Horizon 2020
European Union funding
for Research & Innovation

D5.8 – SECOND 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	30.06.2018
Actual date of submission	30.06.2018
Dissemination level	Public
Nature	Report
Version	1.2
Work Package	WP5 (Centre Operation)
Lead beneficiary	PSNC
Responsible scientist/administrator	Norbert Meyer (PSNC)
Contributor(s)	Michael Gienger (HLRS), Sergiy Gogolenko (HLRS), Andreas Geiges (GCF), Damian Kaliszan (PSNC), Sebastian Petruczynik (PSNC), Radosław Januszewski (PSNC), Paweł Wolniewicz (PSNC)
Internal reviewer	Leonardo Camiciotti (Top-IX),

	Steffen Fuerst (GCF)
Keywords	GSS, Benchmarks, HPC, scalability, efficiency, exascale
Total number of pages:	60

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	28.02.2018
Version 0.1	Damian Kaliszan	PSNC	24.04.2018
Version 0.2	Damian Kaliszan	PSNC	20.05.2018
Version 0.3	Damian Kaliszan	PSNC	23.05.2018
Version 0.4	Damian Kaliszan	PSNC	29.05.2018
Version 0.4	Damian Kaliszan	PSNC	02.06.2018
Version 0.6	Damian Kaliszan	PSNC	04.06.2018
Version 0.7	Sebastian Petruczynik	PSNC	07.06.2018
Version 1.0	Sergiy Gogolenko	HLRS	08.06.2018
Version 1.1	Norbert Meyer, Damian Kaliszan	PSNC	25.06.2018
Version 1.2	Norbert Meyer	PSNC	28.06.2018
Version 1.3	Norbert Meyer	PSNC	29.06.2018
Approved by	ECM		02.07.2018

1. Abstract

This is a second and final report belonging to WP5 CoeGSS (Centre Operation) and task 5.2 (Preparing the Future) presenting the results of benchmarks with the aim to find the most suitable HPC architecture and the best processor which allows to run Global Systems Science (GSS) applications effectively.

As already stated in D5.7 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. D5.7 of the CoeGSS project has already defined three exemplary challenges as pilot studies: Health Habits, Green Growth, Global Urbanisation and has been already extended with the following applications:

- 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.

At this point, D5.8 extends this list by four additional programs:

- CMAQ/CCTM (Community Air Multiscale Quality Modelling System/ The CMAQ Chemistry-Transport Mode)
- CM1 (Cloud Modelling)
- ABMS (Agent based modelling and simulation)
- OpenSWPC (An Open-source Seismic Wave Propagation Code).

The above list seems to be quite rich and reflects the real GSS world as much as possible having in mind the real-world applications availability and CoeGSS project applications maturity.

Additionally, the authors have managed to acquire four novel architectures, such as Intel® Gold® 6140, AMD Epyc™ 7551, ARM Hi1616 and Power8+. Due to physical hardware availability, the testbeds consisted of one or two nodes only. This limited the ability of the authors to test full scalability for given applications. However, this little number of available computational units (cores) can provide valuable outcome including architecture comparison for different applications based on execution times, TDPs, processors prices. These are the basic metrics the authors used for ranking of architectures.

Finally, this document is thought to be a valuable information for the GSS community for future purposes and analysis to determine their specific demands as well as – in general - to help develop a mature final benchmark set reflecting the GSS environment requirements and speciality. As in the number of existing benchmarks there is none dedicated to the GSS community, the authors decided to create one by calling it *GSS benchmark* to serve and help GSS users in their future work.

2. Table of Contents

3. Introduction

Deliverable D5.8 is the second stage report trying to verify the use of new and fresh HPC architectures for running Global Systems Science (GSS) applications. GSS is quite a new branch of science using specific knowledge and techniques to evaluate the impact of policies and people's relation to various global phenomena such as climate change, financial crises, pandemics, and growth of cities – urbanization and migration patterns. Task 5.2 – to which deliverables 5.7 and 5.8 belong – is searching for the answer to the question "which HPC architectures among the recently introduced ones are the best to run GSS applications most effectively?". Of course, "the best" or "most effectively" aliases may have different meanings to different people. While for some it may be the fastest execution time, others may be interested in the price performance calculated as the price of a processor multiplied by total execution time (for given architecture) or the least carbon footprint left calculated as a product of TDP and total execution time.

For the purpose of this deliverable the authors acquired as many novel processors from vendors as possible. Along with those benchmarked in D5.7, the total range of tested architectures seems to be sufficient and substantial. These architectures are presented in section 5 and are as follows:

- Intel® Xeon® Gold 6140 2-node cluster,
- ARM Hi1616 2-node cluster,
- AMD Epyc™ 7551 single node,
- IBM Power8+ single node,

and – again – Eagle cluster with Intel Xeon Haswell E5-2697 v3 processors as a reference testbed.

The initial list of applications created in D5.7 was supplemented by the new codes from the various research areas covering the entire GSS field. Section 6 leads the reader through the description of whole range of tests carried out with graphical representation of the results achieved on above-mentioned hardware platforms moved to the Annex. It is worth to mention that output parameters measured for all tests are – similarly to these in D5.7 – as follows:

- **%e** Elapsed real time (in seconds; not in tcsh) (1);
- **%M** Maximum resident set size of the process during its lifetime, in KB;
- **%I** Number of file system inputs generated by the process;
- **%O** Number of file system outputs generated by the process.

All of the applications included in this benchmark were launched on each architecture. The only exception is HWRF (Hurricane WRF), which was not compiled on the ARM and Power8+ architectures, due to the scale of this application and the difficulty in compiling it. The characteristics of this application will be presented separately as a test of x86_64 architectures.

Section 7 of this document concludes the report, draws conclusions and creates the vision of a – what the authors call – 'GSS benchmark', which is going to fill the gap inside the whole range of different types of benchmarks and is trying to answer the questions raised in this deliverable.

3.1 Glossary of Acronyms and Terms

Acronym	Definition
ABMS	Agent based modelling and simulation
ARM	Advanced RISC Machine
CMAQ	Community Multiscale Air Quality Modeling System
CPU	Central Processing Unit
D	Deliverable
DoA	Description of Action
EC	European Commission
FLOPS	Floating Point Operations per Second
GB	Gigabyte
Gbps	Gigabit per second
GPU	Graphics Processing Unit
HDF5	Hierarchical data format version 5
HLRS	High Performance Computing Center Stuttgart
HPC	High Performance Computing
IPF	Iterative proportional fitting
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
RAM	Random Access Memory
Skylake	Intel's new generation CPU
SLURM	Simple Linux Utility for Resource Management, Workload Manager
TDP	Thermal Design Power
TFLOP/s	TeraFLOPS per second
WP	Work Package
WRF	The Weather Research and Forecasting

4. Description of applications used for GSS representation

This section contains the description of new applications used for the purposes of deliverable D5.8: OpenSWPC, ABMS, CCTM, and CM1. Summary, details, and installation processes of other benchmarks tested in D5.8, such as IPF, Green Growth using Pandora, and HWRF model are already presented in previous document D5.7 in section 4 and the authors do not duplicate this information here.

1. OpenSWPC

Open-source Seismic Wave Propagation Code (OpenSWPC) is an open-source software that simulates seismic wave propagation by solving motion equations using the Finite Difference Finite Diversity (FDM) Constituent Elastic/Visibility Site Equations (FDM) in the Interface Environment (MPI) in 3D and 2D (P-SV or SH). OpenSWPC is widely used in seismology and has a high portability, allowing for excellent performance from PC clusters to supercomputers. Without modifying the code, users can simulate seismic wave propagation using their own speed structure models and the necessary source representations in the input parameter file. The software code is equipped with a frequency-independent damping model based on a generalised Zener (standard linear solid - SLS model) body and an efficiently selected, perfectly matched boundary absorbing layer. It has different modes for the different input data types of the velocity structure model and different source representations, such as single force, torque tensioner and flat frequency, which can be easily selected by input parameters. Common binary data formats, a common network data form (NetCDF), and a seismic analysis code (SAC) are used to input a heterogeneous structure model and simulation results so that users can easily operate their input and output data sets. All codes are written in Fortran 2003 and are available with detailed documents in a public repository.

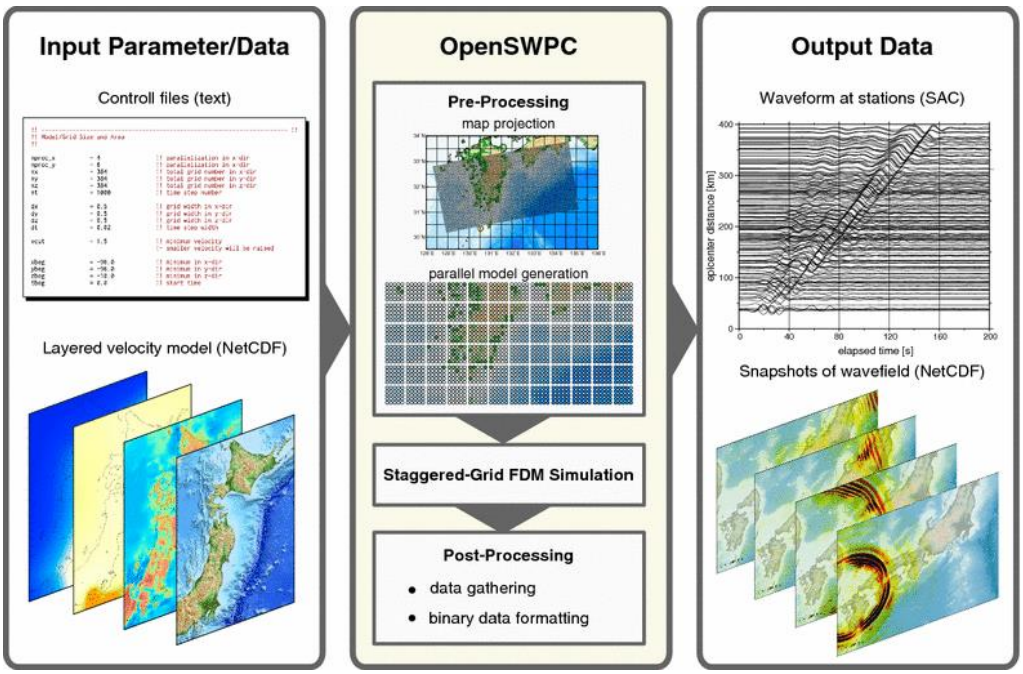

Fig. 1 OpenSWPC in operation[1]

Use case description

The main goal of the benchmarking process was to test the OpenSWPC library against various available architectures and check its behaviour and possible scalability. The first step was to install the required linux packages such as gmt, gmt-dcw, gmt-gshhg and libnetcdf-dev.

Then, it was required to run the `gen_JIVSM.sh` script to generate files used as a model input around Japanese Islands, for seismic wave propagation in and around Japan. They describe 3D complicated subsurface structures of the earth. The above script used the following input files available from public repository at:

```
curl -O http://www.jishin.go.jp/main/chousa/12_choshuki/dat/nankai/lp2012nankai-e_str.zip
curl -O http://www.jishin.go.jp/main/chousa/12_choshuki/dat/nankai/lp2012nankai-w_str.zip
curl    -O    https://www.ngdc.noaa.gov/mgg/global/relief/ETOPO1/data/bedrock/grid_registered/
netcdf/ ETOPO1_Bed_g_gmt4.grd.gz
```

to be installed in `dataset/vmodel` folder.

Finally, inside `OpenSWPC/src` folder the files `shared/makefile.arch` and `shared/makefile-tools.arch` need to be updated to contain valid system paths.

After successful compilation an input file (e.g `example/input.inf`) needs to be provided to run the application. Critical attributes influencing the run times include:

```
nx              = 1000          !! total grid number in x-dir
ny              = 875           !! total grid number in y-dir
nz              = 200           !! total grid number in z-dir
nt              = 100           !! time step number
```

[1] https://earth-planets-space.springeropen.com/articles/10.1186/s40623-017-0687-2

Depending on the hardware and installed libraries authors had the following MPI versions on-board:

- Intel(R) MPI Library for Linux* OS, Version 2017 2017.1.132 (2-node Intel Gold 6140, Eagle cluster)
- Open MPI 1.10.2 (ARM, AMD Epyc™)

2. ABMS

This code is a basic educational example for the interaction of agents and basically tests the performance of the ABM framework including computation, IO, waiting time, and synchronization time.
Two types of entities are created: *location*, as part of a spatial domain, and acting *agents* that follow some decision rule and interact with each other. The agents are distributed uniformly on a 2D area. The spatial proximity increases the likelihood of agents to get connected. Each agent has only one property "A", which is initiated as a random floating point number between 0 and 1 and is connected with other *m* agents.
At each step the agents perform the same action pattern. If the average of all values "A" of all connected agents is less than 0.5, the agents draws a new property "A" for themselves. It is to observe if the global average of "A" is affected by the decision rule and, if so, how. This minimal example therefore contains a computational component per agent (action) and a communication component. The property "A" is synchronized via ghost agents between the processes.
Dependent on whether the focus is on weak or strong scaling, the example setup can adapt to the number of used processes.

Use case description

The application is purely python-based code that requires several modules among which the most important is h5py (Pythonic interface to the HDF5 binary data format) that can be installed as follows:

```
CC="mpicc" HDF5_MPI="ON" HDF5_DIR=$HDF5_DIR HDF5_VERSION=1.10.2 pip install --no-binary=h5py h5py
```

ABMS application additionally requires `class_auxiliary.pyc`, `class_graph.pyc` and `lib_gcfabm.pyc` files containing the application's byte code. Additionally, the python environment has to be equipped with additional modules preinstalled such as *numpy, mpi4py, matplotlib, pandas*, etc.

The application was tested against two cases, one where layer shape is set to size of 64x64 and a second with a size of 128x128. For both cases the possible numbers of MPI processes used were limited to square numbers (1,4,9,16…).

3. CMAQ / CCTM

CMAQ (Community Air Multiscale Quality Modelling System) is an active open-source development project of the U.S. EPA (Environmental Protection Agency) Computational Exposure Division that consists of a suite of programs for conducting air quality model simulations. CMAQ is supported by the CMAS Center: (http://www.cmascenter.org).

CMAQ combines current knowledge in atmospheric science and air quality modelling with multi-processor computing techniques in an open-source framework to deliver fast, technically sound estimates of ozone, particulates, toxics, and acid deposition.

The CMAQ Chemistry-Transport Mode (CCTM) is the only CMAQ program that can be run in parallel.

Benchmark data have been downloaded based on the following information provided by EPA for single day: https://www.epa.gov/cmaq/cmaq-inputs-and-test-case-data.

