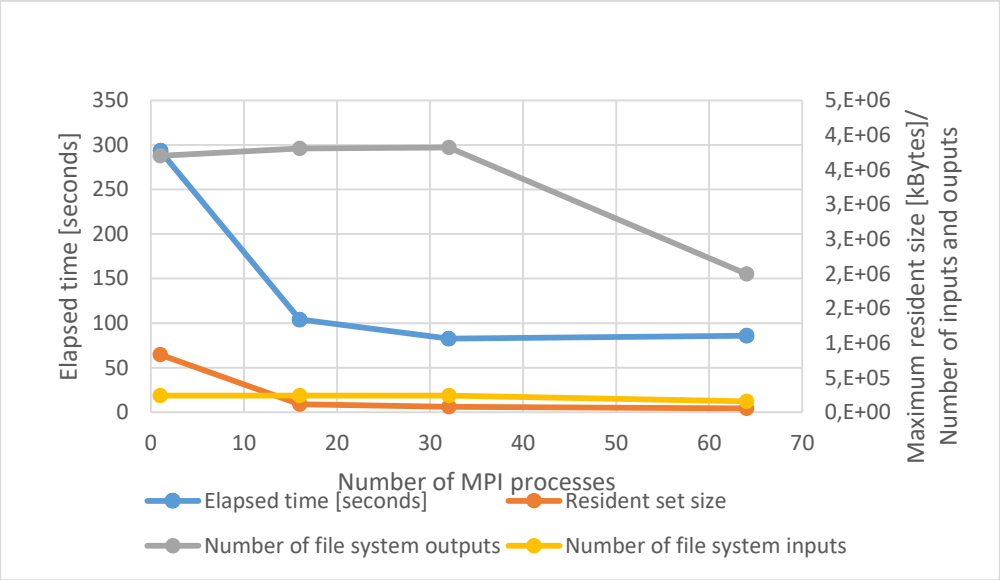

Fig. 9 GG-pilot w/Pandora scalability on 2-nodes ARM system for EU map and cache clearing

Clearing the cache before test execution results (for more than 1 MPI process) in longer time execution in this case. This is natural because input data have to be read in again from disk, not from memory. The difference in execution times between these two cases varies between 3 and 13% . The best result for the EU map in the context of execution time is for 64 MPI processes with input data already existing in cache memory. For WW map the winner is 64 MPI processes too, although there is no direct comparison to results with cache clearing.

Xeon Phi™ 7250 without cache clearing

EU map

Tab. 5 GG-pilot w/Pandora scalability on Xeon Phi (KNL) 7250 without cache clearing

	Xeon Phi™ 7250 / EU map			
#MPI procs	%e	%M	%I	%O
1	1315,42	823256	0	7662072
16	251,68	124700	992	7687088
32	165,09	86424	0	7597296
64	127,96	71844	0	7602376
128	124,12	102652	0	7312176
256	173,42	173144	0	7339264
272	178,80	291360	178632	7717184
512	280,83	178312	1671024	3668312
544	288,12	185188	1696032	3639296

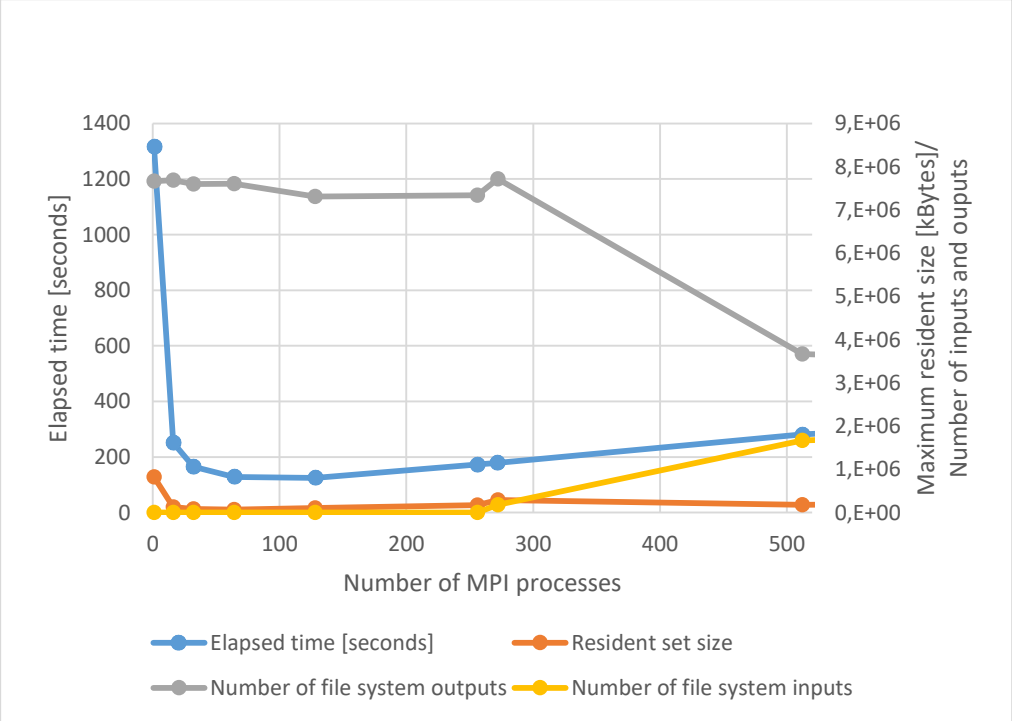

Fig. 10 Pandora scalability on Xeon Phi (KNL) 7250 without cache clearing for EU map

World Wide (WW) map

Tab. 6 Pandora scalability on Xeon Phi (KNL) 7250 without cache clearing for WW map

	Xeon Phi™ 7250/ WW map			
#MPI procs	%e	%M	%I	%O
1	35035,43	24040800	1078992	234104640
16	9141,26	3565742	4984	234188656
32	5644,77	1927244	2304	234219104
64	4065,12	1103516	19976	234273920
128	4441,21	649240	3047992	234377832
256	4711,22	345596	2808424	234147584
272	8090,34	373016	493131656	234135440
512	2365,91	621156	2896	123577592
544	4640,04	386084	247615048	126988480

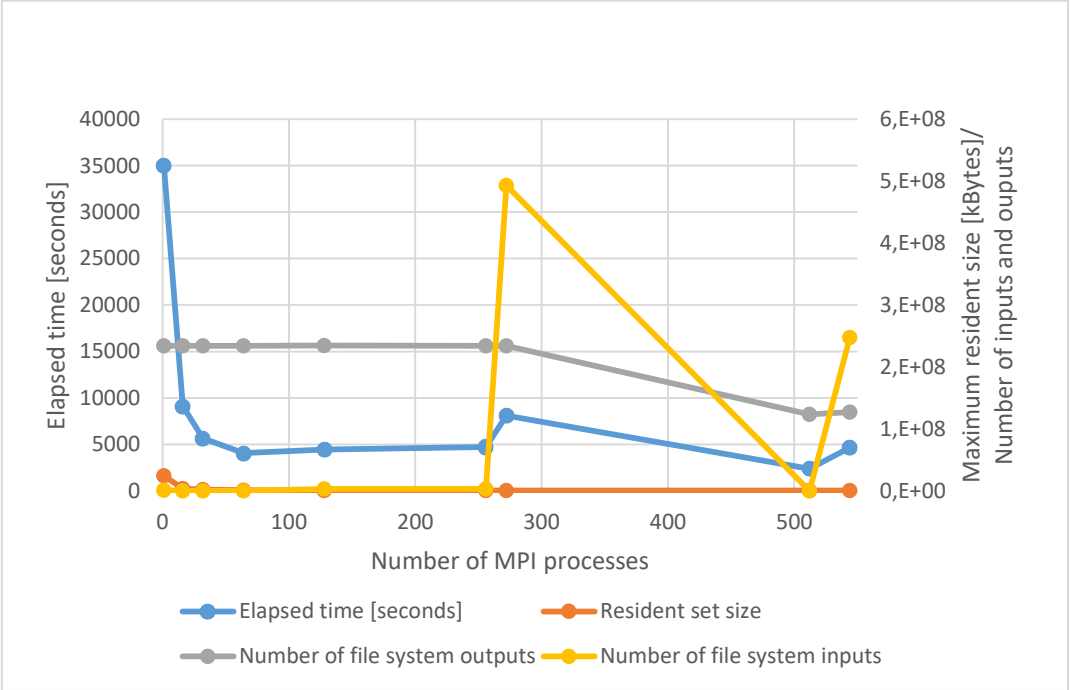

Fig. 11 Pandora scalability on Xeon Phi (KNL) 7250 without cache clearing for WW map

Xeon Phi™ 7250 with cache clearing

EU map

Tab. 7 Pandora scalability on Xeon Phi (KNL) 7250 with cache clearing for EU map

	Xeon Phi™ 7250/ EU map with cache clearing			
#MPI procs	%e	%M	%I	%O
1	1586,05	822532	238384	7661736
16	309,58	125680	240448	7679248
32	192,03	85980	240304	7594992
64	170,93	73448	3616056	7633160
128	196,85	105124	3615384	7649848
272	179,77	292168	240112	7727720
544	294,38	185436	1861992	3702016

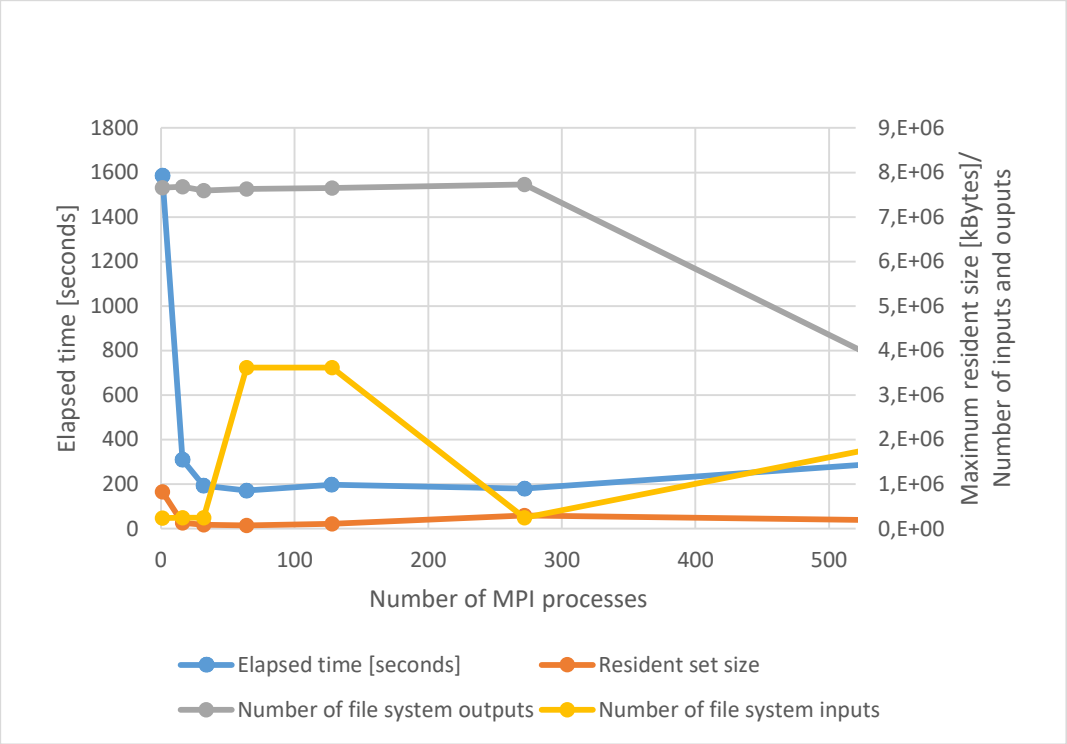

Fig. 12 Pandora scalability on Xeon Phi (KNL) 7250 with cache clearing for EU map

In general, for Xeon Phi™ 7250 execution time is longer comparing to the case when input data are read from cache. For 64, 128 and 544 numbers of input readings are significantly greater. Most probably in these cases map pieces distribution between processes imposes the situation in which fractions of these pieces have to be read redundantly to cover the whole map.

Execution time is the best for 128 processes when data already reside in memory. The worst case is for a single process job, which is completely natural because this processor family normally shows its power when more cores are harnessed.