Use case description

The following outlines a sequence of steps to build the base suite of CMAQ software, including the CMAQ Chemical Transport Model (CCTM):

1. Install NetCDF

```
CC=mpiicc FC=mpiifort ./configure --disable-netcdf-4  --disable-dap
make check
make install
```

2. Install NetCDF Fortran

```
CC=mpiicc      FC=mpiifort      CPPFLAGS="-IPATH_TO_INCLUDES      "LDFLAGS="-
LPATH_TO_LIBRARIES -lnetcdf" ./configure --prefix=PATH_PREFIX
make check
make install
```

3. Install IOAPI

```
wget https://www.cmascenter.org/ioapi/download/ioapi-3.2.tar.gz
tar zxvf ioapi-3.2.tar.gz
cd ioapi-3.2
cp Makefile.nocpl Makefile
export BIN=Linux2_x86_64ifort (example for Intel compiler)
```

4. Build CCTM

```
export CMAQ_HOME=PATH
tcsh -c "./config_cmaq.csh intel"
tcsh -c "source ./bldit_project.csh intel"
cd CMAQ_HOME/CCTM/scripts
tcsh -c "./bldit_cctm.csh intel |& tee bldit.cctm.log"
```

5. Update *run_cct.csh* file in order to set a valid number of mpi processes

4. Cloud Modelling (CM1)

CM1 is a three-dimensional, time-dependent, non-hydrostatic numerical model that has been developed primarily by George Bryan at The Pennsylvania State University (PSU) (circa 2000-2002) and at the National Center for Atmospheric Research (NCAR) (2003-present). CM1 is designed primarily for idealized research, particularly for deep precipitating convection and for studies of relatively small-scale processes in the Earth's atmosphere, such as thunderstorms. (Source: CM1)

CM1 is also designed for distributed-memory computing systems. In CM1 there are three models of parallelization. Shared memory parallelization with OpenMP, distributed memory using MPI and mix of both of them – hybrid OpenMP / MPI.

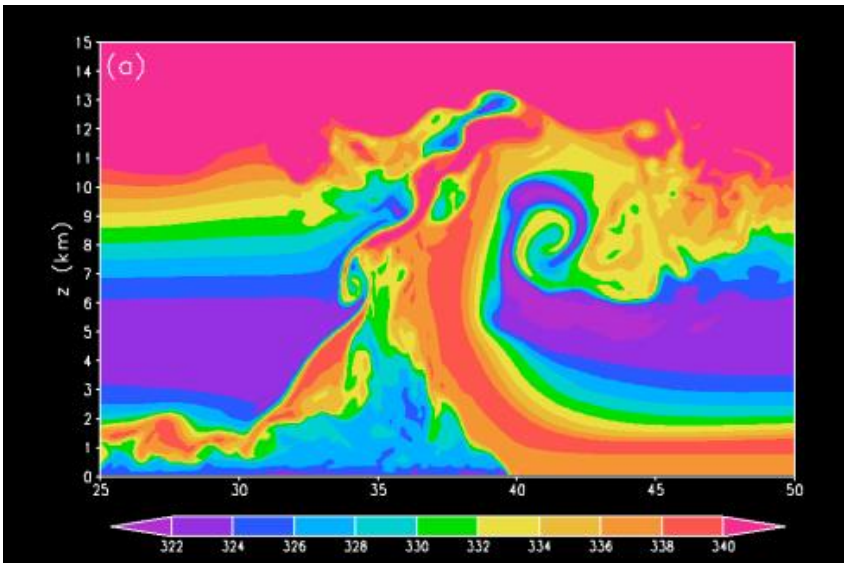

Fig. 2 Visualisation of cloud system based on CM1 model

Use case description

CM1 use case contains the following steps:

1. Download software
2. Compilation
3. Configuration
4. Run

1. Download software CM1 from http://www2.mmm.ucar.edu/people/bryan/cm1/getcode.html and unpack:

```
wget cm1r19.tar.gz
tar zxvf cm1r19.tar.gz
```

2. Compilation
Edit the Makefile and uncomment following lines:

```
vim cm1r19/src/Makefile
```

```
# FC = mpif90
FC = mpiifort
OPTS = -I..include -O3 -xHost -ip -assume byterecl -fp-model precise -ftz -no-fma -nofor-main
CPP  = cpp -C -P -traditional -Wno-invalid-pp-token -ffreestanding
DM   = -DMPI
```

```
make clean
make
```

After successfully built, cm1.exe binary file will be created under cm1r19/run directory.

3. To configure base run, edit following file:

```
vim cm1r19/run/namelist.input
```

Edit following lines:

```
&param0
 nx         =     200,
 ny         =     256,
 nz         =     40,
 nodex      =     2,
 nodey      =     2,
 ppnode     =     16,
 timeformat =     2,
 timestats  =     1,
 terrain_flag = .false.,
 procfiles  = .false.,
 /
```

This example is created to start simulation on nodex * nodey = 4 CPUs on ppnode = 16 processors node. To run simulation on 16 CPUs, change nodex to 4 and nodey to 4 etc.

4. Run
To start simulation simply call mpirun as follows:

```
mpirun -np 4 ./cm1.exe
```

5. Novel Advanced HPC Architectures

1. Intel® Xeon® Gold 6140 (SkyLake SP)

Architecture description

Cores	18 (36 Threads)
L1 cache	64 KB per core
L2 cache	1 MB per core
L3 cache	24.75 MB (shared)
Created	2017
Architecture	Haswell x86
Extensions	x86-64, Intel 64 SSE, SSE2, SSE3, SSSE3, SSE4, SSE4.1, SSE4.2 AVX, AVX2, AVX-512, MPX, TXT, TSX, SGX, VT-x, VT-d
Socket(s)	FCLGA 3647
Energy	Power consumption 85W - 200W (140W Xeon Gold 6140)

The new core for Skylake-X, technically called the Skylake-SP core architecture, delivers some new improvements compared to the previous Broadwell-E platform. One of those "upgrades' has been towards better SIMD performance: clustering multiple data entries into a single element and performing the same operation to each of them at once in one go. This has evolved in many forms, from SSE and SSE2 through AVX and AVX2 and now into AVX-512 (technically AVX-512-F + some others).
Other important changes available in Xeon Gold include:

- Up to 22 processor cores per socket (with options for 4-, 6-, 8-, 10-, 12-, 14- ,16- ,18-, and 22-cores)
- Support for Hexa-channel ECC DDR4 memory speeds up to 2666 MHz
- Direct PCI-Express (generation 3.0) connections between each CPU and peripheral devices such as network adapters, GPUs and coprocessors (48 PCI-E lanes per socket)
- Advanced Vector Extensions (AVX 512):
  - AVX-512 is a set of 512-bit SIMD extensions that allow programs to pack sixteen single-precision eight double-precision floating-point numbers, or eight 64-bit or sixteen 32-bit integers within 512-bit vectors. The extension provides double the computation capabilities of that of AVX/AV2.
  - Vector Length Orthogonality: ability to operate on sub-512 vector sizes
  - 512-bit Byte/Word support
  - Additional D/Q/SP/DP instructions (converts, transcendental support, etc.)
  - Conflict Detect: used in vectorizing loops with potential address conflicts

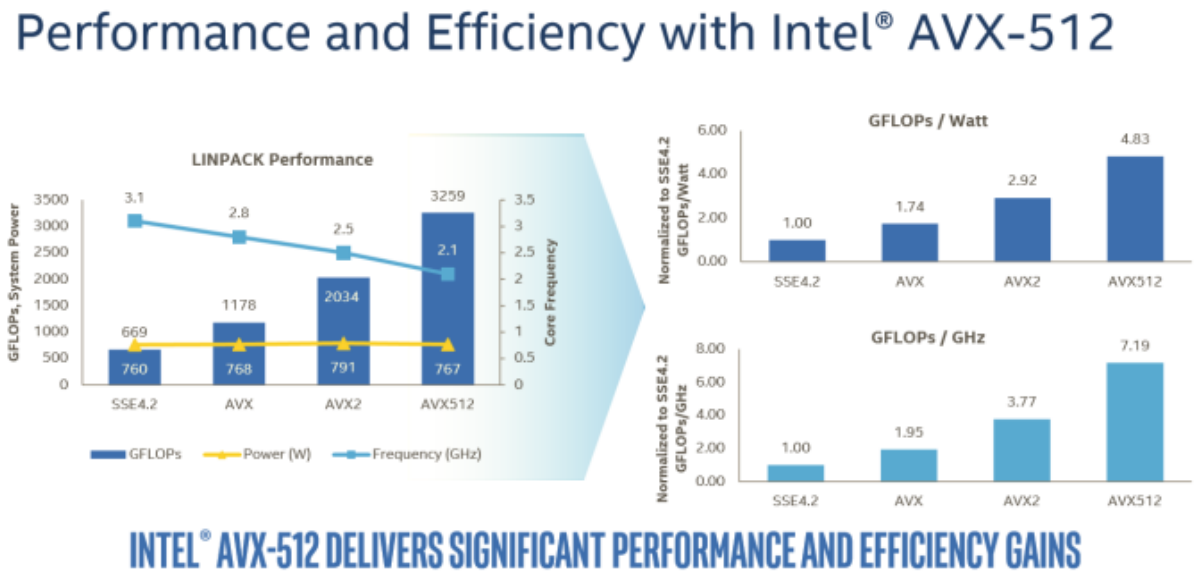

Fig. 3 Performance and efficiency with Intel® AVX-512; source: Intel

In this case, Intel is reporting 60% better performance with AVX-512 versus 256-bit AVX2.

- Mesh architecture (from ring in Broadwell). Sub-NUMA Clustering (SNC) support (replaces the Cluster-on-Die (COD) implementation)
- µOP Cache
  - instruction window is now 64 Bytes (from 32)
  - 1.5x bandwidth (6 µOPs/cycle, up from 4)
- Execution Engine
  - Larger re-order buffer (224 entries, up from 192)
  - Larger scheduler (97 entries, up from 64)
  - Larger Integer Register File (180 entries, up from 168)

Cluster (node) configuration

Number of nodes	2
CPUs per board	2
RAM	192 GB
Interconnect description	10Gb Ethernet
I/O and disks	SSD
OS version	Ubuntu 16.04.03 LTS
Programming environment, compilers, libs etc	Intel Parallel Studio 2017 (Intel MPI, Intel icc compiler, Intel Fortran compiler, Intel MKL)

2. AMD Epyc™ 7551

Architecture description

Cores	32 (64 Threads)
L1 cache	64 KB 4-way + 32 KB 8-way per core
L2 cache	512 KB per core
L3 cache	64 MB (shared)
Created	2017
Architecture	Zen x86
Extensions	AMD64/x86-64, MMX(+), SSE1, SSE2, SSE3, SSSE3, SSE4a, SSE4.1, SSE4.2, AES, CLMUL, AVX, AVX2, FMA3, CVT16/F16C, ABM, BMI1, BMI2, SHA
Socket(s)	SP3
Energy	Power consumption 180W

AMD Epyc™ processor is based on the Zen microarchitecture and is manufactured on a 14 nm process. This microarchitecture was designed from the ground up with data centres in mind, for optimal balance and power. This new core design can process four x86 assembler instructions per cycle and also introduces Simultaneous Multithreading (SMT). Another improvement that Zen has had over Bulldozer (the previous microarchitecture) are the instruction set. The part of them are exclusive for AMD:

- ADX – extending multi-precision arithmetic support
- RDSEED – complement to RDRAND random number generation
- SMAP – Supervisor Mode Access Prevention
- SHA1/SHA256 – Secure Hash Implementation Instructions
- CLFLUSHOPT – CLFLUSH ordered by SFENCE
- CLZERO – Clear Cache Line
- PTE Coalescing – Combines 4K page into 32K page size

Zen microarchitecture also introduces considerable amount of improvements and design changes over Bulldozer:

- Utilizes 14 nm process (from 28 nm)
- Simultaneous Multithreading (SMT) support
- Improved branch mispredictions
  - Better branch predictions with 2 branches per BTB entry
  - Lower miss latency penalty
- Large Op cache (2K instructions)
- Wider µop dispatch (6, up from 4)
- Larger instruction scheduler
  - Integer (84, up from 48)
  - Floating Point (96, up from 60)

- Larger retire throughput (8, up from 4)
- Larger Retire Queue (192, up from 128)
- Larger Load Queue (72, up from 44)
- Larger Store Queue (44, up from 32)
- Cache system:
  - L1
    - 64 KiB (double from previous capacity of 32 KiB)
    - Write-back L1 cache eviction policy (From write-through)
    - 2x the bandwidth
  - L2
    - 2x the bandwidth
  - Faster L2 cache
  - Faster L3 cache
  - Large Op cache
  - Better L1$ and L2$ data prefetcher
  - 5x L3 bandwidth
  - Move elimination block added
  - Page Table Entry (PTE) Coalescing

Cluster (node) configuration

Number of nodes	1
CPUs per board	2
RAM	512 GB
Interconnect description	10Gb Ethernet
I/O and disks	SSD
OS version	CentOS 6.9
Programming environment, compilers, libs etc	Intel Parallel Studio 2017 (Intel MPI, Intel icc compiler, Intel Fortran compiler, Intel MKL)

3. ARM Hi1616

Architecture description

Cores	32 ARM® Cortex®-A72 processor cores
L1 cache	48KB I-cache, 32KB D-cache per core
L2 cache	1MB per cluster (4 cores), 16MB in total(8MB per chip)
L3 cache	16MB per supercluster (4 neighbour clusters – 16 cores), 32 MB in total
Created	2017
Architecture	ARMv8-A 64-bit

Extensions	fp asimd evtstrm aes pmull sha1 sha2 crc32
Socket(s)	N/A
Energy	70W (per chip)

The HiSilicon Hi1616 V100 products are based on ARM Cortex-A72 cores. These are high-performance, low-power processors based on the ARMv8-A architectural platform and the Hi16xx Family supports all features of the ARMv8-A architectural platform.

The Cortex-A72 processor is a high-performance processor that implements the Armv8-A architecture.

Cortex-A72 can be paired with Cortex-A53 in big.LITTLE configuration for mobile applications. The Cortex-A72 processor cluster has one to four cores, each with L1 instructions and data cache, together with a single unified L2 cache.

Main benefits

- Arm's state-of-the-art high-performance processor for the infrastructure, mobile and automotive industries.
- Market leading computational density in all coefficients of application forms.
- Enhanced performance and efficiency, with full 64-bit support for the Armv8-A.

This processor can also be implemented in Arm big.LITTLE configuration.

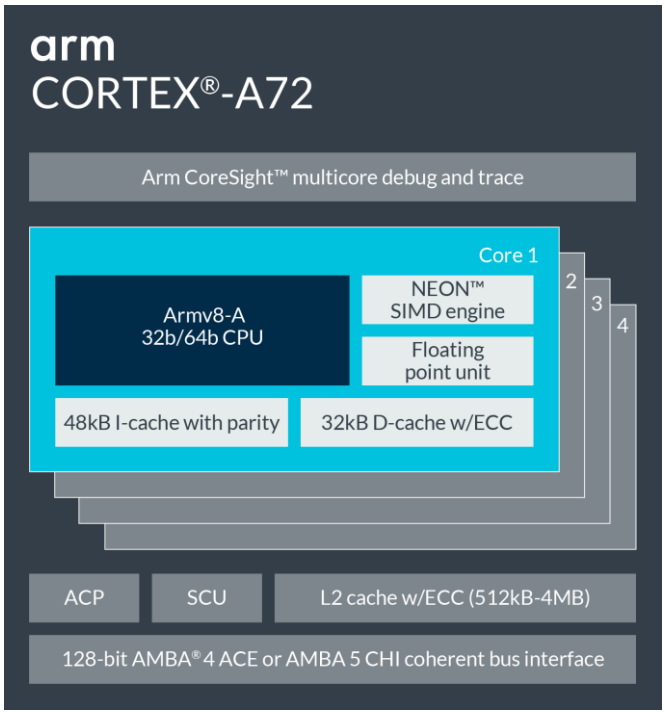

Fig. 4 ARM A-72 architecture

The high-performance Cortex-A72 processor is designed for a wide range of high-performance applications, combined with the benefits of an energy-efficient arm architecture.