For WW map for 512 and more processes execution terminates with exit code equals 13 which stands for Permission denied. This is most probably related to pipe failure - one process is trying to write to a process but there is no process to receive the data.

```
===================================================================================
=  BAD TERMINATION OF ONE OF YOUR APPLICATION PROCESSES
=  PID 60461 RUNNING AT ……
=  EXIT CODE: 13
=  CLEANING UP REMAINING PROCESSES
=  YOU CAN IGNORE THE BELOW CLEANUP MESSAGES
===================================================================================
 Intel(R) MPI Library troubleshooting guide:
  https://software.intel.com/node/561764
===================================================================================
```

Command exited with non-zero status 13

Xeon E5-2697 v3 without cache clearing

EU map

Tab. 8 Pandora scalability on Xeon E5-2697 v3 without cache clearing for EU map

	Eagle (E5-2697 v3)/ EU map			
#MPI procs	%e	%M	%I	%O
1	223,89	1012572	369360	7660584
28	48,51	96592	8	7668744
56	65,73	82952	287328	1425504
112	78,43	100504	104688	755232
224	88,90	140224	8	344272
448	85,87	221228	8	174544
700	215,84	320908	8	81560
1400	247,28	581960	8	89608
2800	329,90	1105284	3504	58632

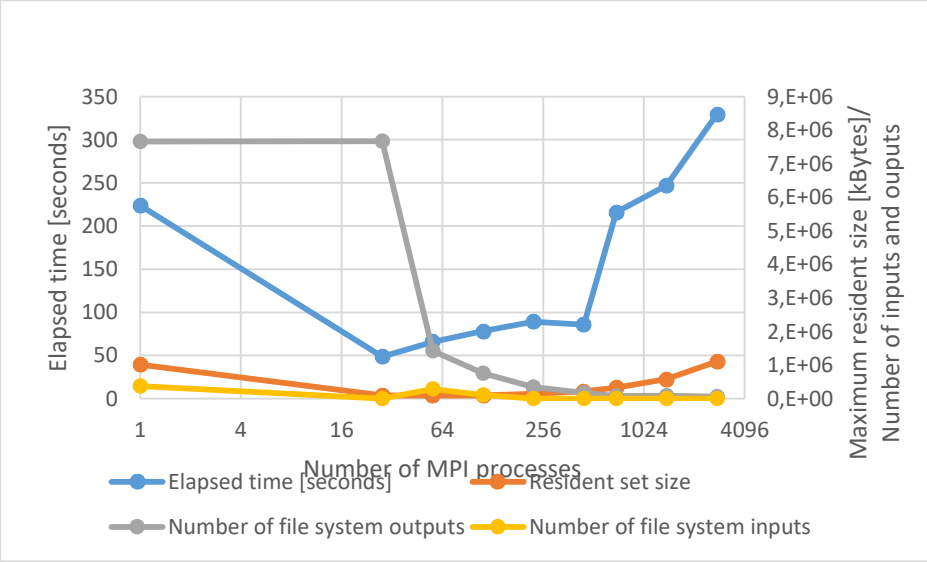

Fig. 13 GG-pilot w/Pandora scalability on 100-node Eagle (E5-2697 v3) cluster (EU map)

WW map

Tab. 9 Pandora scalability results on 100-node Eagle (E5-2697 v3) cluster (WW map)

	Eagle (E5-2697 v3)/ WW map			
#MPI procs	%e	%M	%I	%O
1	7134,70	29686808	19800	234114504
28	1828,70	2646524	4207568	234158960
56	1127,16	1700808	1771528	58498680
112	783,58	742276	8	24421744
224	699,80	458728	991240	12017368
448	744,28	262420	65776	5998232
700	542,85	333356	0	3844944
1400	853,44	592884	8	3833624
2800	996,88	1115616	294488	1914328

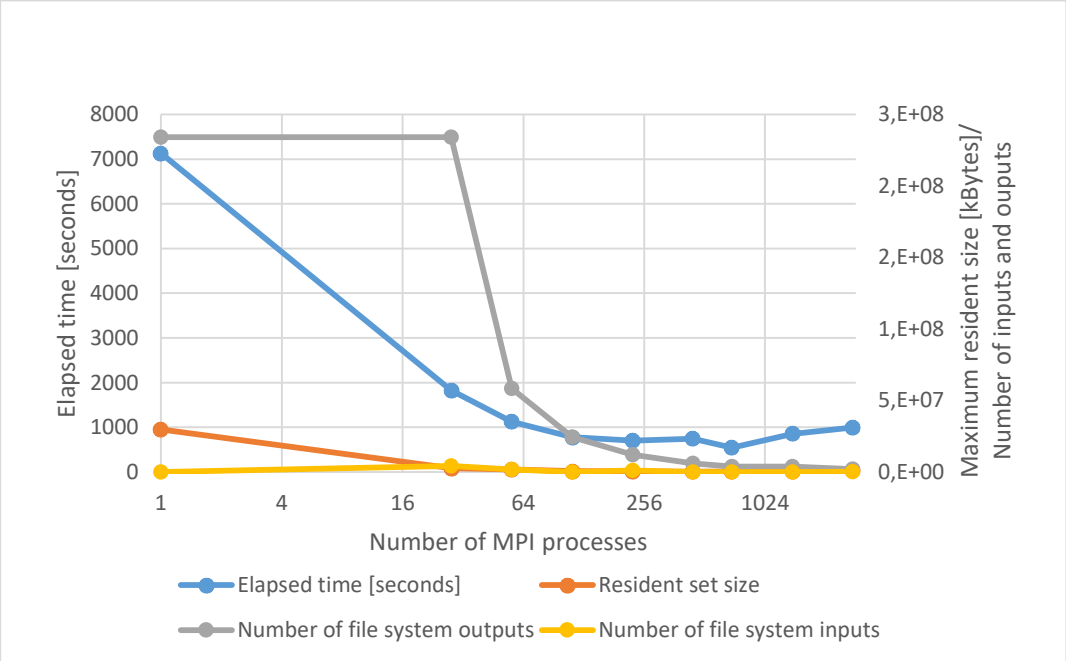

Fig. 14 GG-pilot w/Pandora scalability on 100-node Eagle (E5-2697 v3) cluster (WW map)

For this architecture testbed the data have only been collected in case when intentional cache clearing was switched off. It can be observed for the EU map the best execution time is for only 28 MPI processes. After this point the more processes are incorporated, the worse results are achieved. Above that point the application stops scaling. For the WW map the observations are completely different. The sweet spot is around 700 MPI processes.

Xeon E5-2682 v4 without cache clearing

EU map

Tab. 10 Pandora results on 50-node Eagle (E5-2682 v4) cluster (EU map) without cache clearing

	Eagle (E5-2682 v4)/ EU map			
#MPI procs	%e	%M	%I	%O
1	224,70	818652	416680	7660568
16	77,38	141100	431448	7668584
32	54,88	94208	431856	7584096
64	79,95	86752	431856	1489000
128	134,06	108080	8	543016
256	101,15	160132	0	338384
512	181,02	258568	0	131272
800	133,06	354984	0	108024
1600	156,63	646736	8	109424

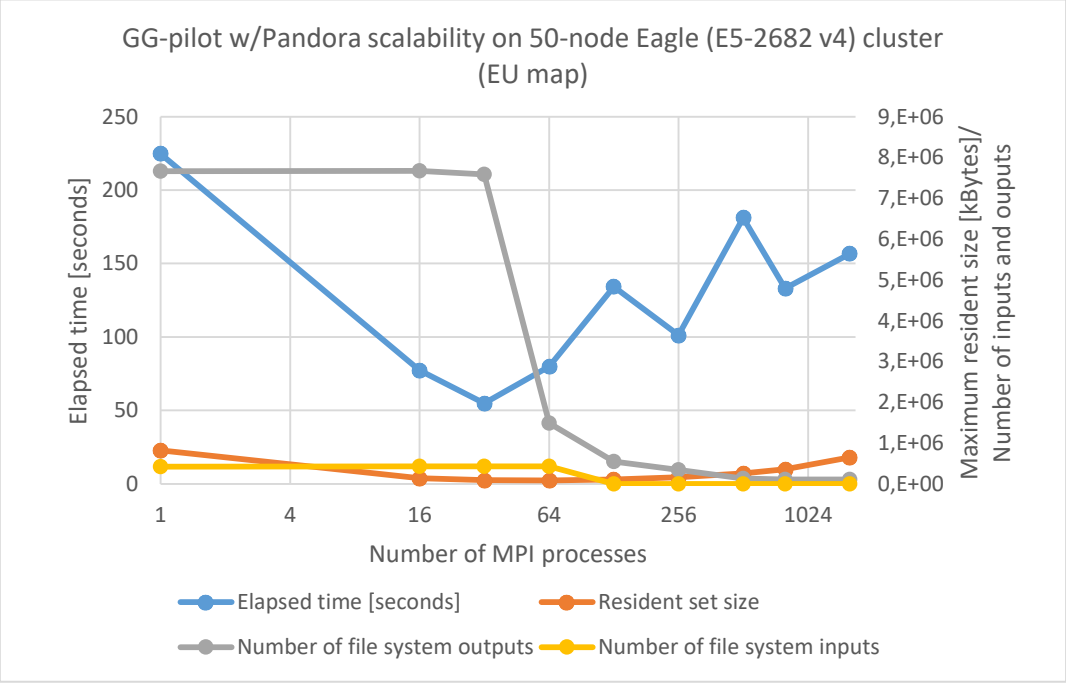

Fig. 15 Pandora results on 50-node Eagle (E5-2682 v4) cluster (EU map) without cache clearing

WW map

Tab. 11 Pandora results on 50-node Eagle (E5-2682 v4) cluster (WW map) without cache clearing

	Eagle (E5-2682 v4)/ WW map			
#MPI procs	%e	%M	%I	%O
1	7167,31	24211408	4097624	234114840
16	2579,59	4187388	4246400	234148440
32	1685,83	2390652	4097624	234158856

64	1159,09	1224076	2726640	56369616
128	793,32	594164	2763504	24606496
256	712,03	376600	441072	11976792
512	620,93	279988	441216	5998224
800	888,68	371772	294488	3839392
1600	559,22	655820	0	3810800

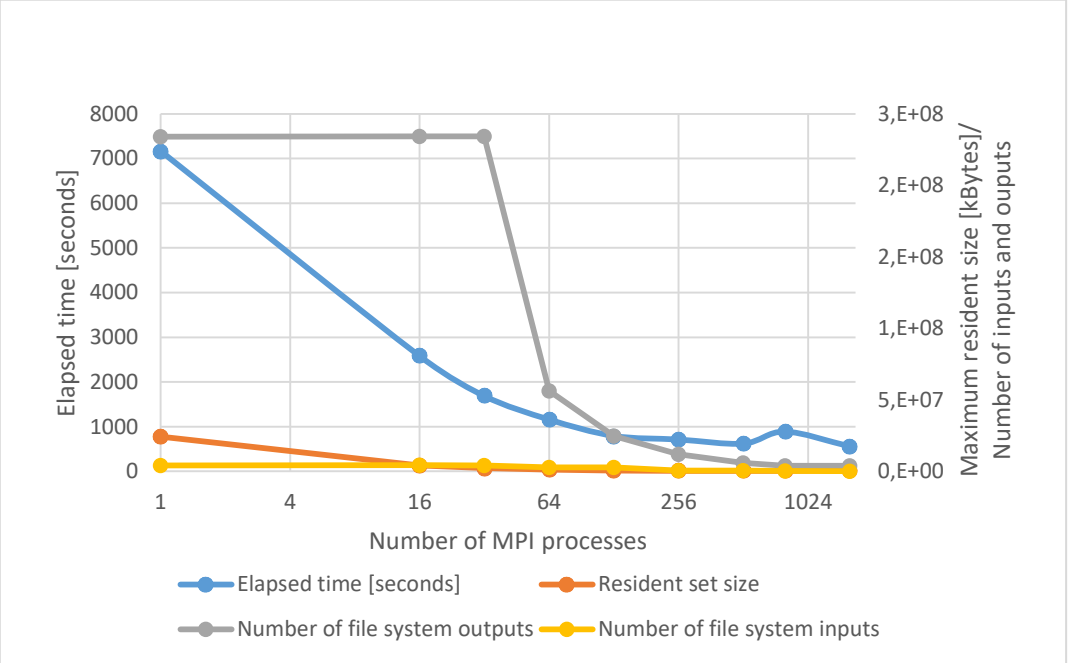

Fig. 16 Pandora results on 50-node Eagle (E5-2682 v4) cluster (WW map) without cache clearing

Similarly to Xeon E5-2697 v3 the winner for the EU map is the case when all cores from the chipset are used: 28 and 32, respectively. Applying more nodes and cores does not improve the results. The application stops scaling. For WW map, though, the best result has been achieved around 512 processes and then again at number 1600.

Xeon E5-2682 v4 with cache clearing

EU map

Tab. 12 Pandora results on 50-node Eagle (E5-2682 v4) cluster (EU map) with cache clearing

Eagle (E5-2682 v4)/ EU map with cache clearing				
#MPI procs	%e	%M	%I	%O
1	233,23	1012604	437968	7660576
16	69,35	140564	440456	7668536
32	46,88	94140	440456	7583832
64	71,84	86660	440456	1488680
128	106,58	108244	440048	543024
256	111,96	158432	440456	338648

512	184,57	258468	440456	131272
800	174,97	352928	440600	108232
1600	196,41	646640	433992	109800

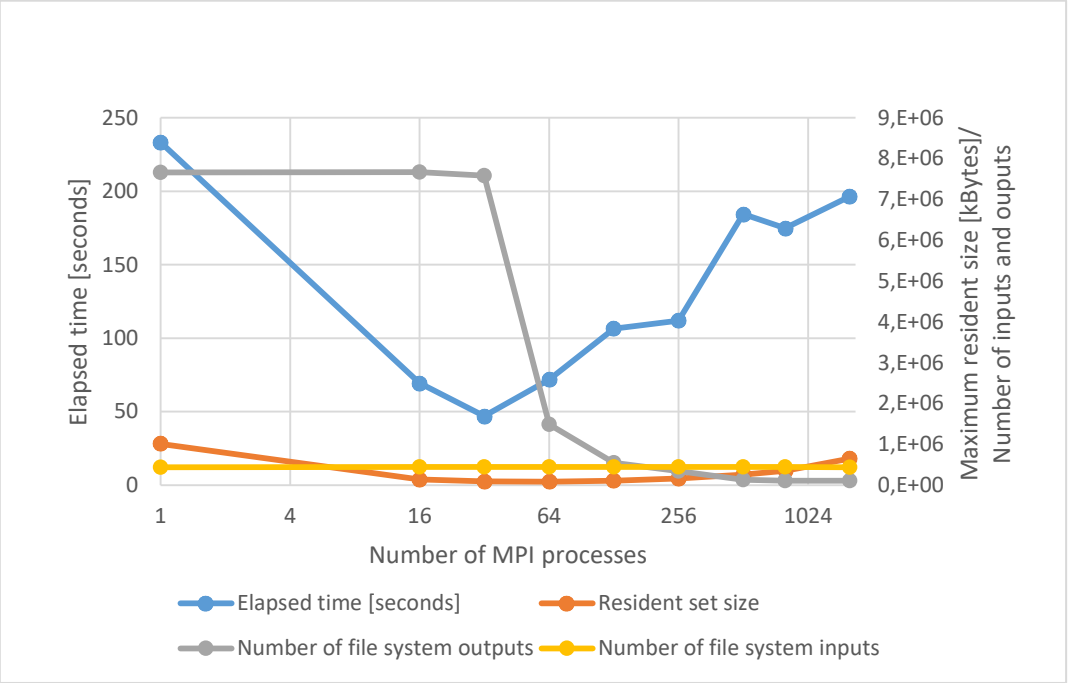

Fig. 17 Pandora results on 50-node Eagle (E5-2682 v4) cluster (EU map) with cache clearing

Similarly to the case without cache clearing, the sweet spot belongs to a single node and 32 MPI processes configuration. Above this number of 32 MPI processes the result values are worser and worser, i.e the application stops scaling. Comparing two tested Xeon E5 architectures, the winner for GG-pilot application is E5-2682 v4.