Increasing shipping and computational throughput over Cortex-A57 maximized the performance of the 3-level off-level bus to eliminate code dependency for high peak and sustained instruction throughput at frequencies above 3GHz in 16FF+ process technology.

The new, sophisticated algorithm dramatically increases the accuracy of predictions, which reduces energy wastage during the execution of erroneous code paths.

Every aspect of the Cortex-A57 microarchitecture has been optimized to introduce a new level of Cortex-A energy efficiency with radical improvements in all aspects of the PPA measurement (performance, power and area).

Support for network applications and storage with full ECC cache and 44-bit address up to 16TB.

Cortex-A72 delivers 3.5 times the performance of smartphones compared to Cortex-A15 28nm in 2014. The processor features several major microarchitectural improvements that build on the current generation of Armv8-A cores. Improvements in floating point, integer, and memory performance improve the performance of each major load class.

The processor is optimized for 16nm FinFET technology, enabling Cortex-A72 to clock up to 2.5GHz in a mobile power envelope for even greater overall performance.

In addition to this key performance improvement, Cortex-A72 also benefits from significantly lower power consumption. Enhanced performance combined with 16nm FinFET technology enables the Cortex-A72 to achieve a 75% power reduction in representative, premium mobile workloads.

The Cortex-A72 processor can be built into a SoC system using a wide range of technologies including graphical IPs, system IPs, and physical IPs. The Cortex-A72 processor is fully supported by ARM programming tools.

Cluster (node) configuration

Number of nodes	2	
CPUs per board	2	
RAM	128GB	
Interconnect description	Ethernet 10Gb/s	
I/O and disks	SSD	
OS version	EulerOS release 2.0 (SP2)	Ubuntu 16.04.3 LTS
Programming environment, compilers, libs etc	OpenMPI 1.10.2 gmt-5.4.3 netcdf-c 4.5.0 netcdf-fortran 4.4.4	OpenMPI 1.10.2 gmt-5.4.3 netcdf-c 4.4.0 netcdf-fortran 4.4.3

4. Power8+ S822LC

Architecture description

Although the IBM 8335 Power System S822LC for High Performance Computing server Model GTB (8335-GTB) was released in September 2016, it still remains one of the most interesting hardware platforms for HPC available at the market. This platform delivers breakthrough accelerated computing performance. Being designed for "accelerated workloads in high-performance computing (HPC), high-performance data analytics (HPDA), enterprise data centers, and accelerated cloud deployments" (2), it perfectly suits for all kinds of CoeGSS applications including preprocessing (e.g., creating synthetic populations and synthetic networks), simulation, HPDA, and visualization.

S822LC brings together two POWER8 CPUs with two or four -- HLRS testbed exploits four -- NVIDIA Tesla P100 GPUs through POWER8 with NVLink Technology. NVLink is used to transfer data to GPUs, while actual instructions are still transferred to accelerators via PCIe. Thus, the user can benefit from NVLink, when changes from POWER8 without GPUs (S822L) to POWER8+ (S822LC), only if the target application uses modern features of NVIDIA GPUs.

Cluster (node) configuration

Number of nodes	1
Type/Model of node	IBM Power System S822LC (8335-GTB)
CPU Model name	10-core 2.860 GHz (3.857 GHz turbo) POWER8NVL Processor, altivec supported
Threads per core	8
Cores per socket	10
Sockets	2
NUMA nodes	2
CPU max MHz	4023.0000
CPU min MHz	2061.0000
L1d cache	64K
L1i cache	32K
L2 cache	512K
L3 cache	8192K
Type of accelerators	GPU
Accelerator model	Tesla P100-SXM2-16GB
Accelerator connection	NVLink
RAM	512G (16x32G RDIMMs) DDR4 1600 MHz
I/O and disks	Lustre
Bus Type	PCIe
Created	2016/09/26
OS version	Ubuntu 16.04.2 LTS
Programming environment	GCC 5.4, OpenMPI 1.10.2, Python 2.7.13.1, parallel HDF5 1.10.2, serial HDF5 1.8.16, NetCDF 4.4.0, NetCDF C++ 4.2.1, NetCDFF, OpenBLAS 0.2.18, BLAS 2.6.0, LAPACK 3.6.0, ScaLAPACK 1.8.0, Boost 1.58, GDAL 1.11.3
TDP	190W

6. Tests performed

Similarly to D5.7, benchmarks done for this document purposes used */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 again worth noting that these metrics are reported only by the process that is under control of */usr/bin/time*.

Comparing to the previous document where two versions of tests were provided – with and without cache clearing – in this document the authors decided to focus entirely on benchmarks where data are always read from disk and no from cache.

Benchmarks results have been obtained for all application and available processors combinations to be able to observe the real behaviour of the selected applications. Each test has been repeated twice with the measurements for minimum wall time selected to be reported. Detailed results are presented in the figures within the Annex. The most interesting findings and conclusions from these tests are presented below. It is worth to notice that some figures reveal unusual and difficult-to-explain behaviour in memory consumption and/or I/O, which cannot be anticipated based on processor architecture or node configuration. These cases are supposed to be a subject of future analysis by applications authors or GSS community.

In general, the characteristic of the applications is found to be different and varying from I/O dependant to purely computational.

Pandora-based GG pilot application – delivered by CoeGSS project community – shows for its both input data collections (EU and WW maps) that the total elapsed time drops rapidly when the number of MPI processes is in the range between 1 and the maximum number of physical cores for a given processor architecture. After exceeding this number execution time continues to drop but not as rapidly as before. As an example, figures Fig. 22 and Fig. 27 represent results for benchmarks on AMD Epyc™ 7551 single node. After exceeding the number of 32 cores execution times continues to drop, but not as rapidly as before. After exceeding at 64 MPI processes (maximum number of physical cores on tested node) and reaching value of 128, elapsed time falls to about 80% of the value for 64 cores and about 59% of the value for 32 cores. This shows that using hyper threading with 2 processes per one physical core does make sense and brings further advantages for this GG-pilot application and looks like the only exception among all presented processors.

Benchmarks for both input data collections demonstrate a strong correlation between elapsed time and RAM usage for Pandora. In particular, the intervals of the rapid decrease in elapsed time match to the intervals of rapid reduction in RAM consumption. For the large number of

processes, the slight increase in elapsed time is related to the overheads one of which is a slight increase in the total output. Please note that these conclusions are consistent with exemplary results reported in D5.7 (e.g., for IBM Power8 node).

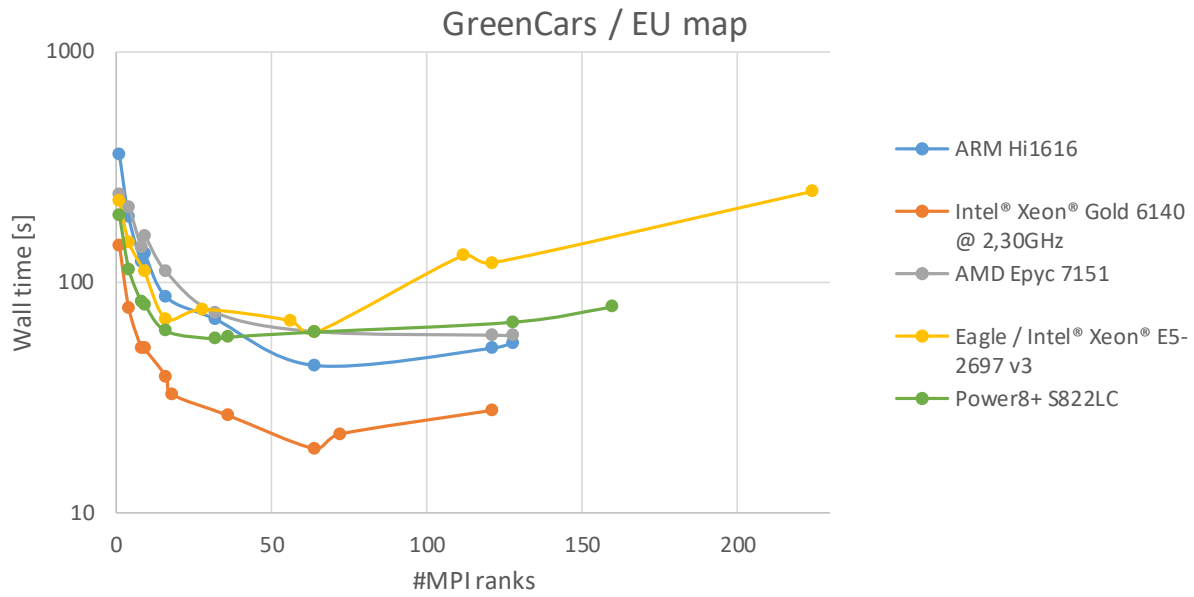

Fig. 5 Pandora scalability results for EU map across all processors

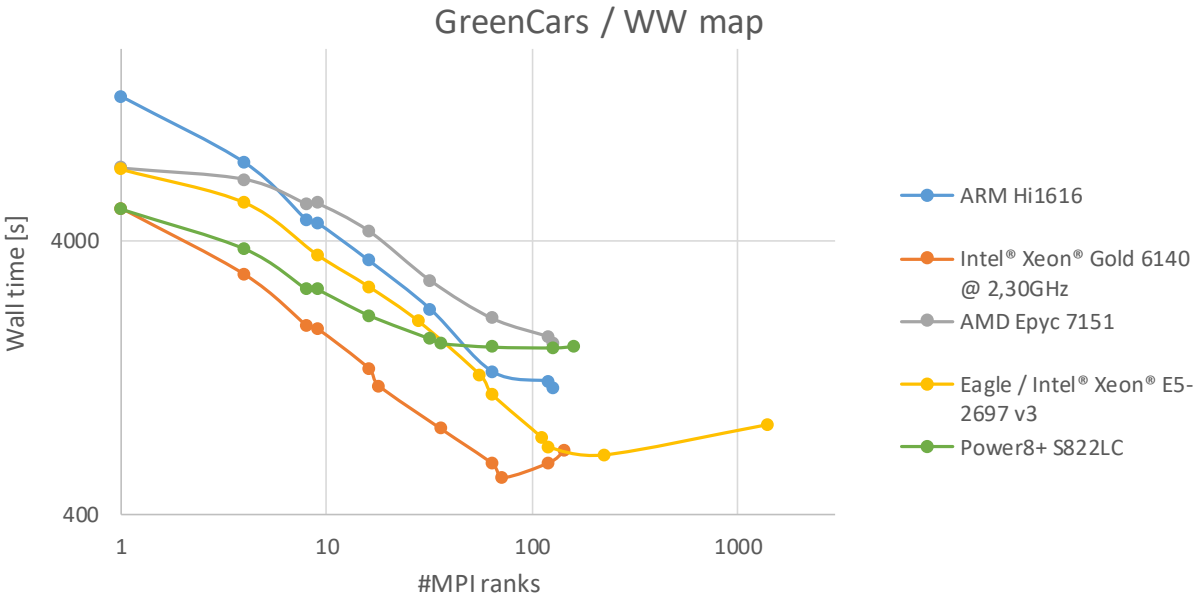

Fig. 6 Pandora scalability results for WW map across all processors

Detailed GG-pilot figures can be found in the Annex (Fig. 20–Fig. 29).

OpenSWPC benchmark for given input configuration reports good scalability. It is especially illuminative in case of the reference testbed where the execution time decreases along number

of cores used until maximum number of 1400 is used. Other processors show similar behaviour. Using hyper threads (where possible) does not provide any further time improvements.

As an example, Power8+ processor reaches minimum 102.15 secs for 64 MPI processes with about x17 speedup. After passing this minimum, the elapsed time remains almost insensitive to the growth of processes number. The fastest rate of decrease in the elapsed time is observed for the number of processes between 1 and 16 (from 1748.65sec to 120.89sec). On this interval the efficiency is higher than 91%. The input fluctuates slightly around 50MB without any obvious dependency on the number of processes. RAM consumption steadily drops with the number of processes reaching its minimum 128.5MB at 128 MPI processes – the largest number of processes used in benchmark on this architecture. The rate of decrease in RAM consumption is particularly high for the small number of processes and slight afterwards. In particular, it decreases more than x15 on the interval between 1 and 16 processes from 43.3GB for a single process run to 2.8GB for 16 MPI processes. Similar to Pandora benchmarks, we observe a strong correlation between elapsed time and RAM usage.

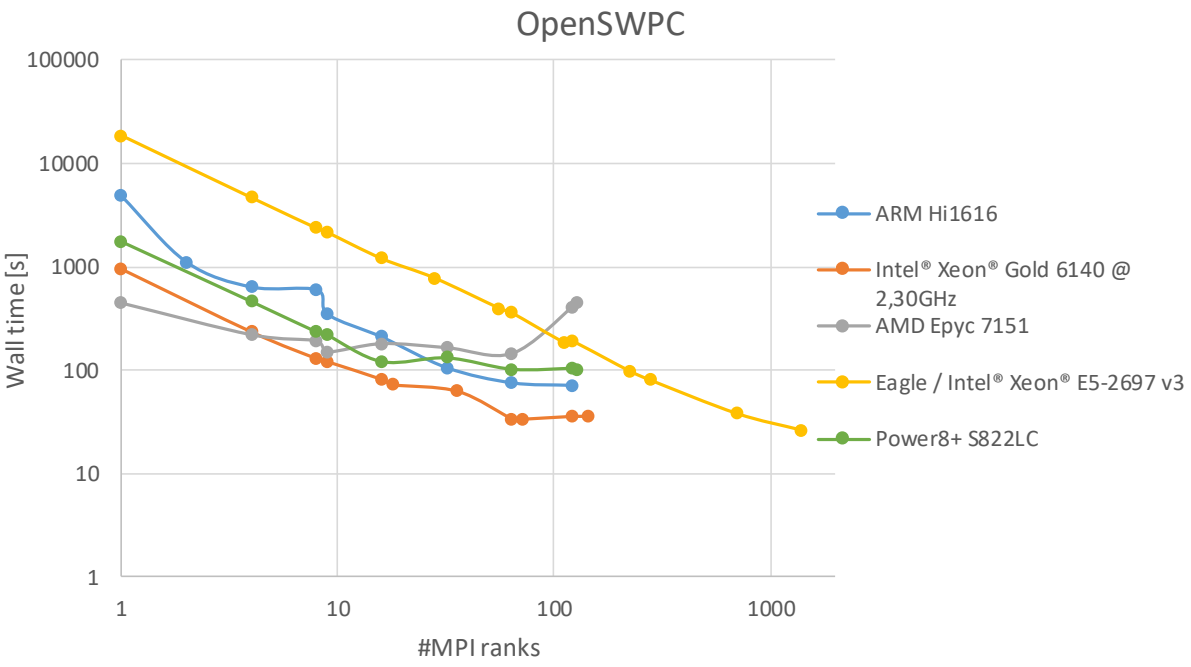

Fig. 7 OpenSWPC results for EU map across all processors

Detailed OpenSWPC figures can be found in the Annex between Fig. 30 and Fig. 34.