Power8 without cache clearing

EU map

Tab. 13 Pandora results on Power8 (EU map) without cache clearing

	Power8/ EU map			
#MPI procs	%e	%M	%I	%O
1	161,14	865984	0	5218688
4	92,02	342272	205328	5255936
8	54,27	227264	0	5257984
16	29,40	158080	0	5265664
32	30,43	114944	0	5300480
64	34,12	108864	0	5520896
128	47,64	138816	0	5936256
160	107,63	159232	0	6750720

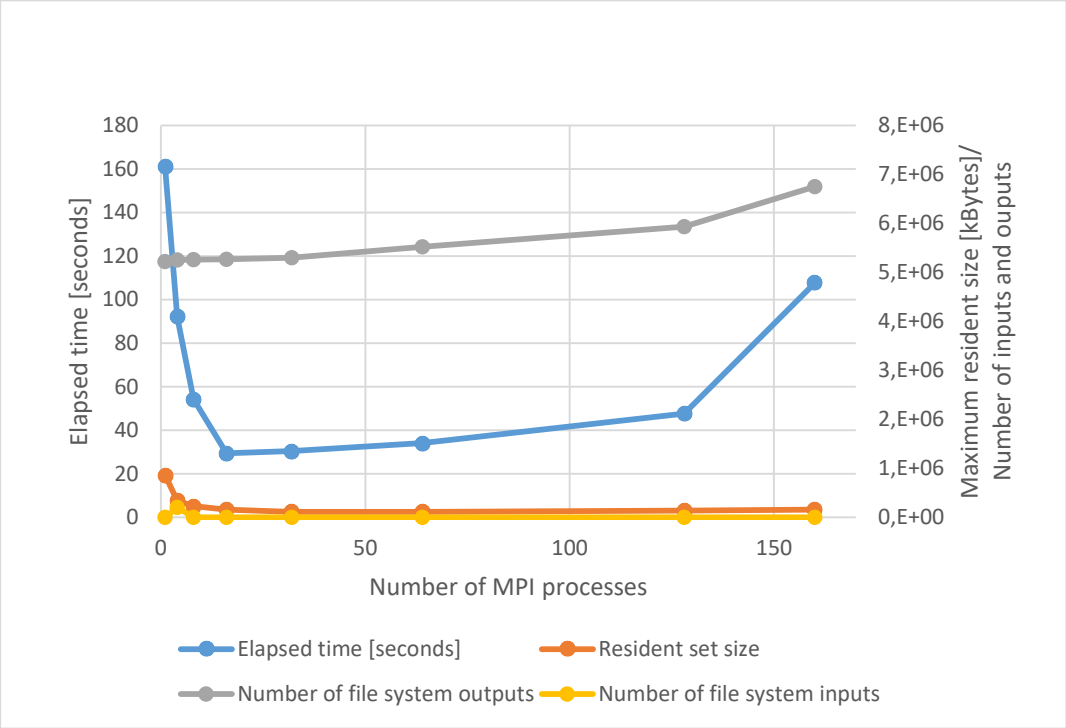

Fig. 18 GG-pilot w/Pandora scalability on Power8 (EU map) without cache clearing

Power8 with cache clearing

EU map

Tab. 14 Pandora results on Power8 (EU map) with cache clearing

	Power8/ EU map with cache clearing			
#MPI procs	%e	%M	%I	%O
1	173,77	865984	312272	5218688
4	129,31	484288	310560	5250944
8	64,74	220992	294216	5265664
16	37,96	158464	295616	5287168
32	47,78	115328	279616	5392768
64	52,60	108032	289440	5585536
128	77,37	138688	309688	6175232
160	131,21	150464	311056	7059328

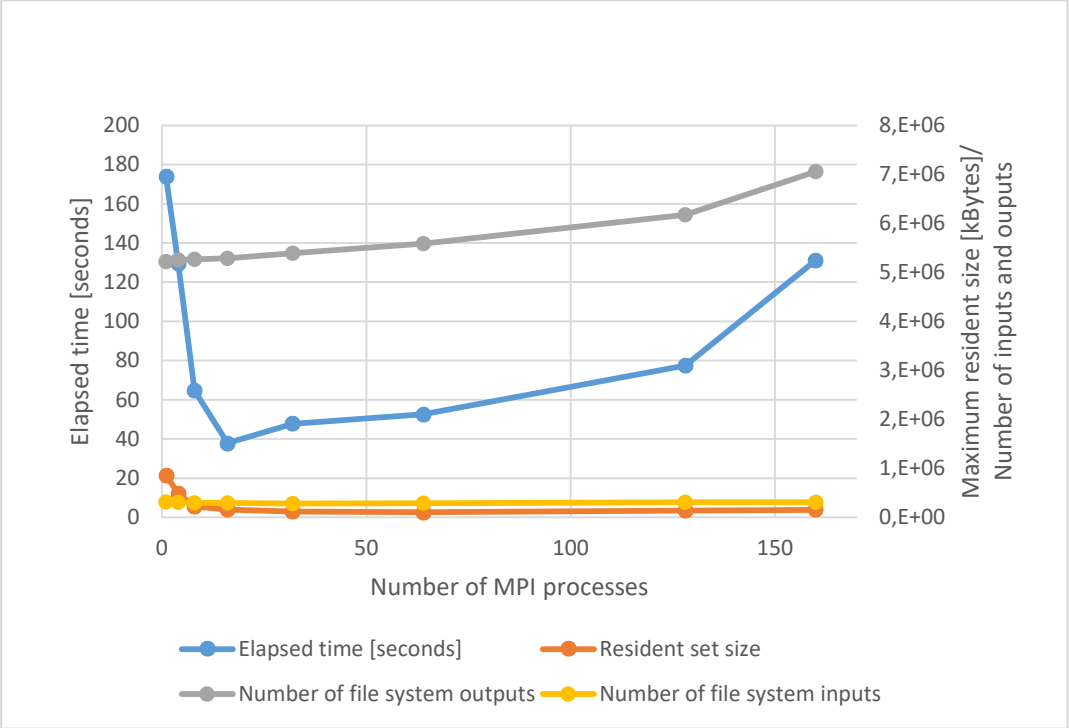

Fig. 19 GG-pilot w/Pandora scalability on Power8 (EU map) with cache clearing

Elapsed time of the simulation dramatically decreases with the number of processes and gains its minimum if the number of MPI processes is between 16 and 20. It is an expected result since the number of cores in Power8 node is equal to 16 (though the maximum number of threads is 10 times higher). In addition, there are several factors that significantly contribute to the final values. One of them is a fast drop in use of the memory resources until the number of processes reaches 32 (comparing with the diagram for the maximum resident set size of the application during its lifetime). After a large plateau between 32 and 64 processes, memory usage starts to grow when the number of MPI processes reaches the value of 64. On the other hand, data output constantly increases, and this growth starts to be relevant when the number of processes is more than 16. At that point, data output becomes one of the dominant contributors to the overall elapsed time. It can be seen from comparing plots for elapsed time and data output. They basically have the same shape if the number of processes is more than 32. Data input has a rather unpatterned behaviour. It varies randomly around 290MB. Nevertheless, one can neglect its contribution to the elapsed time since the fraction of input in the total I/O operations is less than 6%.

6.2 IPF

ARM without cache clearing

Tab. 15 IPF scalability on ARM without cache clearing

	ARM			
#MPI procs	%e	%M	%I	%O
1	4226,00	12512264	0	8
16	293,59	797204	0	35000
32	172,68	408012	32	67840
64	93,43	213220	0	68528

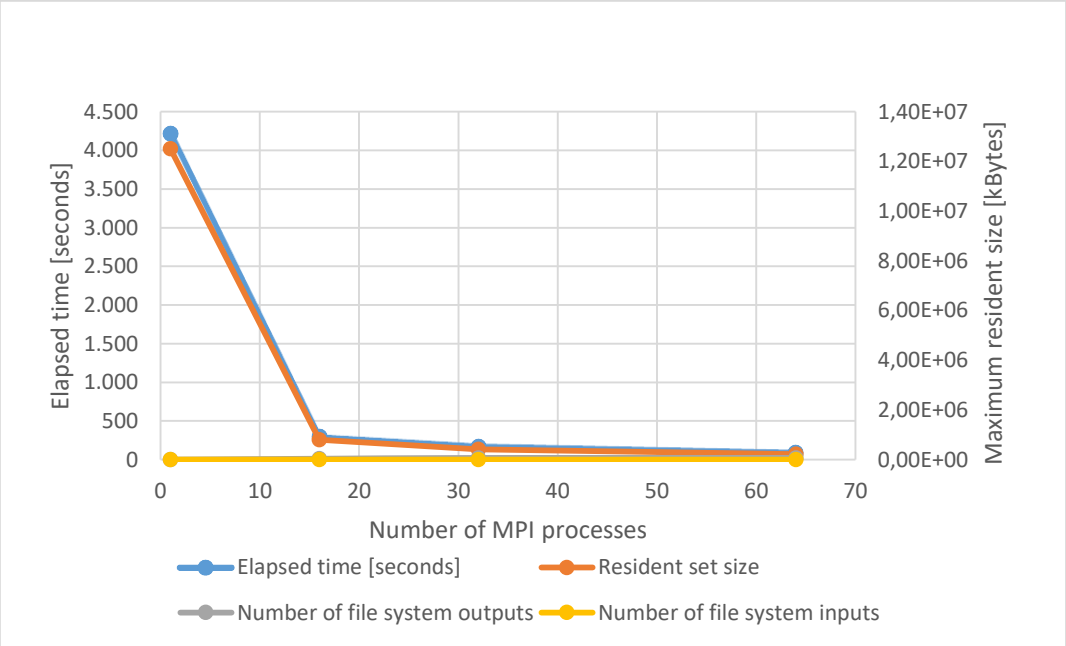

Fig. 20 IPF scalability on 2-node ARM system

ARM with cache clearing

Tab. 16 IPF scalability on 2-node ARM system with cache clearing

	ARM with cache clearing			
#MPI procs	%e	%M	%I	%O
1	3678,19	12512528	22720	8
16	293,07	798980	22712	33512
32	177,49	412580	23064	68176
64	94,04	212480	23112	64328

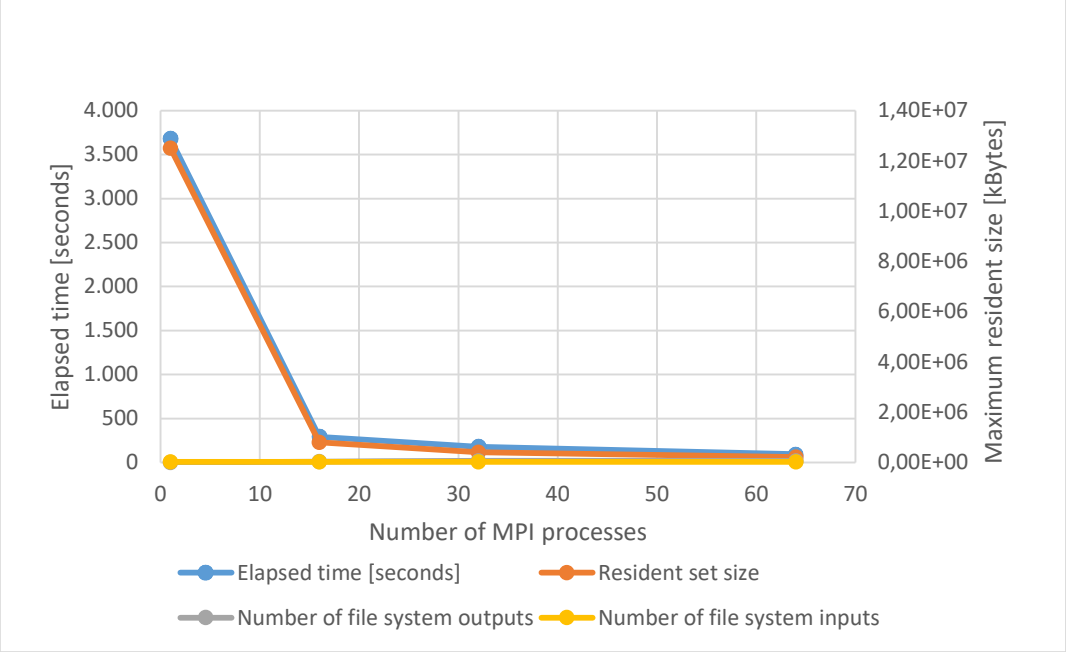

Fig. 21 IPF scalability on 2-node ARM system with cache clearing

For ARM testbed, achieved results turned out to be best in the context of execution time when both nodes have been used (64 cores). Despite a single case (1 MPI process) the results are insensibly better when cache is not cleared before each measurement.

Xeon Phi™ 7250 without cache clearing

Tab. 17 Xeon Phi 7250 results for IPF without cache clearing

	Xeon Phi™ 7250			
#MPI procs	%e	%M	%I	%O
1	3285,01	12522648	5448	0
16	317,19	816144	272	0
32	134,30	432160	0	0
64	131,09	234192	0	0
128	162,54	129528	0	0
272	157,34	76256	240	0
544	131,58	57280	0	0

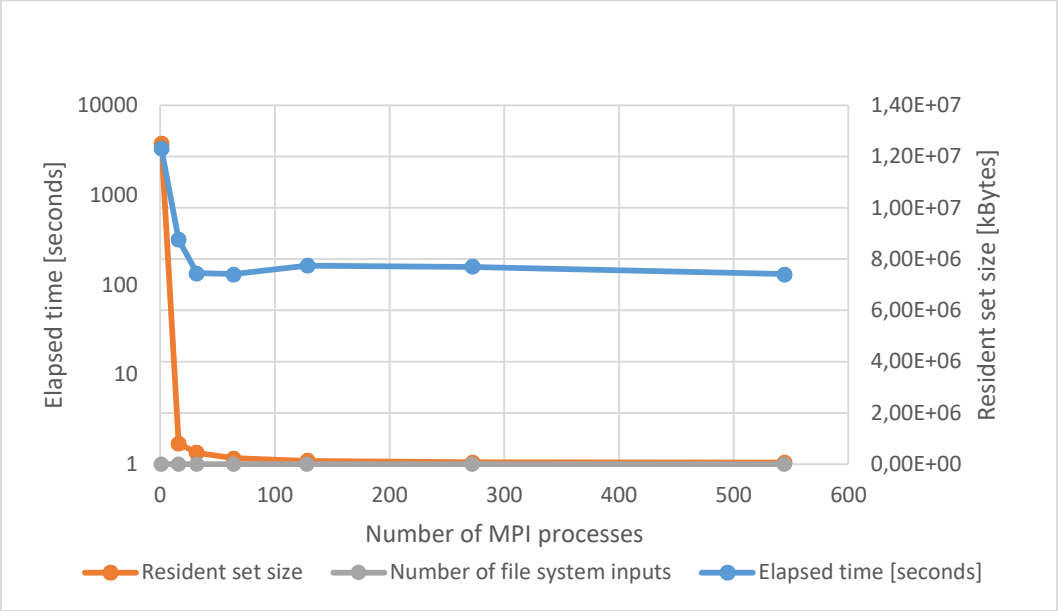

Fig. 22 IPF scalability on 2-node Xeon Phi™ 7250 system

Xeon Phi™ 7250 with cache clearing

Tab. 18 Xeon Phi™ 7250test results of IPF with cache clearing

	Xeon Phi™ 7250 with cache clearing			
#MPI procs	%e	%M	%I	%O
1	3290,81	12543028	15728	0
16	245,76	820168	21888	0
32	93,45	431384	21784	0
64	67,36	226244	21344	0
128	137,83	127188	21408	0
272	153,06	73328	21608	0
544	134,30	55476	29088	0

It is interesting to see that for the cache clearing case the results are much better for 32 and 64 MPI processes. In other cases the additional time required to read input data from a file makes the execution time obviously longer. However, the differences are non-significant.

Xeon E5-2697 v3 without cache clearing

Tab. 19 IPF scalability on 100-node Eagle (E5-2697 v3) cluster

	Eagle (E5-2697 v3)			
#MPI procs	%e	%M	%I	%O
1	664,71	664,71	3776	112
28	32,41	4892	10984	24
56	14,86	4992	10984	16
112	7,43	5192	10984	24
224	5,27	5552	10984	40
448	4,54	6288	0	8

700	3,69	7096	0	8
1400	5,16	7092	0	224
2800	7,37	7280	0	8

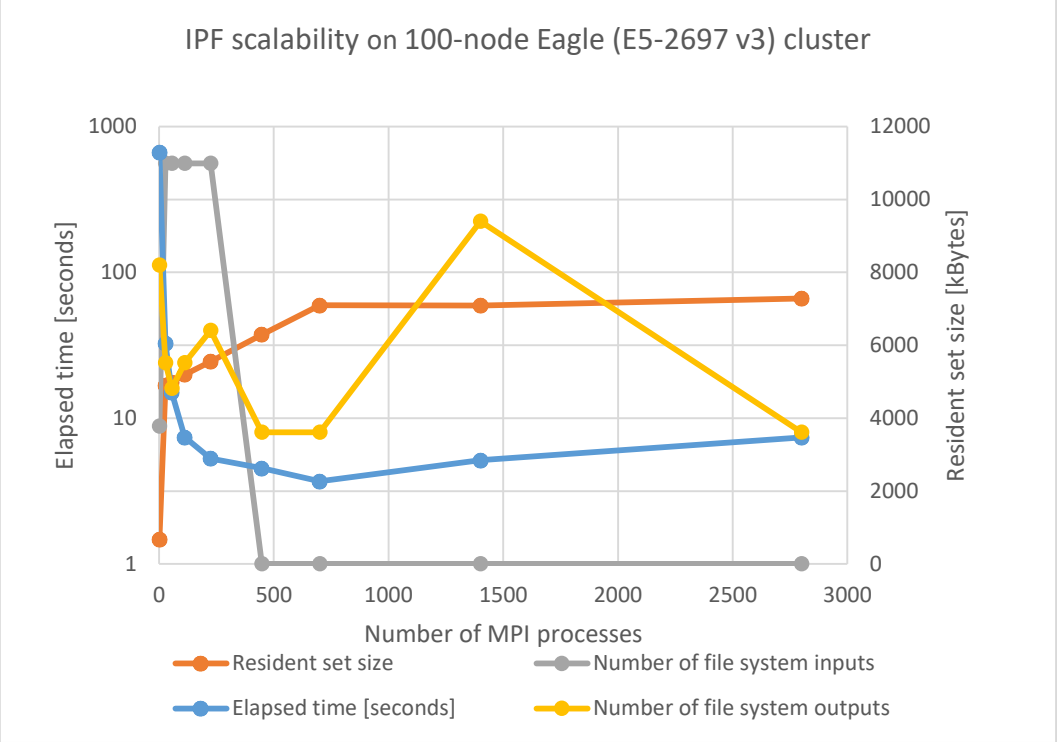

Fig. 23 IPF scalability on 100-node Eagle (E5-2697 v3) cluster

IPF on E5-2697 v3 scales quite well. Optimal results for given input data have been achieved for the number of 700 MPI processes. Above that number applying more resources does not bring any positive influence on the execution time.

Xeon E5-2682 v4 without cache clearing

Tab. 20 IPF results on Xeon E5-2682 v4 without cache clearing

	Eagle (E5-2682 v4)			
#MPI procs	%e	%M	%I	%O
1	645,57	4864	0	96
16	49,66	4864	0	16
32	33,00	4864	0	8
64	13,32	5000	0	16
128	8,27	5192	0	8
256	5,72	5552	0	8
512	5,02	6268	0	8
800	5,23	7084	0	8
1600	7,93	7096	0	8

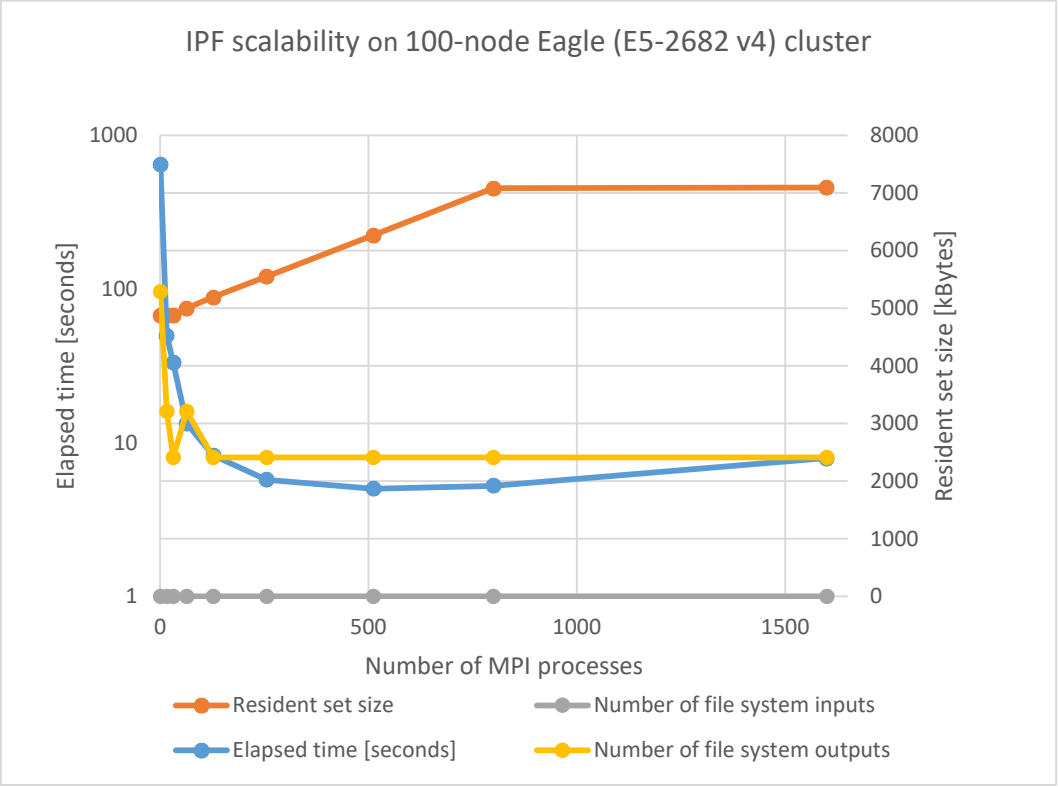

Fig. 24 IPF scalability on 100-node Eagle (E5-2682 v4) cluster without cache clearing

Xeon E5-2682 v4 with cache clearing

Tab. 21 IPF results on Xeon E5-2682 v4 with cache clearing

	Eagle (E5-2682 v4) with cache clearing			
#MPI procs	%e	%M	%I	%O
1	632,23	4920	12072	96
16	46,74	4892	12072	16
32	31,16	4896	12072	8
64	14,01	4992	12072	8
128	9,26	5184	12072	8
256	7,39	5552	12072	8
512	7,54	6272	12072	8
800	8,22	7096	12072	8
1600	10,73	7084	12072	8

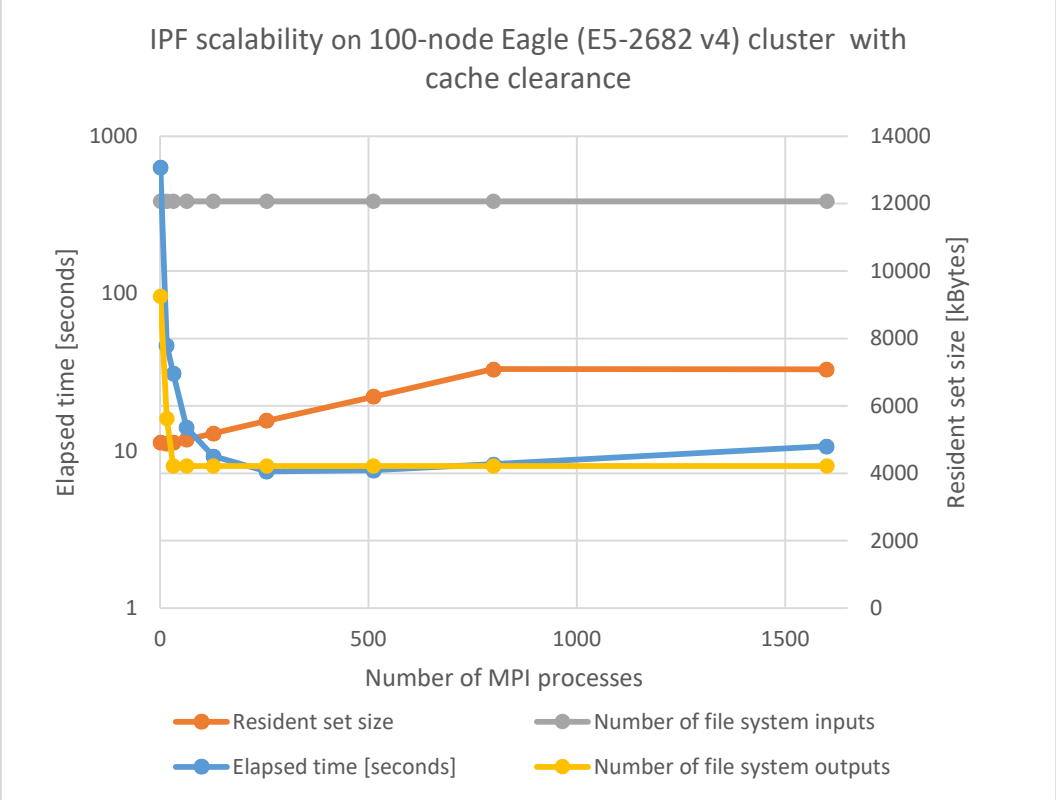

Fig. 25 IPF scalability on 100-node Eagle (E5-2682 v4) cluster with cache clearance

The above tests show that application scales up to 256 (with cache clearing) and 512 (when the cache is not cleared) MPI processes which depending on job configuration is equal to 8-16 nodes. Additionally it is worth noting that the number of input readings is constant.