Among all tested applications **IPF** is the least I/O demanding reading only small input text file and presenting the output in simple text form only. The reference testbed (cluster) shows scalability up to 700 cores. On the other hand, other selected processors scale in the range of the number of physical cores so we expect that using them in a multinode configuration will result in similar behaviour as for the reference cluster. All processors show rapid decrease in the elapsed time between single process run and 16/32 MPI processes. After that a further

slow decrease in the total wall time can be observed until the curve reaches a large plateau and starts to grow slowly. As mentioned above the total I/O is relatively small and does not influence the overall elapsed time crucially.

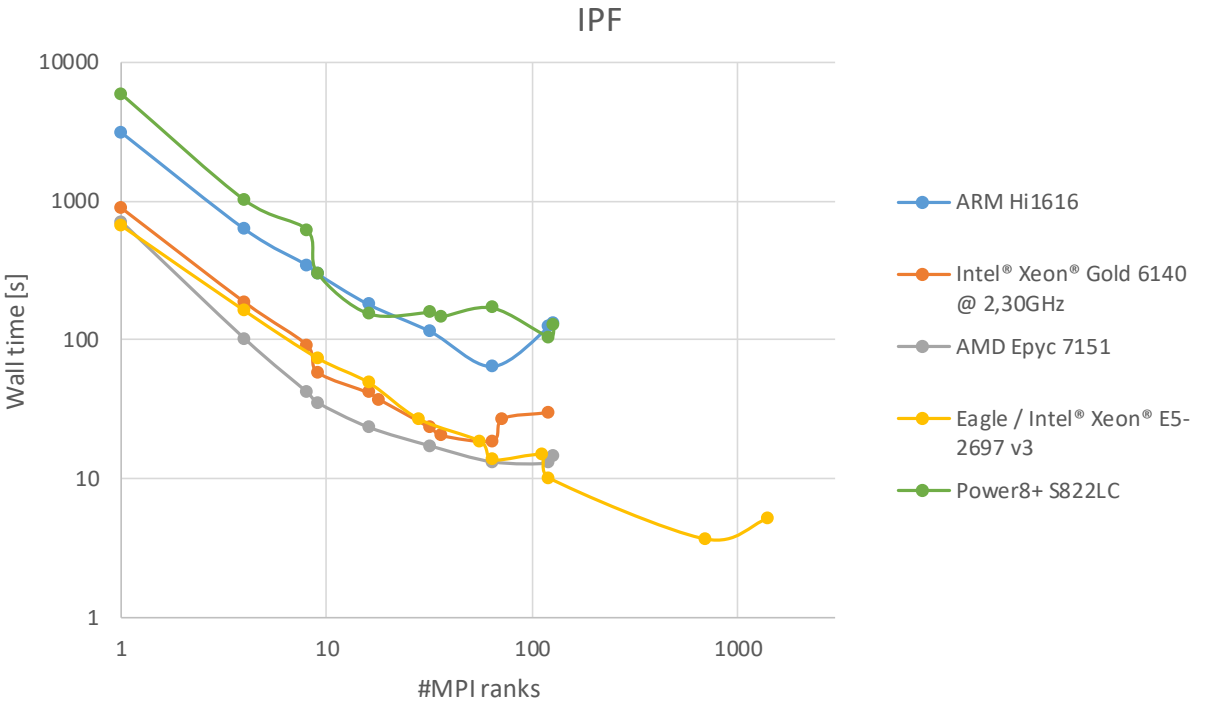

Fig. 8 IPF scalability results for WW map across all processors

Detailed IPF figures can be found in the Annex between Fig. 35 and Fig. 39.

Results obtained for ABMS – another application delivered by CoeGSS project community – prove it should be subject to major improvements. For instance, the best execution time on Intel® Gold® 6140 is observed for only 9 MPI processes, whereas the testbed includes 72 physical cores (2-node configuration with 2 chips each, 18 cores per one chip). The reference testbed and Power8+ scalability using 64x64 layer are limited already by 16 MPI processes. Increasing the size of the shape to 128x128 (potentially to increase compute intensity) does not lead to noticeably improved results. Other testbeds perform slightly better: ARM and AMD Epyc™ produce the lowest wall time for 64 MPI processes. Nevertheless, switching hype threading on and increasing number of HT cores does not bring any further improvements even for the latter architectures. Most probably, the poor scalability cannot be explained by insufficient parallel utilization of resources only and may be explained by some algorithmic causes inside the application when it comes to a higher number of processes.

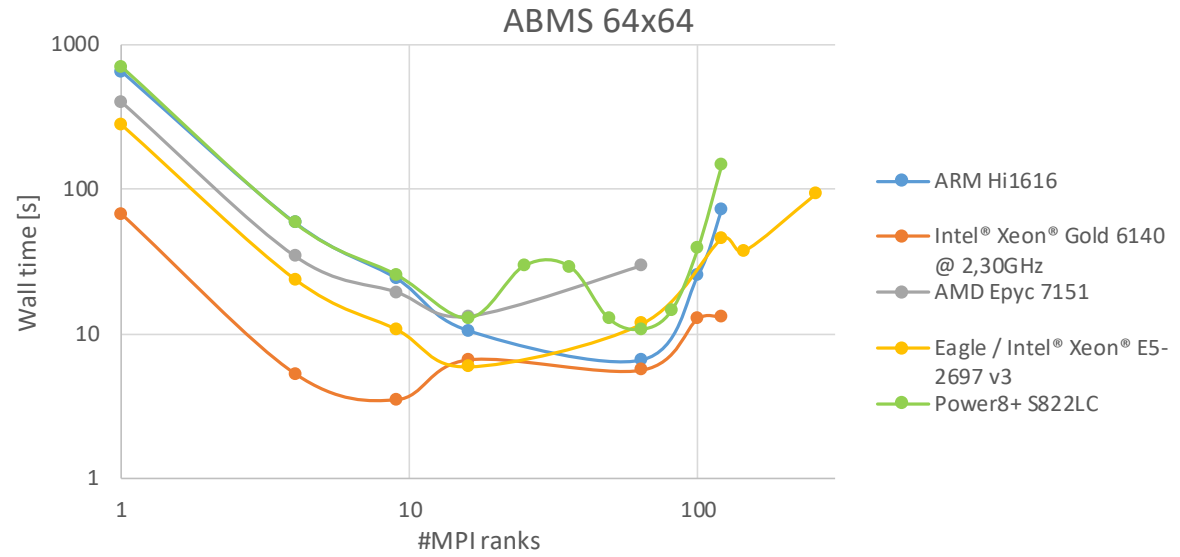

Fig. 9 ABMS scalability results for layer shape 64x64 across all processors

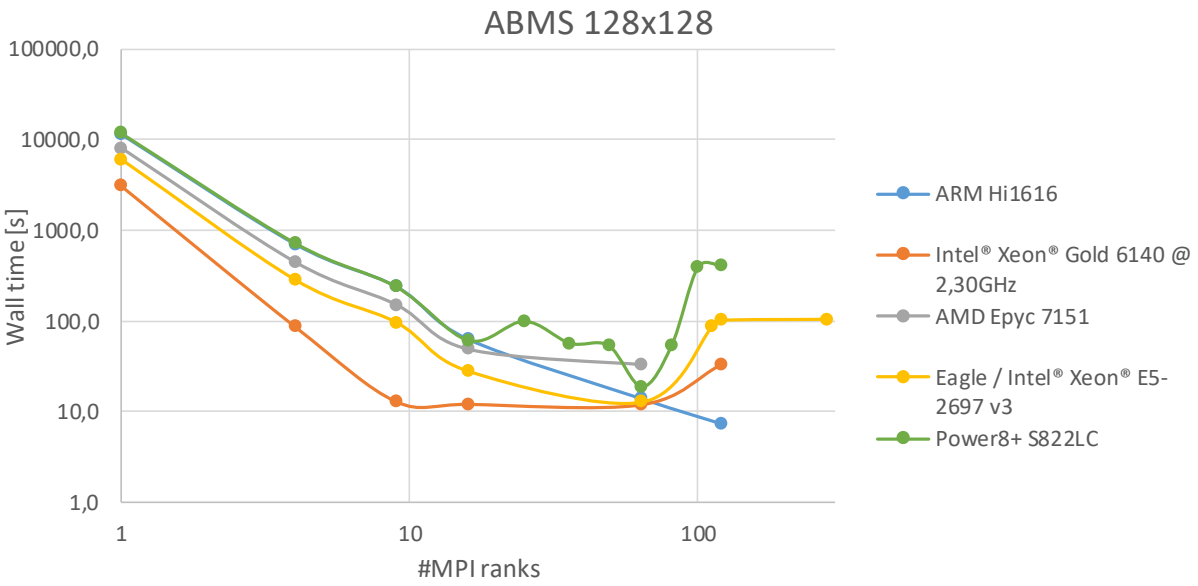

Fig. 10 ABMS scalability results for layer shape 128x128 across all processors

Detailed ABMS figures can be found in the Annex between Fig. 40 and Fig. 49.

CCTM is an example of I/O dominated application and, thus, the impact on the execution times is much higher than for example in IPF case. The reference testbed shows the application scales up to 280 cores for the given input data collection. Other presented processors despite high I/O demand present a similar behaviour as for other applications: high execution time drops to approx. 16 cores, then again a slow decline up to physical number of cores and finally a downturn.

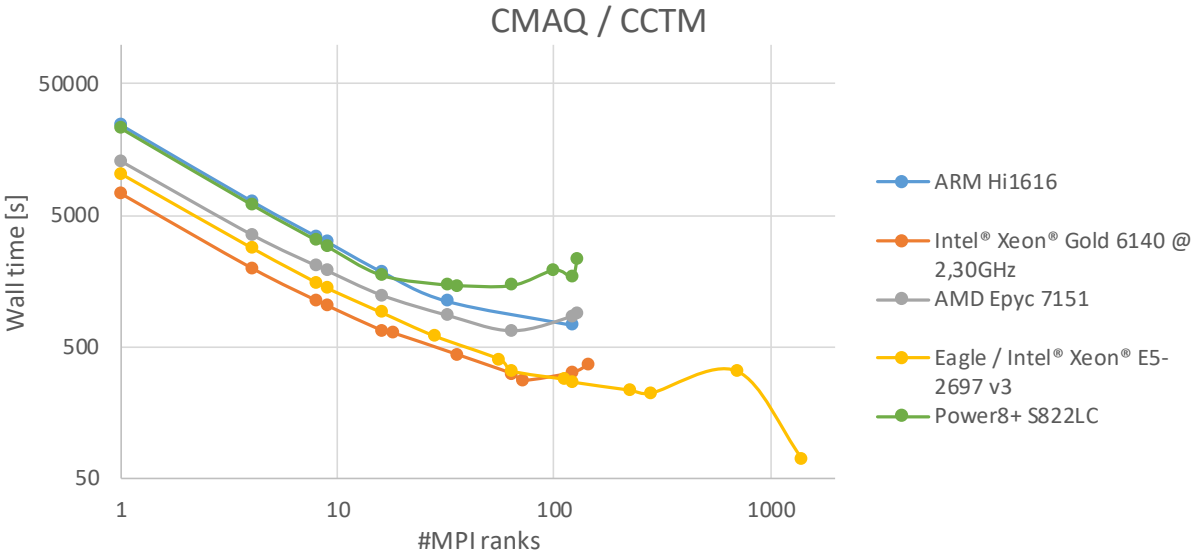

Fig. 11 CMAQ/CCTM scalability results across all processors

Detailed CCTM figures can be found in the Annex between Fig. 50 and Fig. 54.

CM1 results indicate a relationship between the problem size (weather map grid) and the processors mesh, as well as the dependence between the number of threads and nesting of cores in different levels of cache. For example, a graph Fig. 55 shows the results for Cloud Model CM1 launched on HiSilicon Hi1616. From single thread to eight of MPI ranks the application is almost linear scalable – for single thread the execution time took around 11397 sec, but for 8 cores decreased by almost 8 times. This behaviour may be due to the fact that one SCCL (Super CPU Cluster) share L3 cache between four groups of CCL (CPU Cluster) which is composed of four CPU cores. The best speed-up is of course noticeable when software runs on 4 cores in one CCL which share L2 cache. From 16 cores to 64 cores, the execution time falls a little softer, which is caused by SCCL to SCCL latency on Hydra Interface bus, but still shows good and cost-effective scalability. The use of memory is also divided into each process (from ~1,1GB on single thread to 570MB for 2 threads, 320MB for 4 threads etc.) but with a small overhead, probably caused by additional buffers for MPI. The remaining graphs show similar relationships up to the point where the calculations begin to go beyond the physical cores. Fig. 57 presents the results for AMD Epyc™ processor. After 64 cores line is passed, there is significant limit of scalability deterioration, caused by Hyper Threading feature which is not effective for this type of application, thus execution time on 128 cores fall to 181,5 seconds comparable to 181,9 seconds on 32 cores. In the case of Intel® Xeon® Gold (Fig. 56), the results are similar - enabling Hyper Threading feature does not cause such a large deterioration of results between 64 and 128 threads, but does not bring any positive results. In the case of the Results for Power8+ testbed (Fig. 58) also confirm that Hyper Threading function is useless in case of the CM1 application. After reaching a larger number of threads than there are physical cores (i.e., 20 in case of Power8+), the application stops scaling.