Power8 without cache clearing

Tab. 22 IPF tests on Power 8 without cache clearing

	Power8			
#MPI procs	%e	%M	%I	%O
1	2230,70	12548416	0	128
4	311,66	3160256	0	14976
8	128,56	1599168	0	27264
16	59,74	849408	37040	47872
32	146,47	443968	0	108032
64	86,92	263872	0	218368
128	87,69	153344	0	624768
160	93,62	169344	0	940544

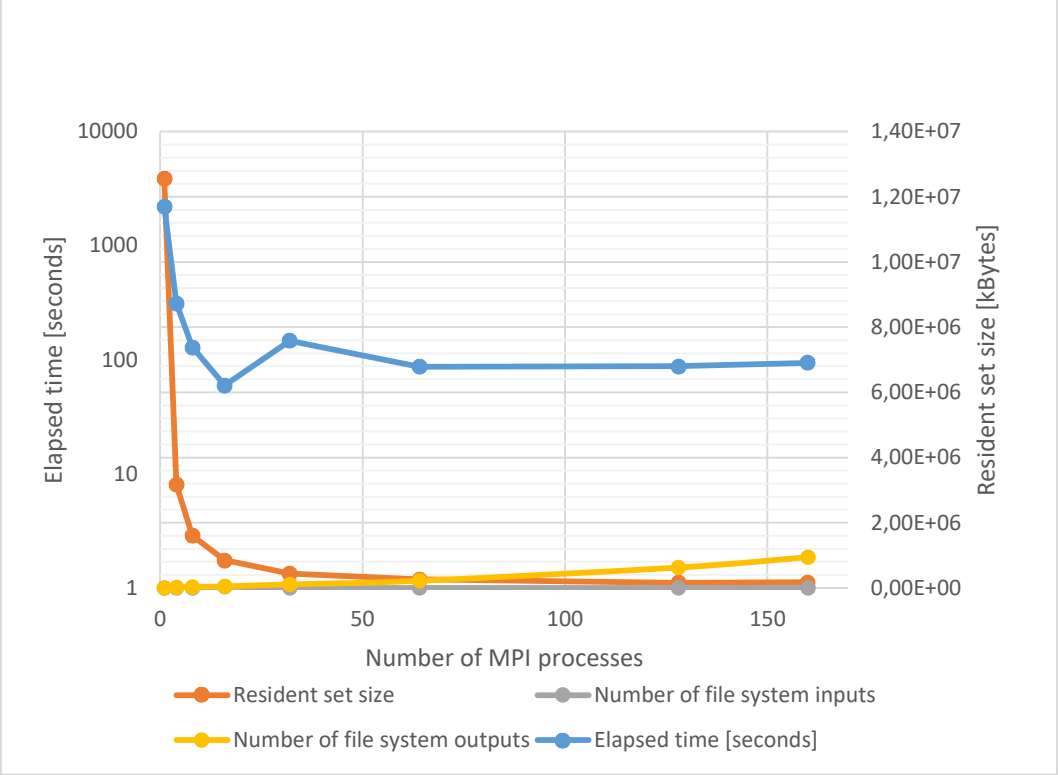

Fig. 26 IPF tests on Power 8 without cache clearing

Power8 with cache clearing

Tab. 23 IPF results on Power8 with cache clearing

	Power8 with cache clearing			
#MPI procs	%e	%M	%I	%O
1	2226,70	12532096	38712	128
4	309,91	3160256	39304	14336
8	127,09	1598976	38992	26752
16	59,63	817984	39504	43136
32	137,90	473792	39904	105728
64	97,01	250944	40808	236544
128	89,10	167744	40464	596480
160	86,49	149184	42064	881920

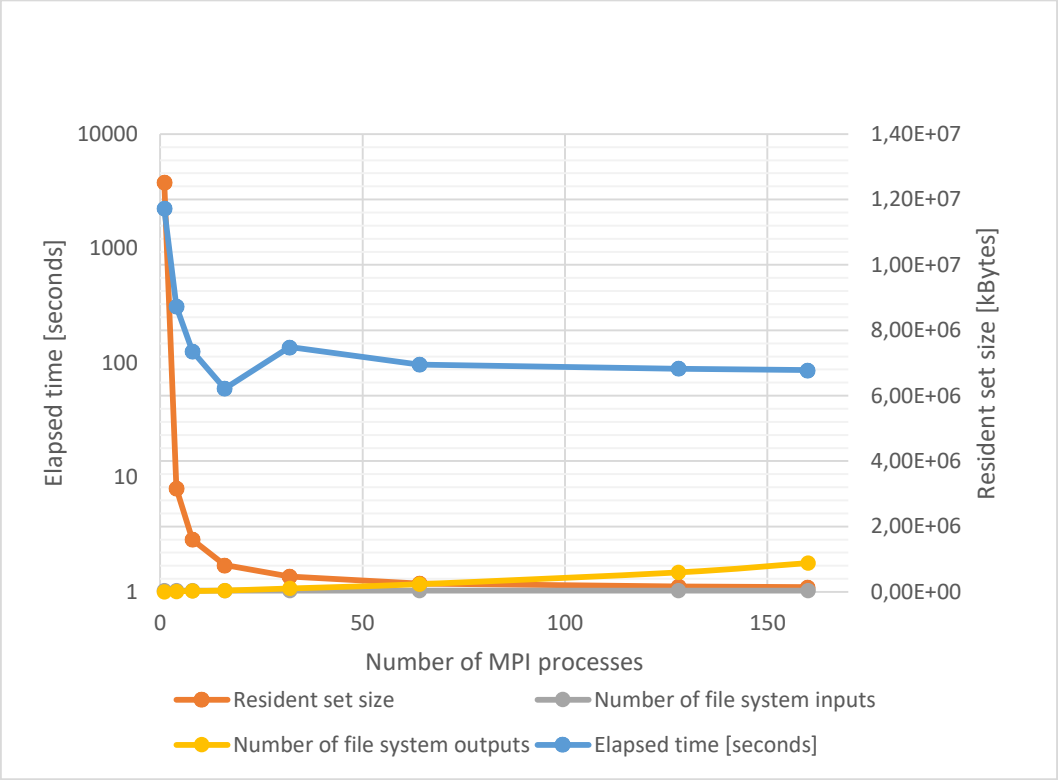

Fig. 27 IPF results on Power8 with cache clearing

Similarly to Pandora/GG-pilot benchmarks, benchmarks results for Power8 show that the elapsed time dramatically decreases with the number of processes until it reaches minimum between 16 and 20. Again, one of the key reasons is a significant reduction in memory consumption. However, in contrast to Pandora/GG benchmarks, afterwards we observe a little growth in elapsed time until the number of processes reaches 32, and a plateau with tiny fluctuations around 90 seconds for the number of processes between 36 and 160. With IPF, we have a plateau since the slight reduction in memory use compensates communication overheads related to the increasing number of processes. The number of I/O operations steadily rises with the number of processes. Nevertheless, due to a little fraction, I/O does not influence the overall elapsed time crucially.

6.3 Health habits (smoking prevalence)

For testing purposes we received two single process applications, the first of which was required for data import and conversion, while the second one took advantage of agents and Pandora library for its computations.

Pre-processing

Tab. 24 Smoking prevalence pre-processing application - systems behaviour for single process run on all architectures

1-node system architecture	%e	%M	%I	%O
Xeon Phi(TM) CPU 7250 @ 1,40GHz	149,75	669308	0	55064
ARM	50,22	650884	8	55048
Xeon(R) CPU E5-2697 v3 @ 2,60GHz	29,56	643784	513944	55040
Xeon(R) CPU E5-2682 v4 @ 2,50GHz	31,54	639948	513944	55040
Power8	26,69	687616	0	55808

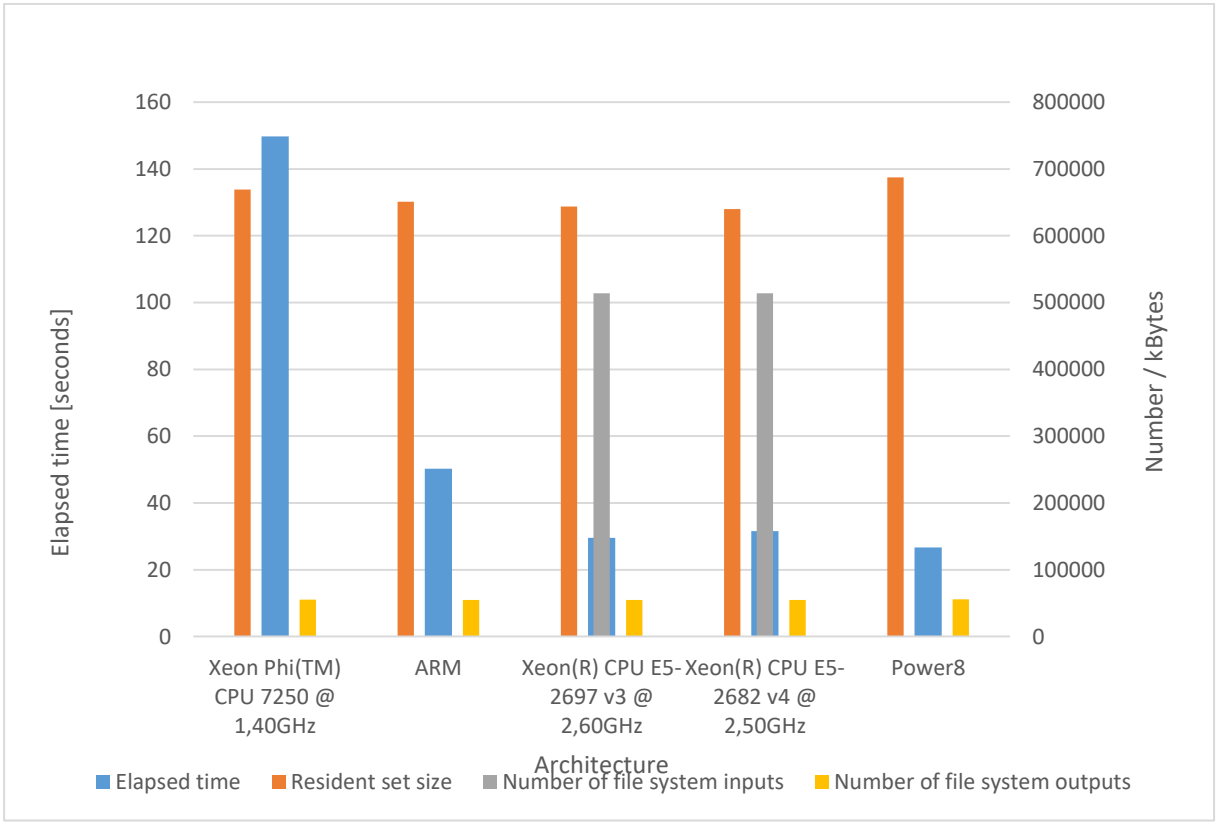

Fig. 28 Smoking prevalence pre-processing application - systems behaviour for single process run on all architectures

Pre-processing with cache clearing

Tab. 25 Smoking prevalence pre-processing application - systems behaviour for single process run on all architectures – option with cache clearing

1-node system architecture with cache clearing	%e	%M	%I	%O
Xeon Phi(TM) CPU 7250 @ 1,40GHz	151,63	668864	274040	55032
ARM	54,45	650352	234144	55032
Xeon(R) CPU E5-2697 v3 @ 2,60GHz	28,94	643788	520888	55040
Xeon(R) CPU E5-2682 v4 @ 2,50GHz	30,92	639936	520888	55040
Power8	32,46	702272	286136	55808

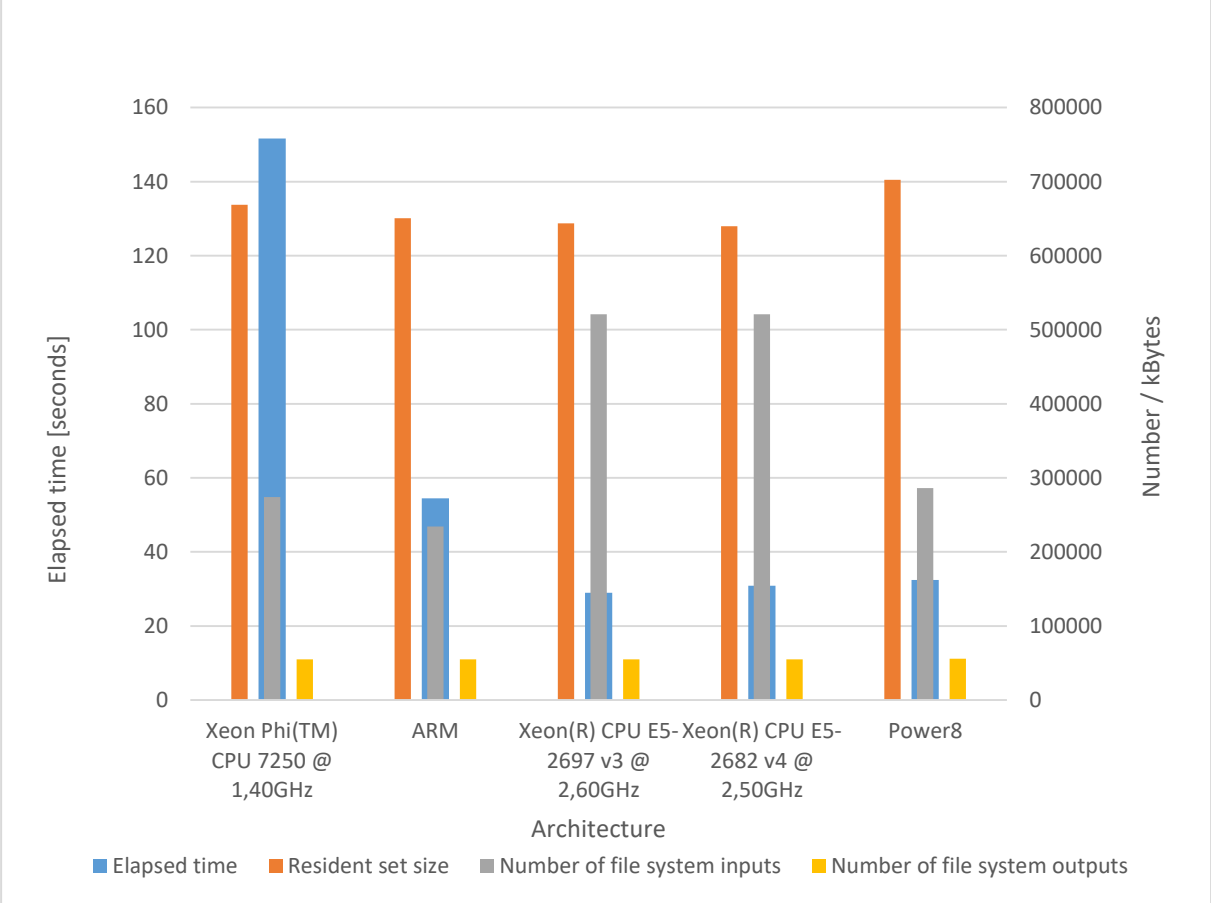

Fig. 29 Smoking prevalence pre-processing application - systems behaviour for single process run on all architectures – option with cache clearing

This pre-processing application differs from the others in many aspects. First of all, in contrast to Pandora and IPF, memory consumption dominates over I/O in this application. Moreover, it uses several times more data input that produces output. All in all, these aspects diminish advantages of the Power8 architecture, and we observe similar measurements for Power8 and Xeon E5 processors which take over other novel testbeds ARM and Xeon 7250. This is a perfect

example proving that a single process applications are not suitable for multicore/multithreaded processors. These ones, in turn, win in the context of the number of input data reading. Moreover, the elapsed time only slightly decreases if we switch off cache clearance.

Agents with Pandora library

Tab. 26 Results of smoking prevalence application w/Pandora- systems behaviour for single process run on all architectures – option without cache clearing

1-node system architecture	%e	%M	%I	%O
Xeon Phi(TM) CPU 7250 @ 1.40GHz	16059,55	3113744	256	40562264
ARM	5393,49	3011612	12496	39111712
Xeon(R) CPU E5-2697 v3 @ 2.60GHz	11611,78	5715500	181968	41398352
Xeon(R) CPU E5-2682 v4 @ 2.50GHz	12713,44	5715168	181968	41464192
Power8	2493,67	3050560	0	38855168

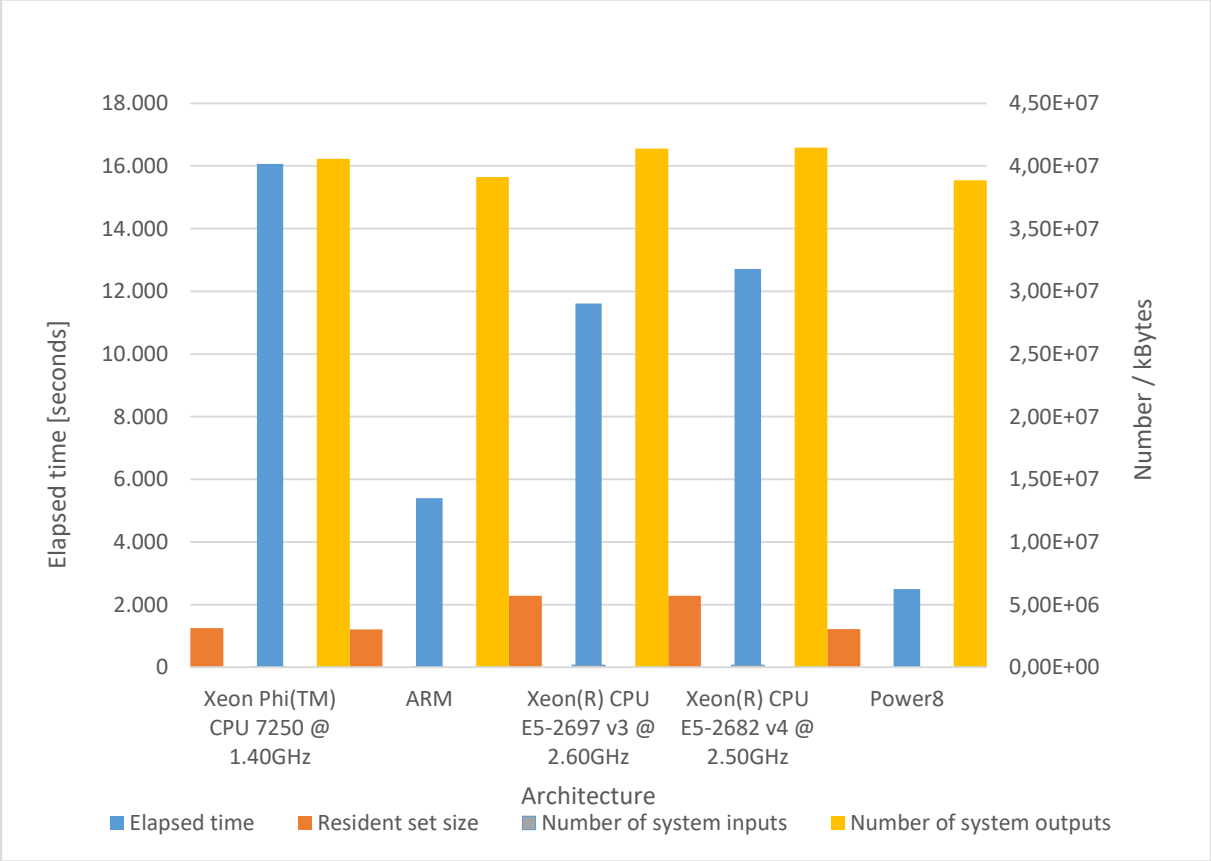

Fig. 30 Results of smoking prevalence application w/Pandora- systems behaviour for single process run on all architectures – option without cache clearing

Agents with Pandora library with cache clearing

Tab. 27 Results of smoking prevalence application w/Pandora- systems behaviour for single process run on all architectures – option with cache clearing

1-node system architecture with cache clearing	%e	%M	%I	%O
Xeon Phi(TM) CPU 7250 @ 1.40GHz	15909,13	3099104	86448	40037232
ARM	5238,36	3011592	106192	39087568
Xeon(R) CPU E5-2697 v3 @ 2.60GHz	11605,64	5715674	184854	41398352
Xeon(R) CPU E5-2682 v4 @ 2.50GHz	12063,87	5715236	190568	41322672
Power8	5016,37	3051200	143288	39083520

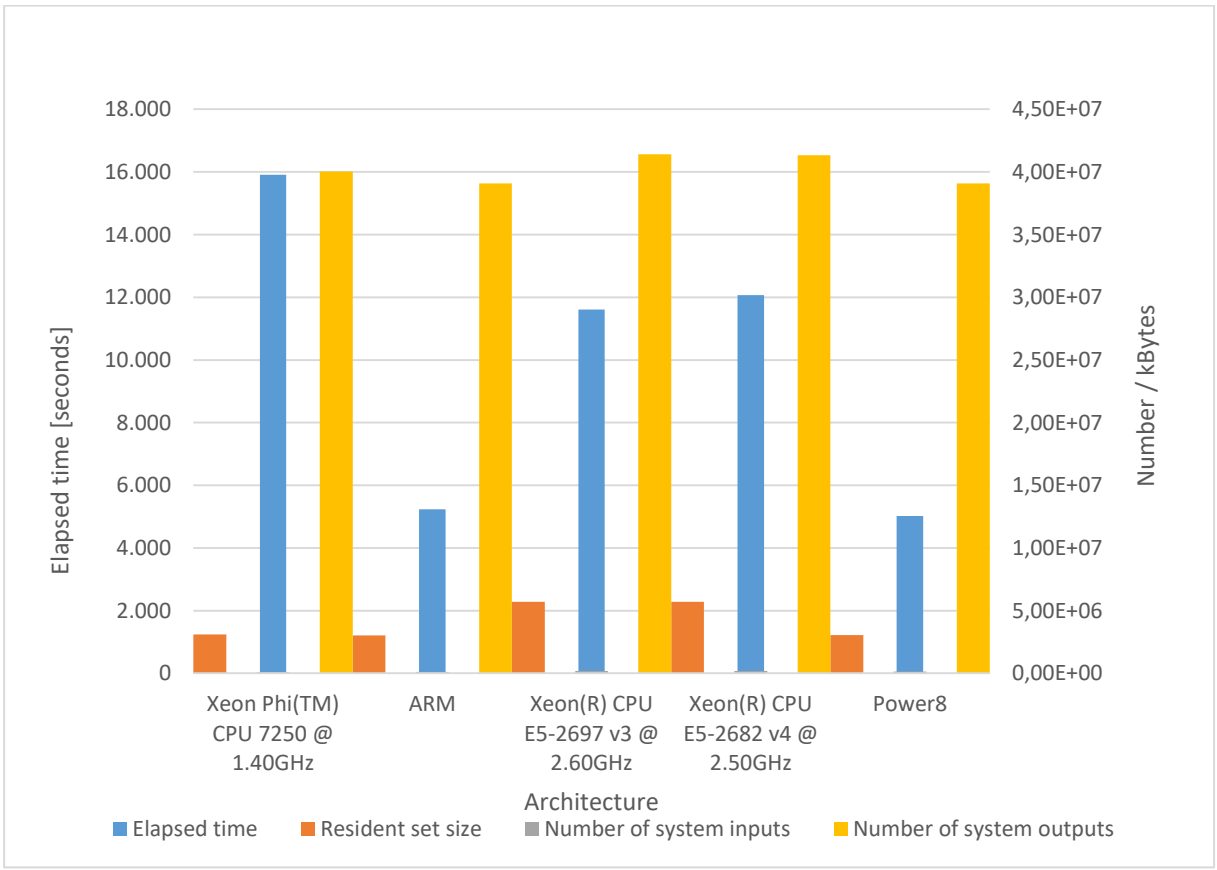

Fig. 31 Statistics of smoking prevalence application w/Pandora- systems behaviour for single process run on all architectures – option without cache clearing

Benchmarking results for Pandora health habits model are similar on Power8 and ARM. We observe less than 5% difference in elapsed time, I/O, and memory consumption. Nevertheless, the timing changes dramatically if we switch off cache clearance. In this case Power8

demonstrates two times smaller elapsed time which distinguishes this IBM product from other tested architectures.

6.4 WRF model

The scalability of the problem was reduced to 96 processors. The forecast was calculated for 2 days, 28th and 29th of August 2005, with accuracy to one hour. The input data comes from the Global Oceanic and Atmospheric Administration (GFS) model. The total input size for this test was 50GB.

For the benchmark purpose, HWRF was compiled with Intel's C and Fortran compilers from Intel Parallel Studio XE 2017 (17.0.1) and the latest version of the Intel MPI library (2017.1.32), Intel MKL (2017.1.32), HDF5 (1.8.18) as well as netCDF (4.4.1.1).

In this benchmark three architectures of the most popular and most powerful processors have been used. These are Intel Xeon Phi 7250 (Knights Landing), Intel Xeon E5-2697-v3 (Haswell) and Intel Xeon E5-2682 v4 (Broadwell).

Servers participating in the tests have been equipped with 128GB of fast DDR4 memory, NFS networked file systems for Intel Xeon Phi and Lustre for the other two architectures, which enabled efficient output file storage. All tests have been run with enabled buffers cleaning so that data is loaded each time from disks.

The tables and graphs below illustrate the results.