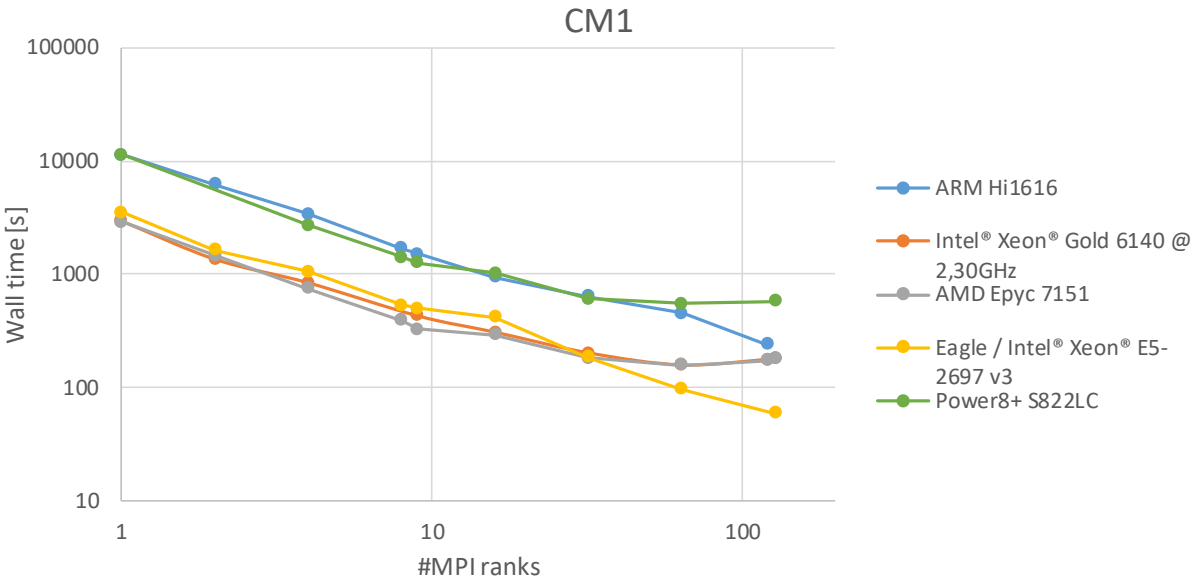

Fig. 12 CM1 scalability results across all processors

Very similar results in terms of scalability on processors equipped with the implementation of simultaneously multithreading have been demonstrated by the **Hurricane WRF** weather model benchmark. After reaching the ending point of the number of physical cores, the execution time begins to slowly increase, which definitely indicates the pipeline overload or filling in registers and waiting for the release of the main processor execution units. The exception is the example presented on Fig. 62, which shows execution time of the HWRF model on a computing cluster equipped with processors based on the older Intel Haswell architecture, but with disabled Hyper Threading feature. In this case, the execution time is still decreasing, and subsequent threads run on the next physical cores of the processors.

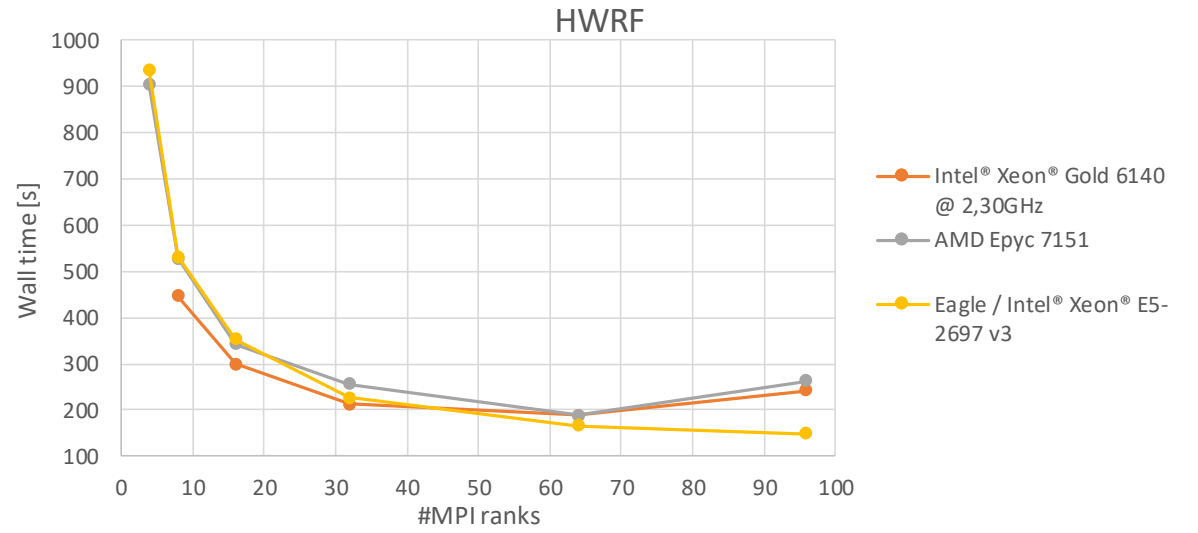

Fig. 13 HWRF scalability results across all processors

7. Final conclusions

The deliverable D5.8 continued the design of benchmark that includes applications used inside the GSS community and compares these applications across the new available HPC architectures. It is again – taking into account the end user requirements which are limited to application efficiency – measured by speed and scalability. Moreover, authors decided to introduce two additional metrics: the efficiency of running benchmarks in relation to energy consumption noted by thermal design power (TDP) and cost efficiency.

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

- OpenSWPC using `swpc_3d.x` application with grid size 1000 x 875 x 200 and MPI partitioning relevant to actual tested MPI ranks.
- IPF application with input file input_40k_3200M and test iterations equals to 5. ROWBLOCK and COLBLOCK parameters were set to 4
- GG (Green Growth) application using two maps with different resolution: European map (EU map) with the size of 640x680 and worldwide map (World map) with the size 8640x3432 processes
- CMAQ/CCTM using CMAQv5.2_Benchmark_SingleDay_Input_09_12_2017.tar.gz and CMAQv5.2_Benchmark_SingleDay_Output_09_12_2017.tar.gz datasets downloaded from CMAQ project website
- ABMS with two test cases using layer shape of 64x64 and 128x128
- CM1 with nx = 200 grid points in the x direction, ny = 256 grid points in the y directions and nz = 40 grid points in the z direction
- Hurricane Weather Research and Forecasting (HWRF) model with historical dataset of Hurricane Katrina based on three days 28.08.2005 – 30.08.2015, interval_seconds = 10800, grid dx = 0.193384, dy = 0.191231

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

- Intel® Xeon® Gold 6140 - 2-node cluster, 2 processors each,
- ARM HiSilicon Hi1616 - 2-node cluster, 2 processors each,
- AMD Epyc™ 7551 - single node with 2 processors,
- IBM Power8+ S822LC - single node with 2 processors,
- and Eagle cluster with 2 Intel® Xeon® E5-2697 v3 processor nodes as a reference configuration,

with one exception – Hurricane WRF could not be installed on the aarch64 (ARM) an Power8+ nodes. Authors of the tests did not succeed to compile HWRF there.

In addition, the authors compared the scalability of particular applications in the context of execution time, memory usage, and I/O operations on different architectures. which allowed to formulate the following observations:

- GG-pilot applications are dominated by I/O operations (mostly output) where a large HDF5 file is created and to which all processes save data;

- CCTM is also heavily I/O dependent;
- Best execution times are usually achieved (when considering single nodes) for the number of processes equal to the number of cores: 64 for 2-node ARM Hi1616 and 1-node AMD Epyc™, 72 for 2-node Intel Xeon Gold 6140, 20 for 2-processor 1-node Power8+;
- In most cases hyper threading does not bring any performance improvements.

As already mentioned, the test results has been used for introducing two additional metrics for ranking of the tested architectures:

- Energy efficiency calculated as a product of walltimes and TDP products which scales and binds the achieved timing results by processors by the theoretical heat generated during the tests and/or the energy consumed by processors.
- Cost efficiency using scaled timing results by the cores price falling on the given number of cores (cores price is calculated by dividing processor price by given number of processor cores).

Purposeful distinction of the above metrics may provide a valuable data for possible GSS cluster owners and users. In particular, they may use it to answer many questions at the time of collection or investments planning, such as which processor architecture suits better or is cheaper for use with GSS applications. They may also anticipate and estimate the cost of the energy used by the system and, thus, provide a balance between system performance and its cost. Of course, presented results are purely based on the data provided by processors vendors (TDPs, costs) and do not include other important factors and obvious costs related to HPC infrastructure components.

The following part presents final results measured as collective timing, energy and cost efficiency comparisons across all tested applications and architectures:

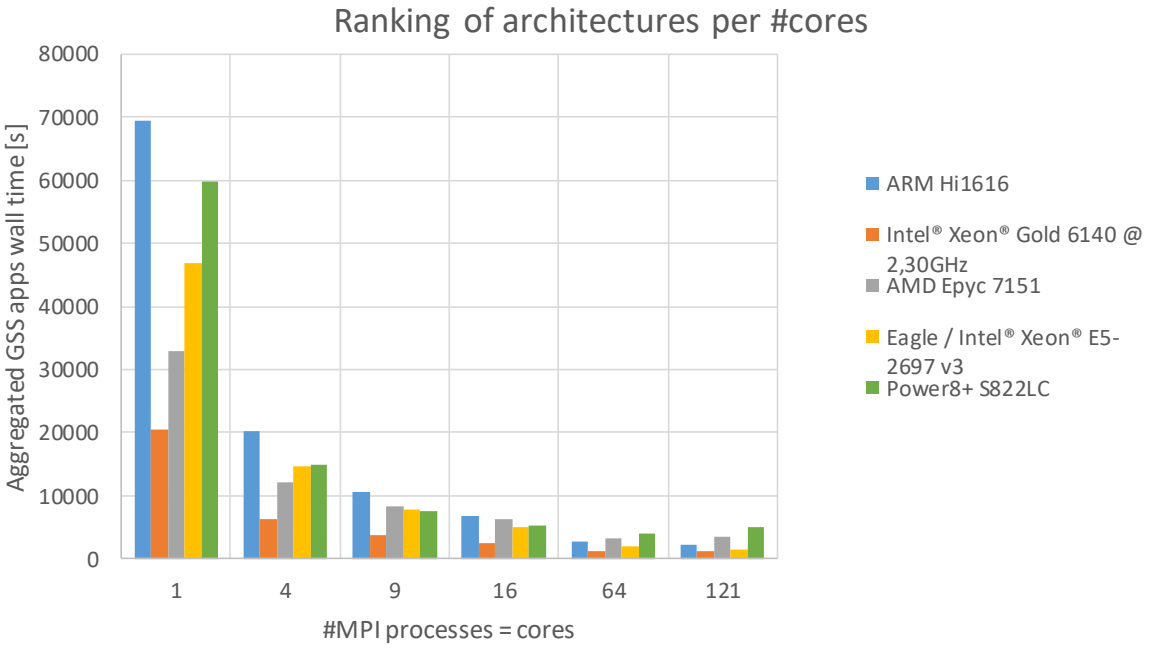

Fig. 14 Ranking of architectures across number of cores (processes)

The chart above (Fig. 14) shows that in the whole range of the number of cores the winner is Intel® Gold® 6140. Surprisingly, the AMD Epyc™ is slower than the ARM Hi1616 when 64 or more cores are used and also is slower than the reference Intel® Xeon® E5 when 9 or more cores are used. The reason is that applications may not have been optimally compiled for the Epyc processor. In case of non-Intel processors, GCC/OpenMPI and other open source libraries were used. It is reasonable to note that even though IBM Power8+ has rather complicated architecture, it beats only ARM Hi1616 in the aggregated elapsed time of the GSS benchmarks.

The following chart (Fig. 15) presents aggregated execution times across all tested applications. Not surprisingly the winner turned out to be Intel® Xeon® Gold 6140 and on the other end was three times slower than ARM Hi1616, mostly due to low clock frequency.

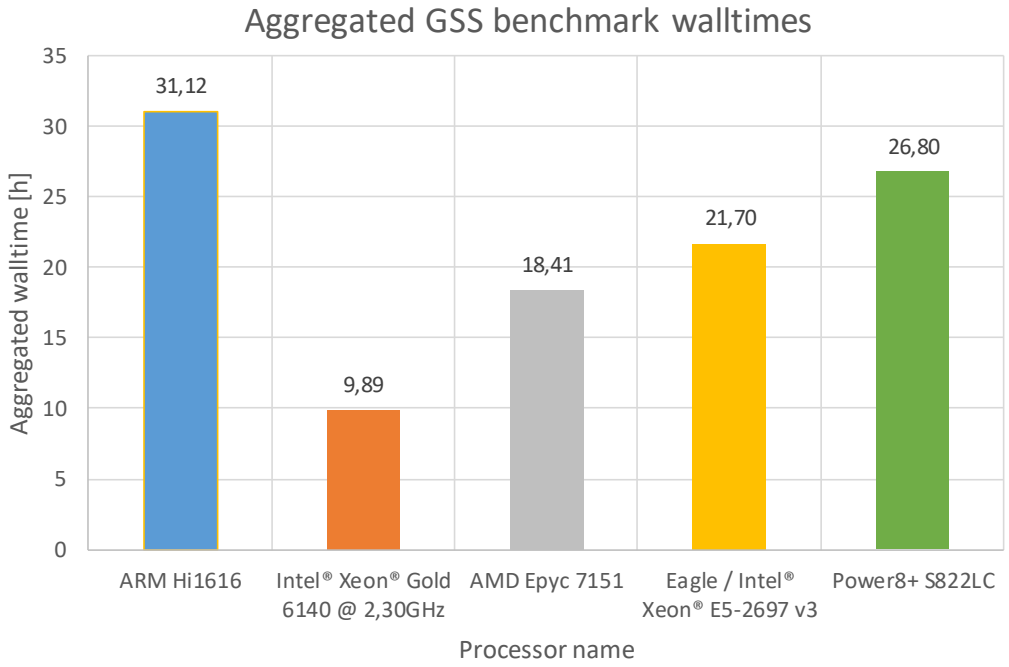

Fig. 15 Ranking of architectures based on GSS aggregated walltimes

The following chart (Fig. 16) presents estimated cumulative energy consumption calculated as a sum of walltimes and TDP per used cores products expressed in kilowatt-hours. Obtained results prove that the winning processor ARM Hi1616 represents the current tendency in HPC, where currently attention is turned to – generally speaking – energy efficient technologies. Second place winner is Intel® Gold® and Power8+ brings up the rear. Power8+ consumes remarkably more energy than the other architectures in the overall benchmark. The latter fact may be explained if we take into account GPU accelerators embedded into Power8+ node: while 4 Nvidia GPUs are integral units of the Power8+ node which consume a significant amount of energy, they are not used by the benchmarked application. It suggests that more fair results may be obtained from comparison with Power8 which has the architecture similar to Power8+, but excludes accelerators unused by benchmarked applications.