Tab. 28 HWRF results on Xeon Phi 7250 (KNL)

	Xeon Phi™ 7250			
#MPI procs	%e	%M	%I	%O
8	361,58	114580	129696	2597016
16	301,67	118140	128304	2895816
32	236,37	109820	129264	3493432
64	259,37	101584	129920	4690264
96	342,25	100700	130336	5890648

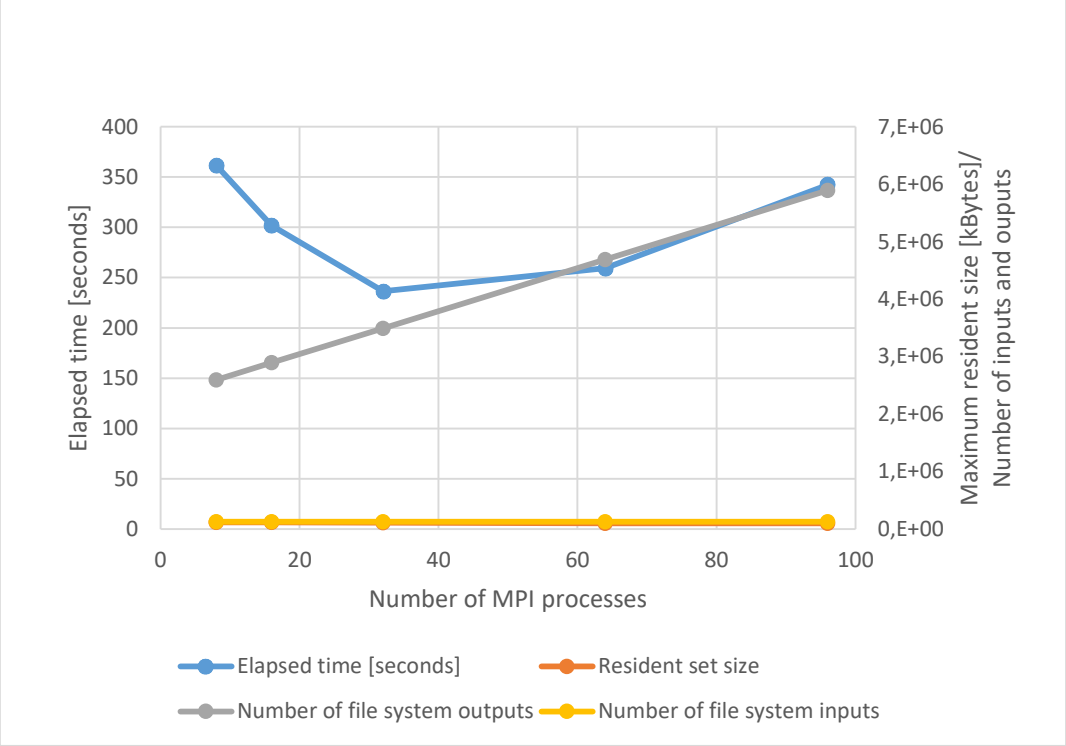

Fig. 32 HRWF results on Xeon Phi 7250 (KNL)

One can see there is almost linear scalability until 32 cores, then the computation is limited by some factor. Using multiple Xeon KNL machines does not make sense for this benchmark scenario.

Tab. 29 HWRF results on Eagle (Xeon E5 2697 v3)

	Eagle (E5-2697 v3)			
#MPI procs	%e	%M	%I	%O
4	934,99	4900	11504	8
8	530,76	4832	6312	8
16	352,85	4880	11744	8
32	227,23	4936	6312	8
64	165,25	4936	13520	8
96	148,34	4944	13520	16

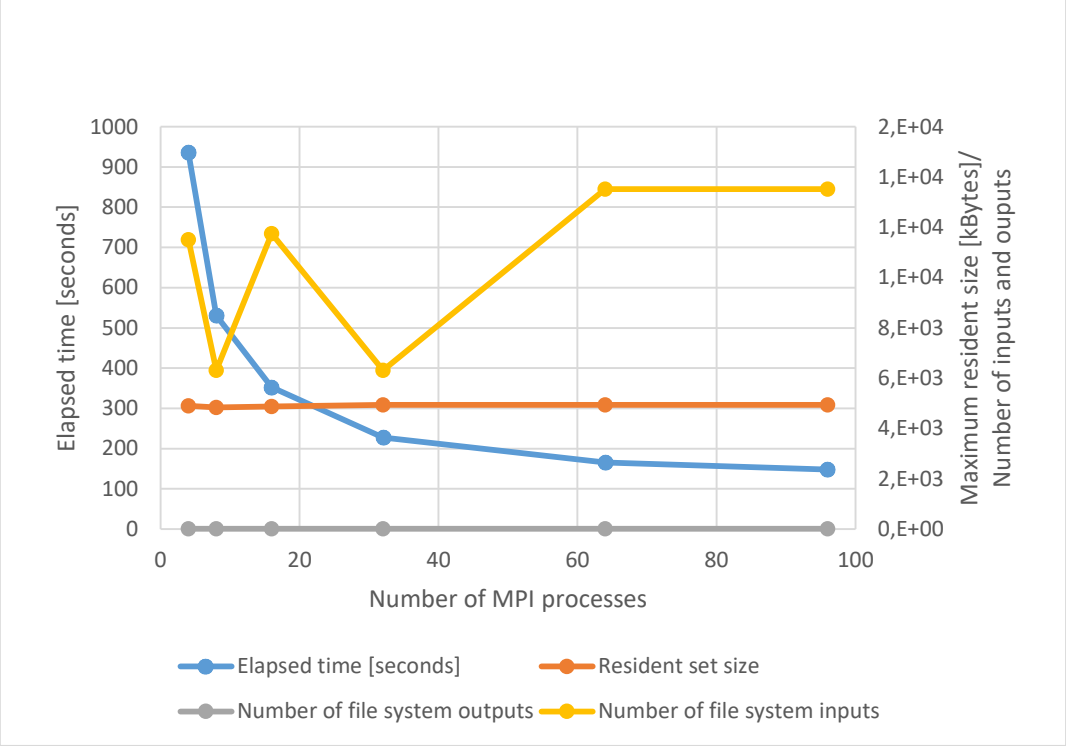

Fig. 33 HRWF results of scalability on single node Xeon E5 2697 v3

For dual Haswell server one can observe scalability beyond one server. Since each machine is equipped with 28 physical cores, the proposed split is not optimal as the tested scenario never used whole server.

Tab. 30 HWRF results on Eagle (Xeon E5 2682 v4)

	Eagle (E5-2682 v4)			
#MPI procs	%e	%M	%I	%O
4	928,02	4816	13536	8
8	537,45	4892	13536	8
16	350,00	4884	13536	8
32	277,25	4892	13536	8
64	155,54	4928	13520	8
96	132,66	5004	13520	8

The Broadwell family Xeon CPU used for benchmarking is better suited for the configuration as it directly reflected the hardware configuration. Similarly to the Haswell run, in this example one can see scalability beyond 1 server. The network is probably not a limiting factor in this case as the performance gains drop even on 32 cores – one physical server. Therefore, we can draw a conclusion that probably the problem size is too small to show the scalability or the memory access pattern is the limiting factor.

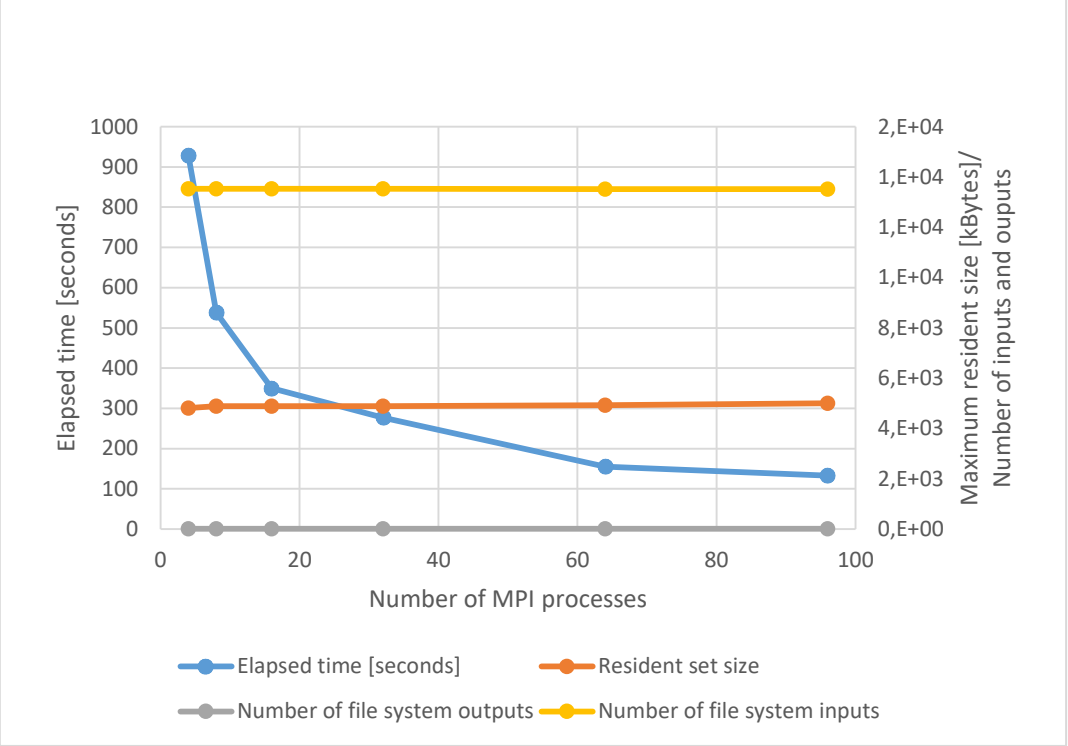

Fig. 34 HRWF results on single node Xeon E5 2682 v4

The Intel Knights Landing computation time decreases in function of the number of cores. After 32 cores it starts to grow slowly. The situation is different for the other two Intel Haswell and Intel Broadwell processors, where the calculation time decreases in exponential function (Fig. 32).

Interesting scalability level can be observed for calculations on 8, 16 and 32 cores for Intel Xeon Phi 7250 processor, where the performance is much better than the Intel Broadwell and Intel Haswell. This may lead to the conclusion that given a proper problem size and right distribution of the workload between the processor cores, Intel Knights Landing yeld better performance results comparing server-to-server with classical Xeon solutions. The results for the Intel Haswell and Intel Broadwell processors are very similar, but slightly better for the Intel Broadwell processor, resulting from the newer architecture of the processor.

7. Summary

Deliverables D5.7 and D5,8 (at month 33) aim to propose a set of applications which will define a benchmark relevant for the GSS community to check and choose the most relevant computing hardware, especially HPC platforms. These defined benchmarks assess the relative **performance** of a computing system, e.g. floating point operation performance of a CPU, efficiency of interconnects (for larger problems in distributed environment) or the speed of I/O operations.

The analysis of results can take into account not only efficiency of running benchmarks (end user point of view) but also financial aspects, financial analysis of performed tests in terms of efficiency of calculations in relation to energy consumption (the service provider's point of view).

The D5.7 is taking into account the end user requirements which are limited to application efficiency measured as speed and scalability.

For purposes of this deliverable numerous tests have been performed using the following use cases:

- GG (Green Growth) application using two maps with different resolution: European map (EU map) with the size of 640x680 and worldwide map (WW map) with the size 8640x3432 processes
- IPF application with input file input_40k_3200M and test iterations equals to 5. ROWBLOCK and COLBLOCK parameters were set to 4
- Health habits (smoking prevalence) pre-processing stage and rasterization application using Pandora library with map size of 1000x800
- Weather Research and Forecasting (WRF) model.

Each use case has been tested for the following various novel architectures:

- Intel Xeon E5-2697 v3 100-node cluster as the reference architecture
- Intel Xeon E5-2682 v4 50-node cluster
- Intel Xeon Phi 7250 2-node cluster
- ARM 2-node cluster
- IBM Power8 8247-22L single node
- Due to various number of nodes (some architectures were delivered by Intel with limited size, some others were available remotely also under limited number of nodes) and thus number of available to launch MPI processes it was numerically and directly hard to compare identical use case across all available testbeds. But at least within the same benchmark the results achieved on various cpu architectures can be compared.

In addition, it is possible to compare the scalability of particular application in the context of execution time, memory usage and I/O operations on different architectures:

- GG-pilot applications are dominated by /O operations (mostly output) where a large HDF5 file is created and to which all processes save data,
- From all tested applications IPF indicates the least influence of cache clearance (or its lack) on achieved results
- Health habits (smoking prevalence) pre-processing application is an only single process case compared to other presented benchmarks. It is dominated by memory consumption and almost constant disk output.
- The WRF model compilation was limited only to Intel architectures; authors of these tests did not succeed to achieve a running version on ARM.

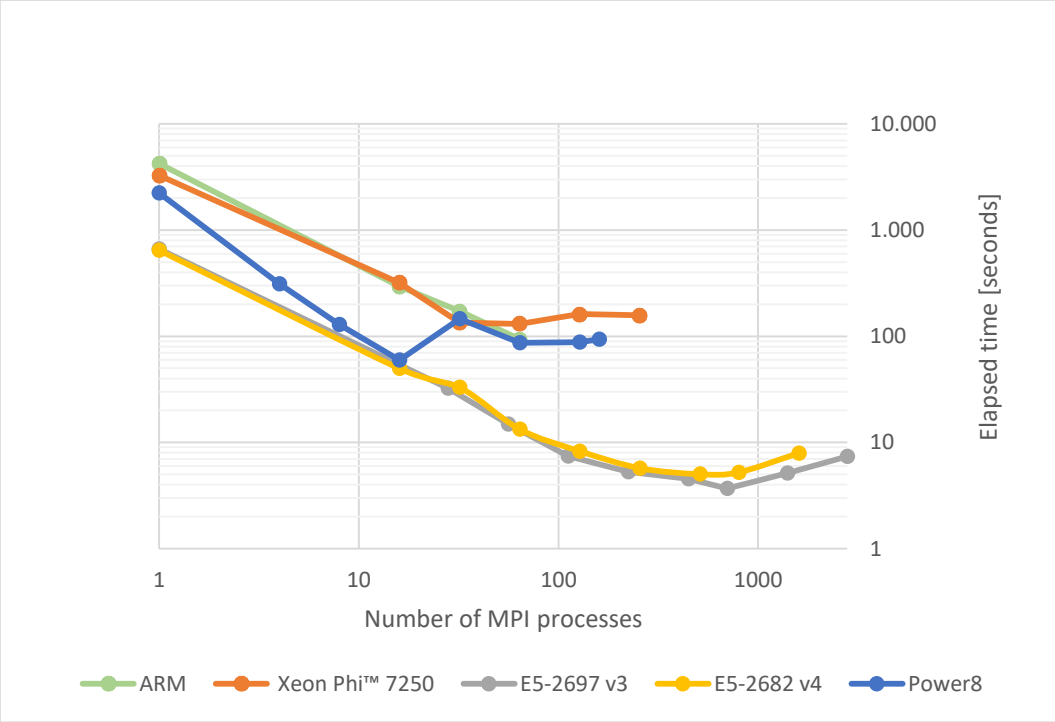

Fig. 35 IPF scalability results for all architectures

Referenced Xeon E5 2697 v3 processor and Xeon E5 2682 v4 in all tested applications showed their good scalability in terms of execution time when compared to other architectures. The less number of MPI processes used, the advantage is better visible.

Xeon Phi 7250 turned out not a good choice for all tested applications except the WRF model, what may surprise. The above charts (Fig. 35, Fig. 36, Fig. 37) show that the processor having 68 cores (4 threads per core) on board, reaches its optimum performance only for single thread per core. A different situation can be observed for calculations on 8, 16 and 32 cores for Intel Xeon Phi 7250 processor and WRF benchmarks, where the performance is much better than the Intel Broadwell and Intel Haswell. This may lead to the conclusion that given a proper problem size and right distribution of the workload between the processor cores, Intel Knights

Landing yeld better performance results comparing server-to-server with classical Xeon solutions.

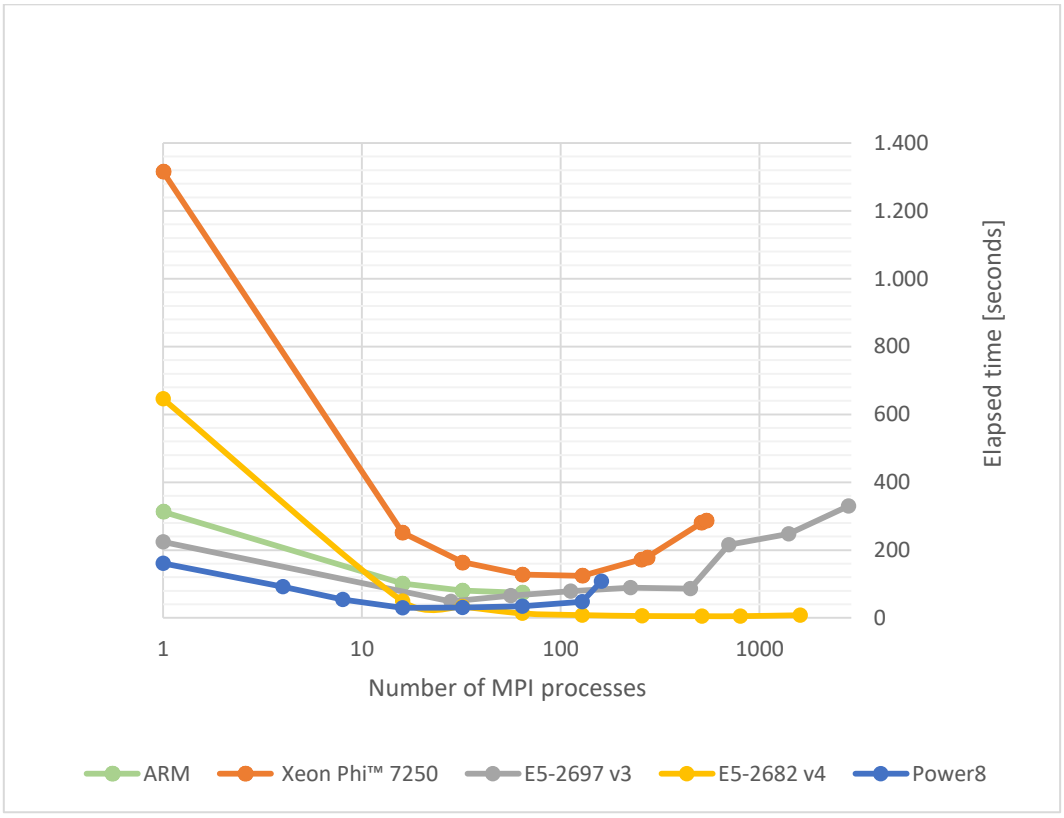

Fig. 36 Pandora EU map results for all architectures

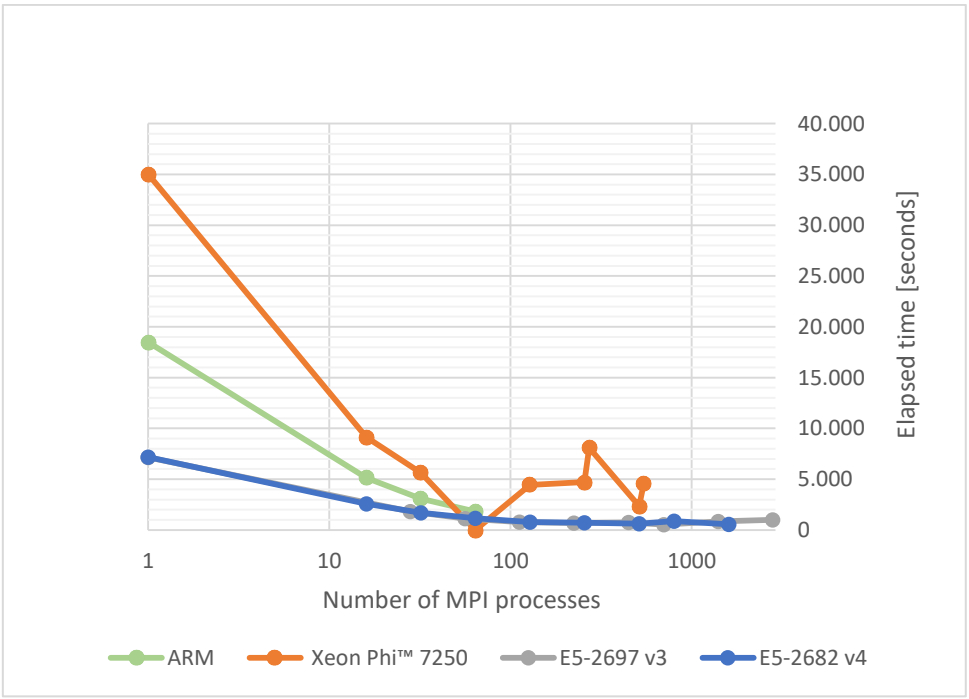

Fig. 37 Pandora WW map results for all architectures

The ARM architecture seems to be a moderate choice for GSS applications. Despite quite good scalability, the execution times place this processor somewhere in the middle of the group. However, adding additional parameters like price and power consumption places this architecture on a good position worth to be taken into account by GSS apps.

Power8 is particularly suitable for application where data output dominates over data input and memory consumption. Both Pandora models (health habits and green growth) demonstrate exactly this behaviour. That is why Power8 shows better results with Pandora models compared to other architectures. Moderate results can also be observed when benchmarking Power8 with IPF application. On the other hand, when data rastering application comes into play, it proves Power8 is less efficient with use cases extensively consuming memory.

Benchmarks showed, Power8 node usually gives the minimum elapsed time for the number of processes between 16 and 20.

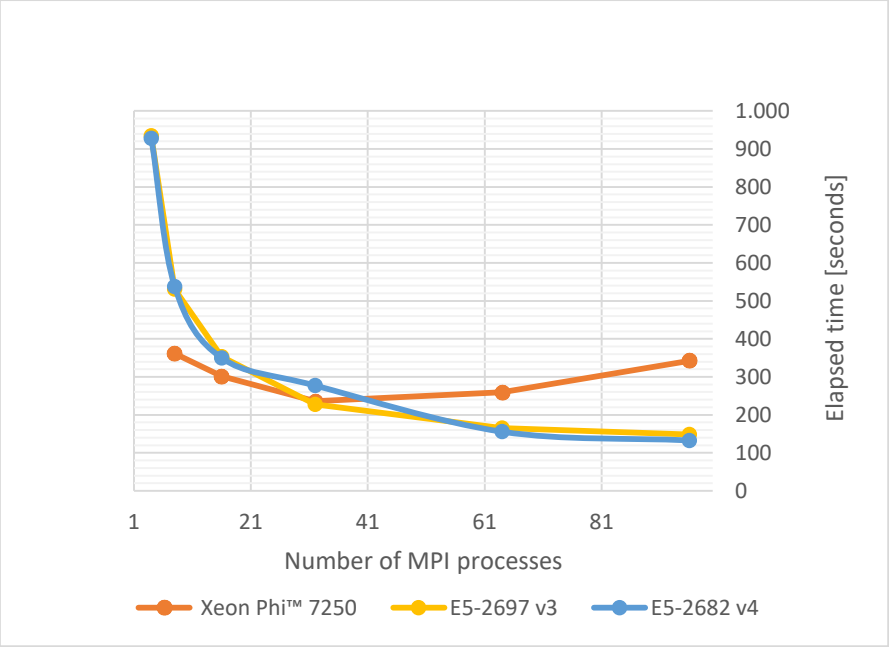

Fig. 38 HRWF scalability across all tested architectures

The D5.7 report is the first step towards producing a good set of GSS benchmarks and choosing the best architecture for such kind of computations. The first analysis did not take into account all issues like financial aspects, power consumption, support etc. The analysed values and parameters have been limited to end user requirements. Next steps will be performed under D5.8 report at month 33.

8. References

1. **benchmarks, Computing.** https://en.wikipedia.org/wiki/Benchmark_(computing)
2. **BAPCO.** https://bapco.com/
3. **(SPEC), Standard Performance Evaluation Corporation.** https://www.spec.org/benchmarks.html
4. **Performance, Transaction Processing.** http://www.tpc.org/
5. Michèle Weiland, Profile of scientific applications on HPC architectures using the DEISA Benchmark Suite, Computer Science - Research and Development, May 2010, Volume 25, Issue 1, pp 33–39, Springer-Verlag. [Online]
6. DEISA Benchmark Suite (2010) http://www.deisa.eu/science/benchmarking. [Online]
7. Global System Science. [Online] http://global-systems-science.eu/gss/content/introduction-gss.
8. *Unix tsch shell - https://en.wikipedia.org/wiki/Tcsh.* [Online]
9. *CoeGSS list of deliverables - http://coegss.eu/resources/#deliverables.* [Online]
10. *Socioeconomic Data and Applications Center (SEDAC) (2016) 'Gridded Population of the World, v4'. http://sedac.ciesin.columbia.edu/data/collection/gpw-v4.* [Online]
11. *OICA (2015) 'Vehicles in use'. http://www.oica.net/category/vehicles-in-use/.* [Online]
12. *World Bank (2015) 'GDP per capita (current US$)' http://data.worldbank.org/indicator/NY.GDP.PCAP.CD.* [Online]
13. *Rubio-Campillo, X. (2014) 'Pandora: A Versatile Agent-Based Modelling Platform for Social Simulation', in SIMUL 2014: The Sixth International Conference on Advances in System Simulation.* [Online]
14. Dargay, J., Gately, D. and Sommer, M. (2007) 'Vehicle Ownership and Income Growth, Worldwide: 1960-2030', The Energy Journal, 28(4), pp. 143–170. [Online]
15. *Rubio-Campillo, X. (2014) 'Pandora: A Versatile Agent-Based Modelling Platform for Social Simulation', in SIMUL 2014: The Sixth International Conference on Advances in System Simulation.* [Online]
16. *A Software Construction tool, http://scons.org/.* [Online]
17. *Socioeconomic Data and Applications Center (SEDAC), http://sedac.ciesin.columbia.edu/data/collection/povmap.* [Online]
18. *Socioeconomic Data and Applications Center (SEDAC) (2016) 'Gridded Population of the World, v4'. http://sedac.ciesin.columbia.edu/data/collection/gpw-v4.* [Online]

19. https://en.wikipedia.org/wiki/Hurricane_Katrina. [Online]

20.https://www.computer.org/cms/Computer.org/dl/mags/mi/2016/02/figures/mmi2016020034 1.gif. *Intel KNL* . [Online]

21. http://www.theverge.com/2016/9/5/12798302/softbank-arm-acquisition-complete

22. *Softbank company, http://www.softbank.jp.* [Online]

23. http://www.anandtech.com/show/9234/amds-20162017-datacenter-roadmap-x86-arm-and-gpgpu

24. https://www.apm.com/news/macom-successfully-completes-acquisition-of-appliedmicro/. [Online]

25. http://www.zdnet.com/article/macom-buys-applied-micro-for-770-million-plans-to-sell-arm-server-chip-unit/. [Online]

26. http://www.cavium.com/ThunderX2_ARM_Processors.html. [Online]

27. https://www.hpcwire.com/2016/12/08/qualcomm-targets-intel-datacenter-dominance/

28. http://www.hpcadvisorycouncil.com/events/2016/stanford-workshop/pdf/Goldstone.RoadtoCoral.LLNL.pdf. [Online]

29. https://3s81si1s5ygj3mzby34dq6qf-wpengine.netdna-ssl.com/wp-content/uploads/2015/07/ibm-openpower-roadmap-2.jpg. *IBM Power roadmap*

List of figures

List of tables