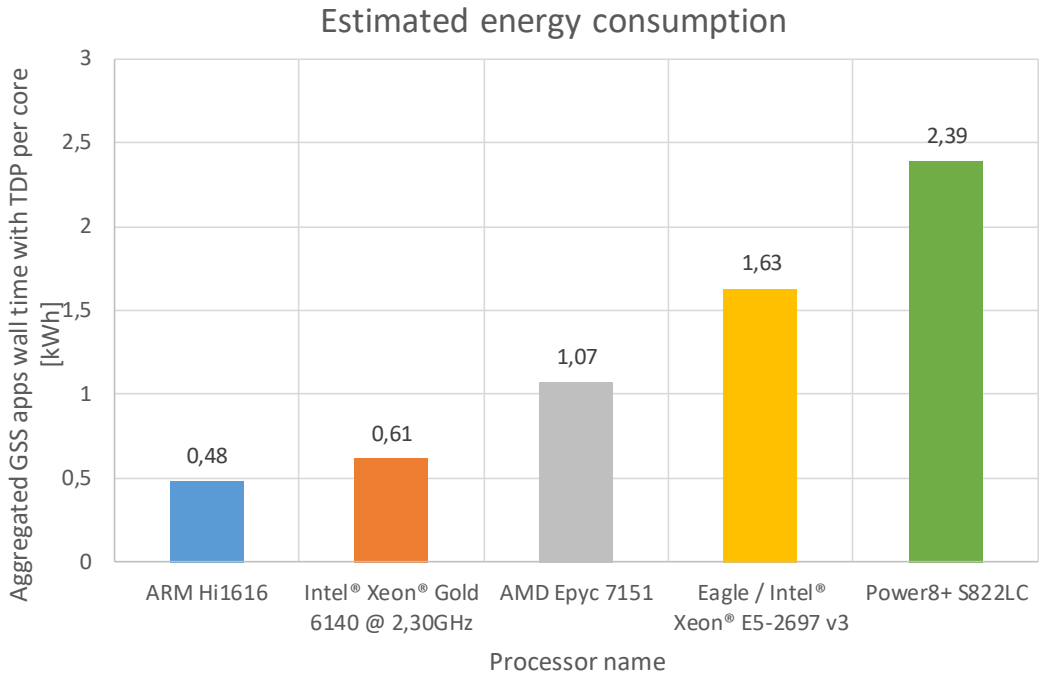

Fig. 16 Ranking of architectures based on estimated energy consumption (total power needed by CPU to finish all tests - lower is better)

The estimated power consumption chart is a good point of view when talking about green HPC computing. Of course, presented results are not exact, because they are dictated only by the estimated values based on the CPU's TDP. Nevertheless, assuming that all architectures use the same memory model - DDR4, it can be considered that most of the energy consumed is the energy utilized by the processor. It should also be noted that the Hyper Threading technology, the implementation of simultaneous multithreading, is just an extension of the pipe, but not the duplication of the main execution units. Therefore, it can be assumed that Hyper Threading cores will be treated as additional functionality of physical cores, and TDP will be divided by their number as the primary energy per core indicator. In the above summary, the best energy consumption ratio is characterized by the ARM architecture, which is absolutely designed for energy-saving solutions, which is also widely used in mobile devices. The results for the new Intel Skylake architecture were a big surprise, which took the second place with a very similar result of about 20% more. The AMD Epyc presented a much worse result, but from our observations, its internal architecture is better suited to applications in which I / O systems play an important role.

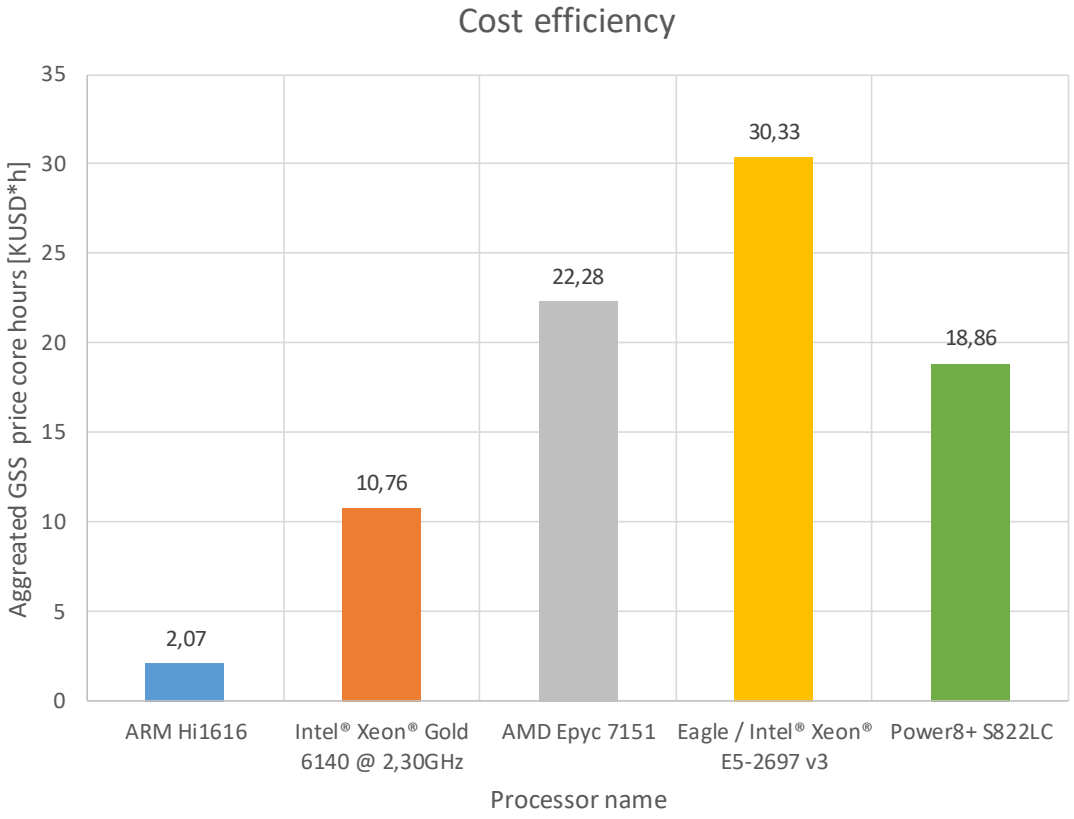

Fig. 17 Ranking of architectures based on cost efficiency

The chart above (Fig. 17) presents aggregated – what is called here – GSS cost efficiency. ARM Hi1616 is approximately 9 times better than for Power8+ (the lower value – the lower cost) and 15 times better than reference Intel® Xeon® E5 processor, mostly due to its little number of expensive cores and relatively middling timing results.

Additionally, when talking about general processors characteristics extracted from all the tests performed, IBM Power8+ demonstrates particularly good performance for the applications with a big number of I/O such as Pandora, OpenSWPC, CMAQ/CCTM. The best results are obtained when the total output dominates over the input and RAM consumption. In many cases, it outperforms ARM Hi1616, Intel Xeon E5, and AMD Epyc™ for I/O intensive applications. On the other hand, Power8+ shows worse results than the above-mentioned processors if the applications are computationally extensive while producing relatively small amount of output. This is the case the IPF and ABMS applications. On all processors, all benchmarks show the highest efficiency if the number of MPI processes is between 2 and the total number of physical cores. After that, the efficiency usually drops remarkably as HT is not utilized properly. At the same time, quite often the highest speedup is reached when the number of MPI processes is significantly more than 20. It would be interesting to perform the tests on the testbeds including more nodes.

In addition, the authors observed an unusually high super linear speedup for ABMS application if the number of MPI processes is between 1 and 4 which suggests that this application requires a more detailed analysis.

All the above aggregated charts, when analysed simultaneously, prove that the most promising ARM processor in the context of cost and energy consumption is the slowest at the same time (mostly due to low clock frequency). Other competitor, Intel® Gold®, is 5 times less cost efficient for GSS benchmark and slightly worse in the energy consumption but it is approximately 3 time quicker regarding the aggregated execution time. In other words, the future HPC investors having the above information in place have the ability to decide which direction to follow: reach high compute intensity, minimize costs, or try to find the golden mean.

The following table presents the spots taken by each individual architecture in three separate domains: walltime, energy efficiency and cost efficiency. In general, the most promising processor is Intel Xeon Gold 6140 but for individuals for whom the most important are costs and environmental aspects, should look closely at ARM Hi1616.

Table 1 Ranking of all tested architectures *(1-means the best result)*

	Walltime	Energy efficiency	Cost efficiency
ARM Hi1616	5	1	1
Intel® Xeon® Gold 6140	1	2	2
AMD Epyc 7151	2	3	4
Intel® Xeon® E5-2697	3	4	5
Power8+ S822LC	4	5	3

Finally, authors separately present the ranking of architectures for HWRF application (Fig. 18 and Fig. 19), which could not be installed on both Power8+ and ARM. What is interesting AMD Epyc™ 7551 turned out to perform slightly better than Intel Xeon® Gold® 6140 when comparing to GSS benchmark.

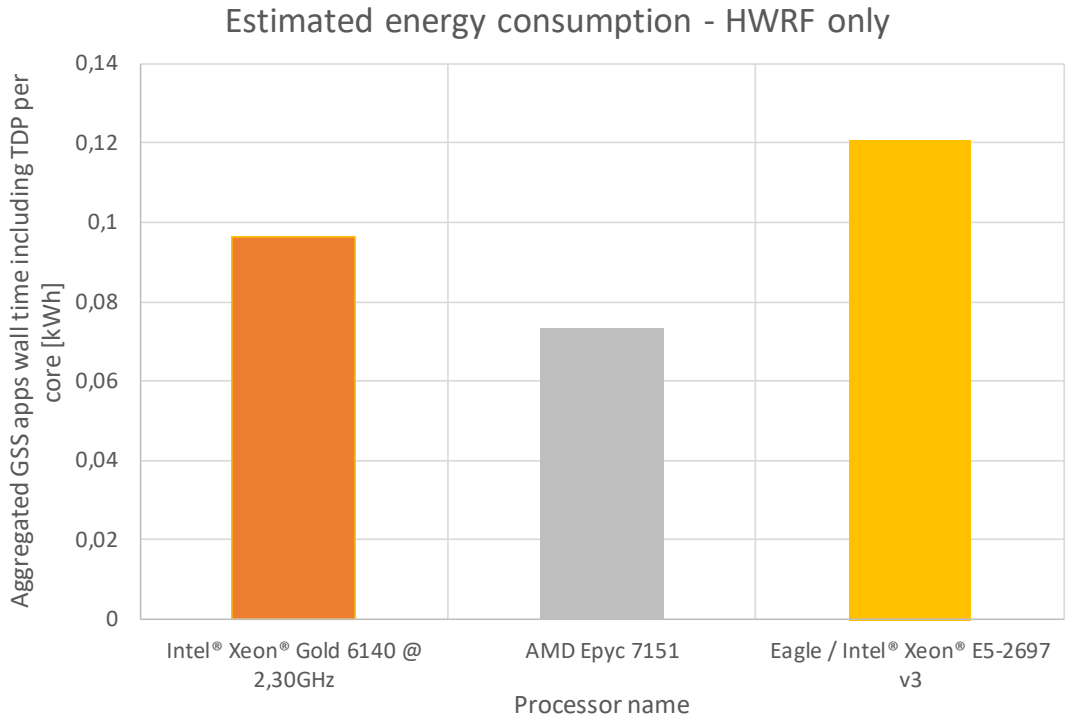

Fig. 18 Ranking of architectures based on estimated energy consumption – HWRF only

Fig. 19 Ranking of architectures based on cost efficiency – HWRF only

8. References

1. ***Unix tsch shell - https://en.wikipedia.org/wiki/Tcsh.*** **[Online]**

2. Family 8335+03 IBM Power System S822LC for High Performance Computing. [Online] https://ibm.co/2cMMb8B.

3. Global System Science. [Online] http://global-systems-science.eu/gss/content/introduction-gss.

4. benchmarks, Computing. https://en.wikipedia.org/wiki/Benchmark_(computing). [Online]

5. BAPCO. https://bapco.com/. [Online]

6. (SPEC), Standard Performance Evaluation Corporation. https://www.spec.org/benchmarks.html. [Online]

7. Performance, Transaction Processing. http://www.tpc.org/. [Online]

8. 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]

9. DEISA Benchmark Suite (2010) http://www.deisa.eu/science/benchmarking. [Online]

10. [Online] https://shop.amd.com/en-us/business/processors/PS755PBDAFWOF.

11. [Online] https://ark.intel.com/products/120485/Intel-Xeon-Gold-6140-Processor-24_75M-Cache-2_30-GHz.

12. [Online] https://ark.intel.com/products/81059/Intel-Xeon-Processor-E5-2697-v3-35M-Cache-2_60-GHz.

9. Annex

Please note some charts present execution times using logarithmic scale to improve the readability.

1. Green Growth using Pandora library

EU map

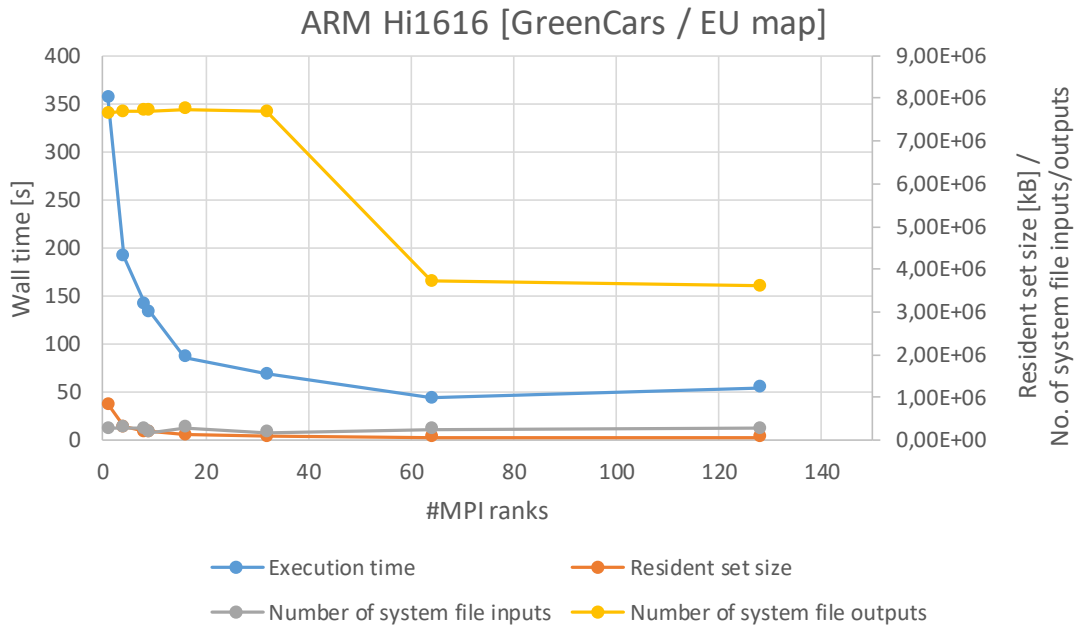

Fig. 20 Pandora results for EU map for ARM Hi1616

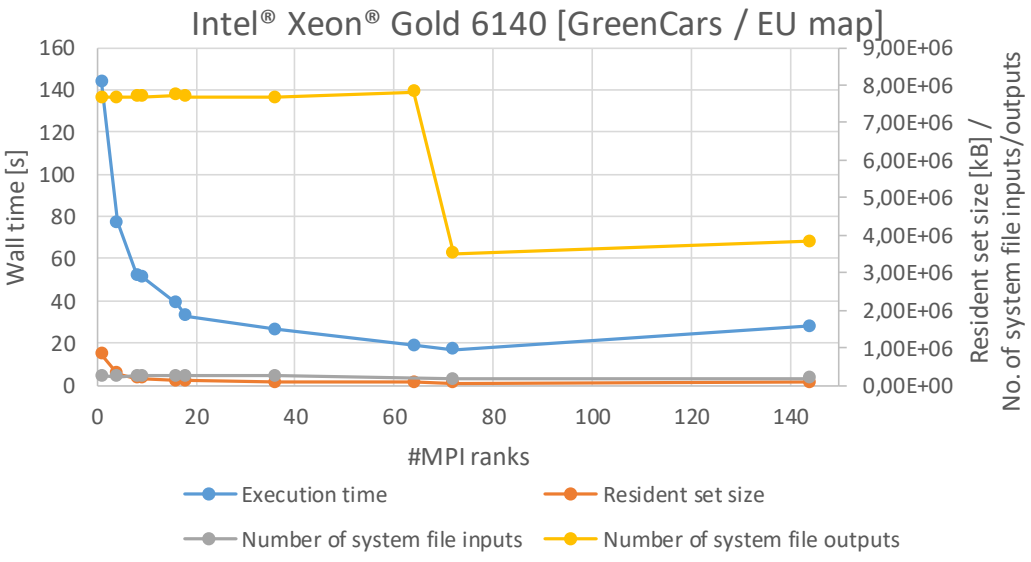

Fig. 21 Pandora results for EU map for Intel® Xeon® Gold 6140

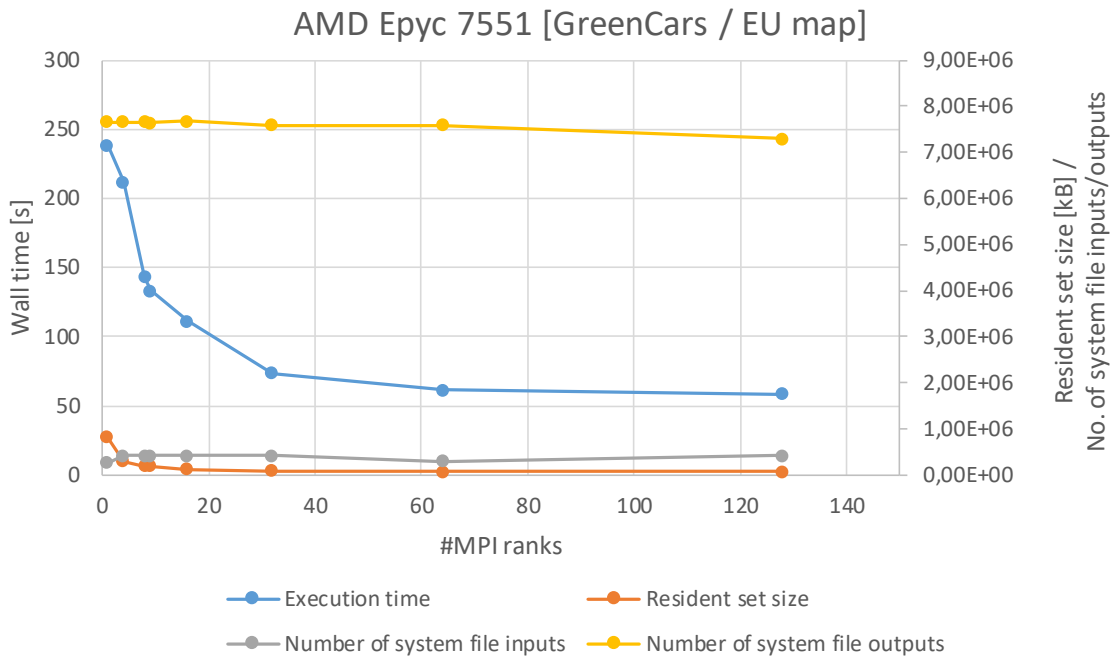

Fig. 22 Pandora results for EU map AMD Epyc™ 7551

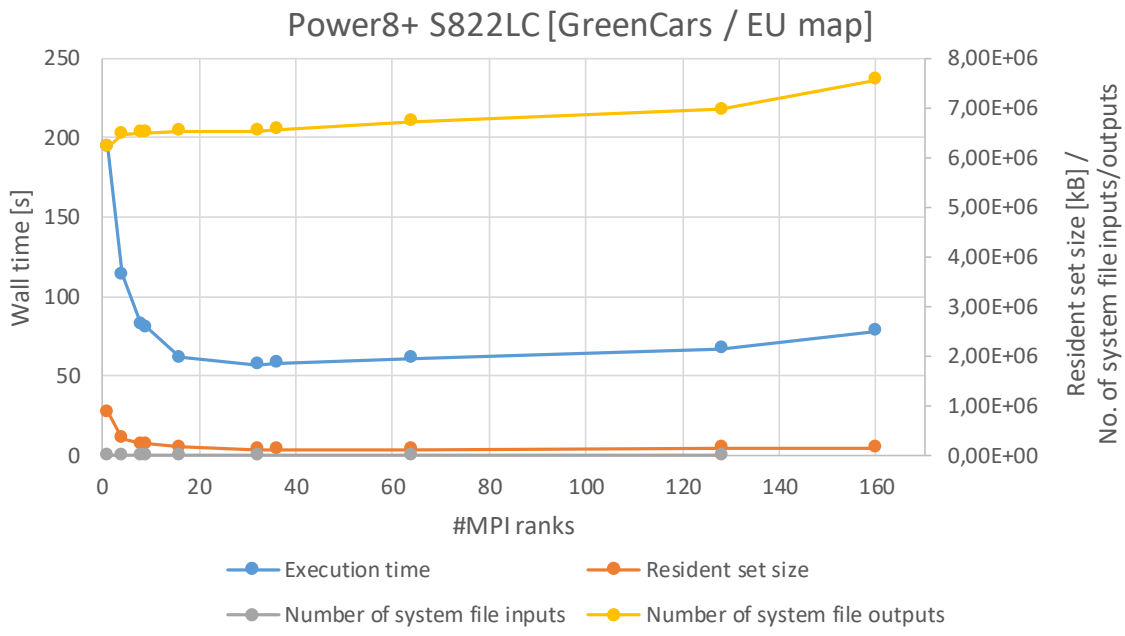

Fig. 23 Pandora results for EU map for IBM Power8+ S822LC

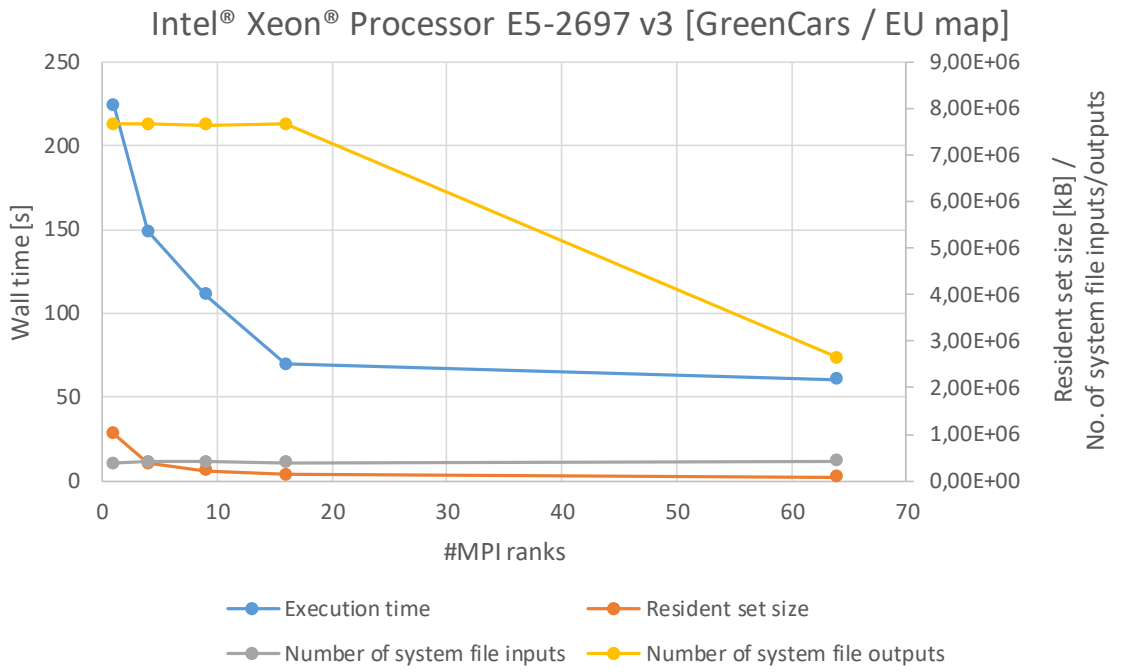

Fig. 24 Pandora results for EU map for Intel® Xeon® E5-2697 v3

WW map

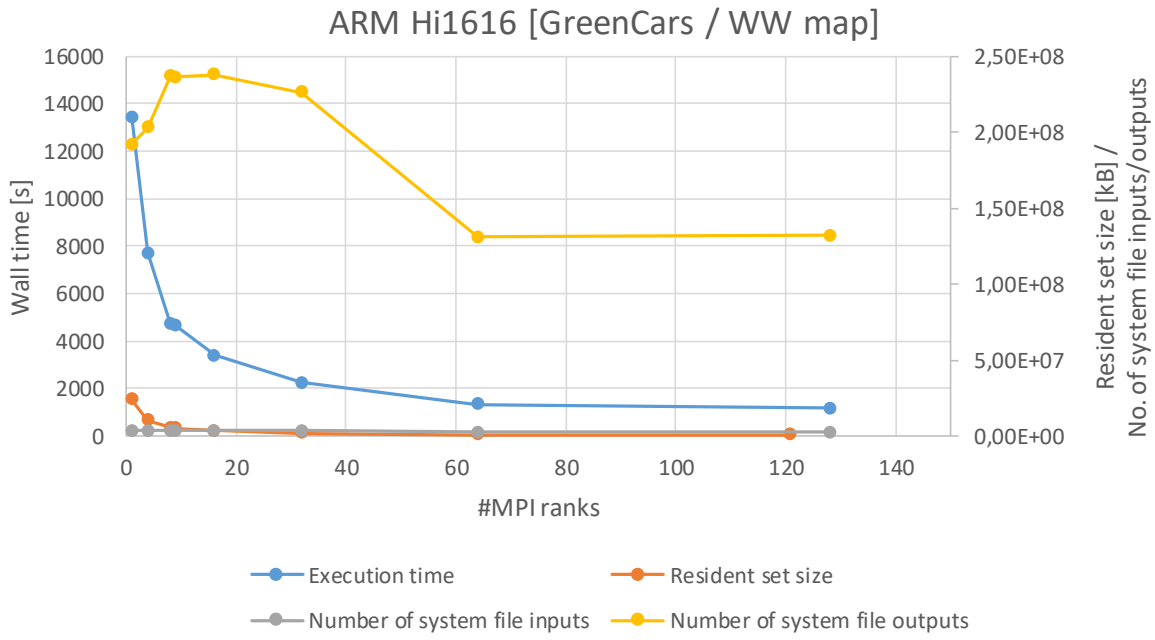

Fig. 25 Pandora results for World map for ARM Hi1616

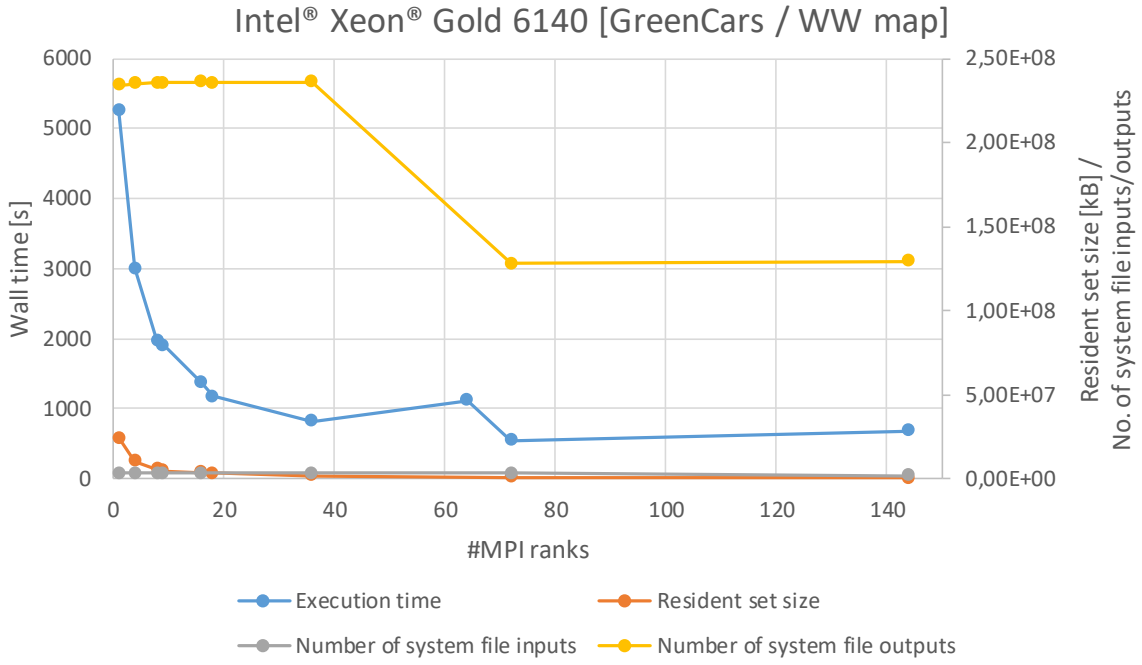

Fig. 26 Pandora results for World map for Intel® Xeon® Gold 6140

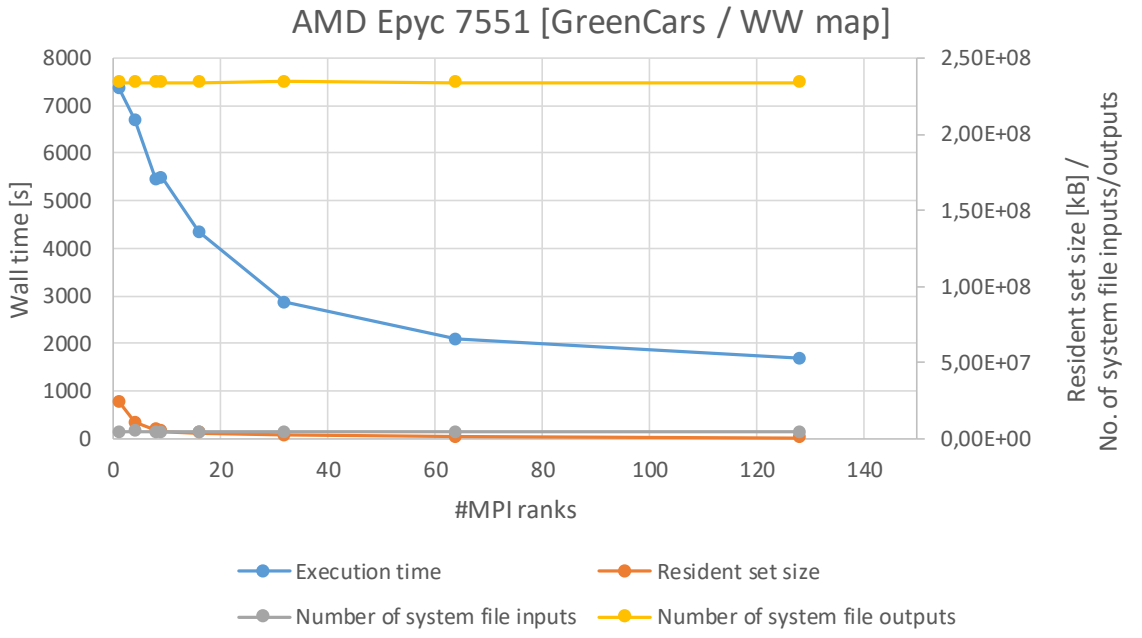

Fig. 27 Pandora results for World map AMD Epyc™ 7551

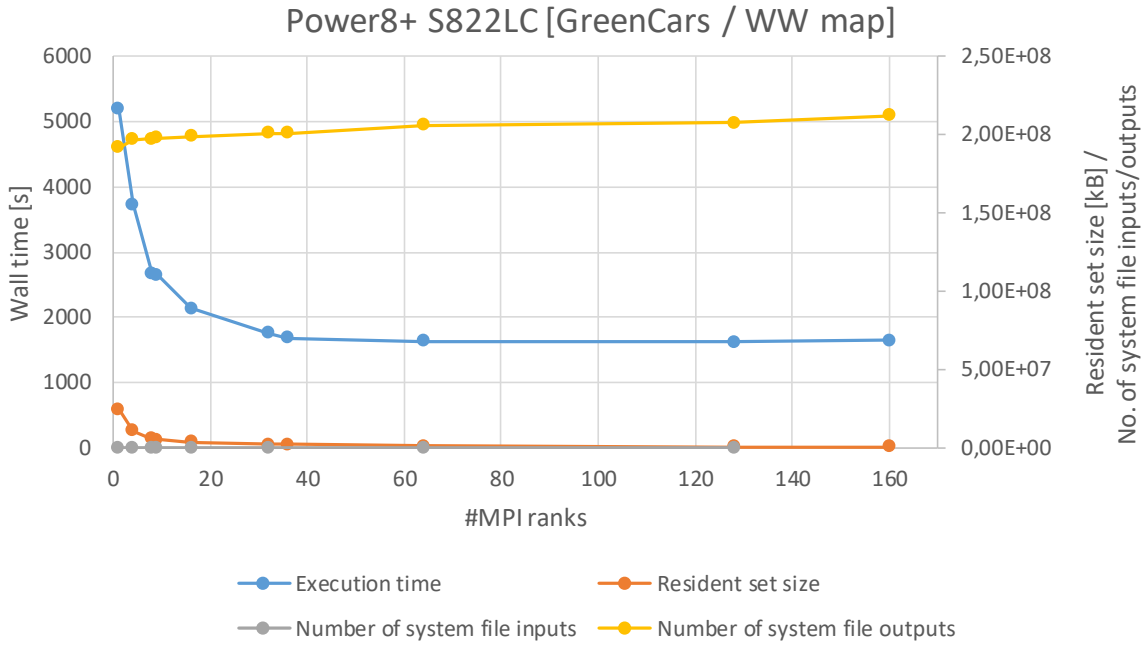

Fig. 28 Pandora results for World map for IBM Power8+ S822LC

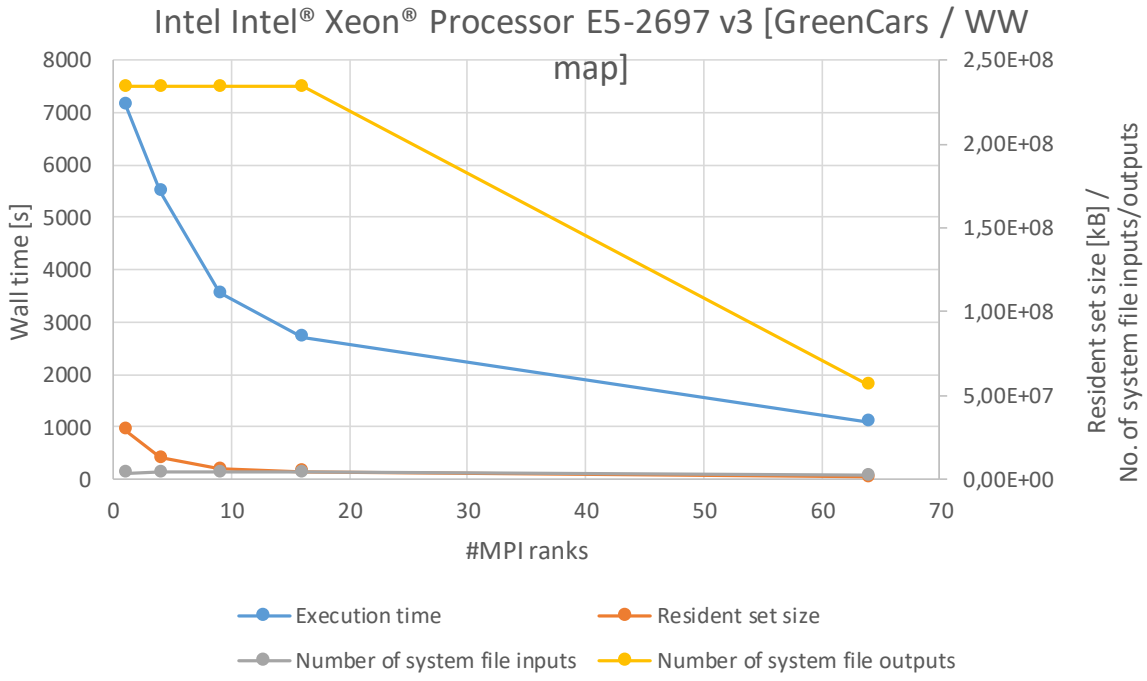

Fig. 29 Pandora results for World map for Intel® Xeon® E5-2697 v3

2. OpenSWPC

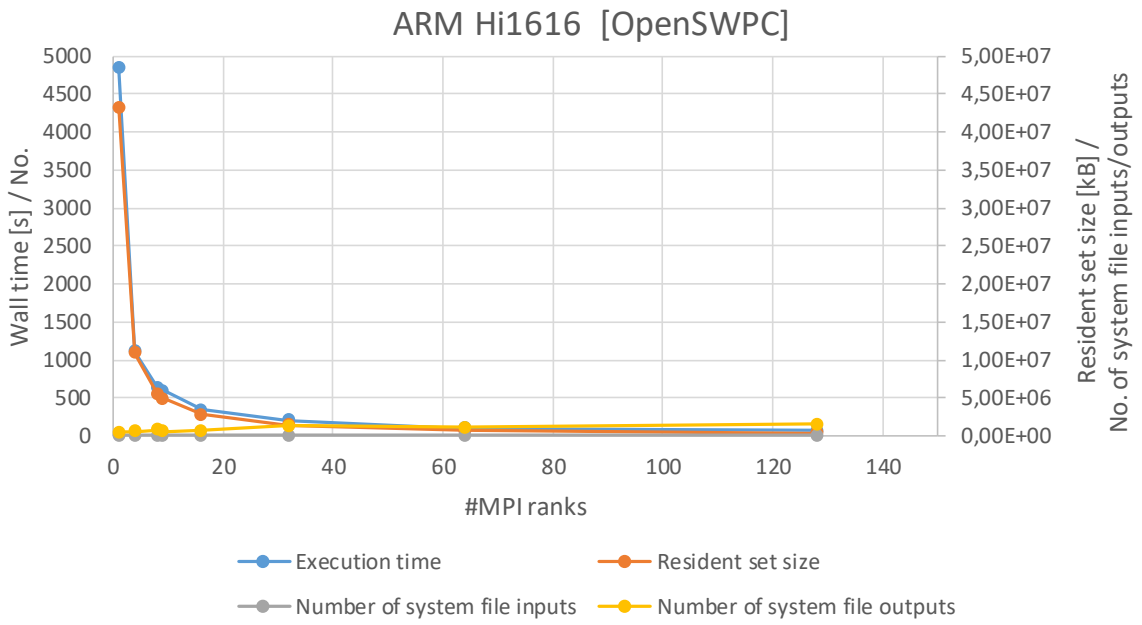

Fig. 30 OpenSWPC results for ARM Hi1616

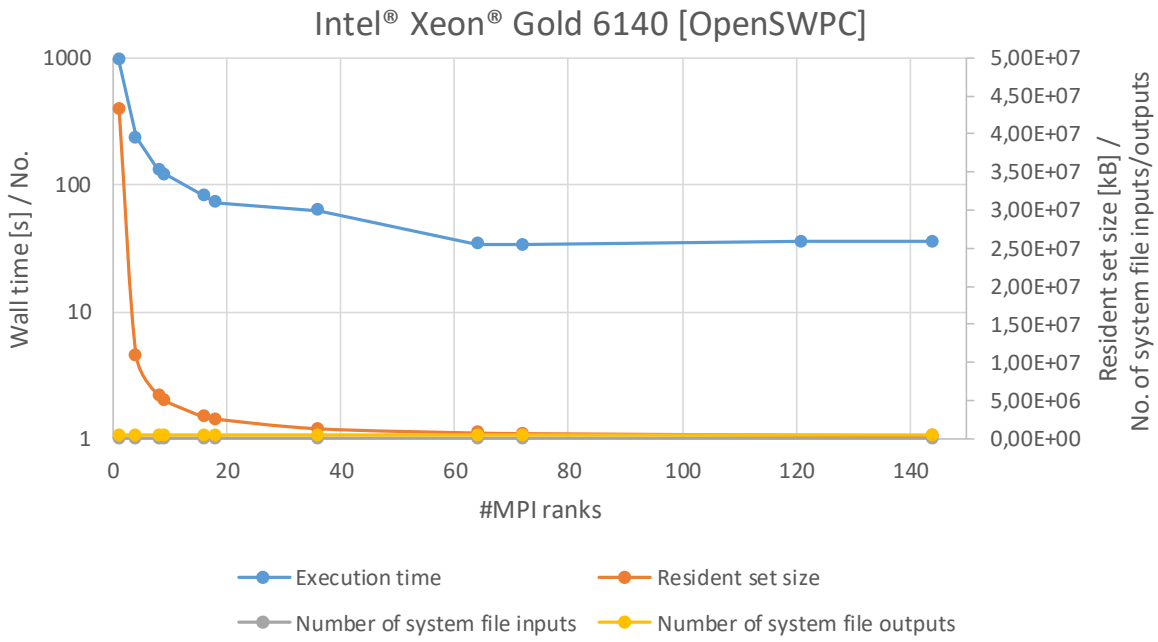

Fig. 31 OpenSWPC results for Intel® Xeon® Gold 6140

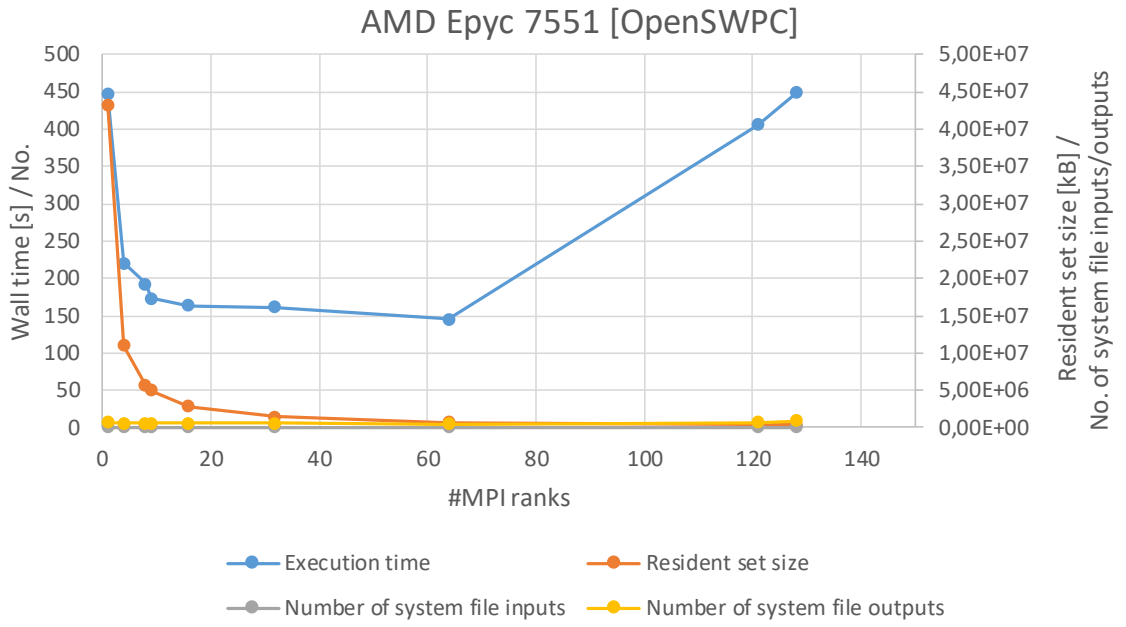

Fig. 32 OpenSWPC results for AMD Epyc™ 7551

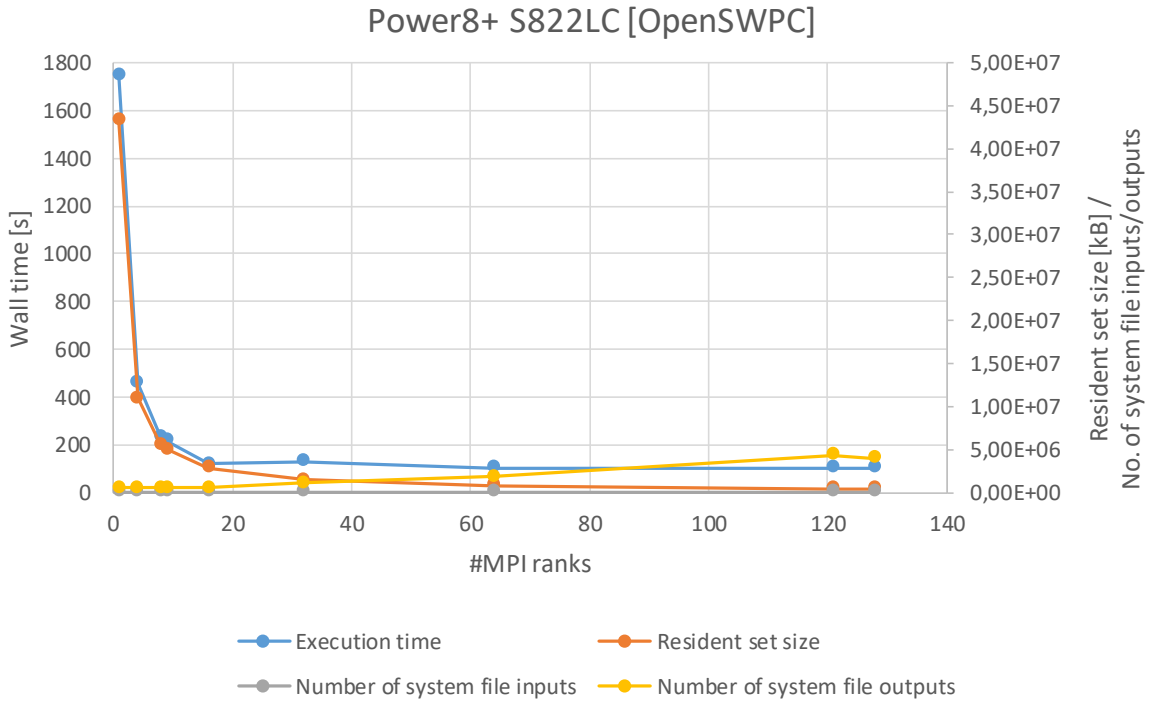

Fig. 33 OpenSWPC results for IBM Power8+ S822LC

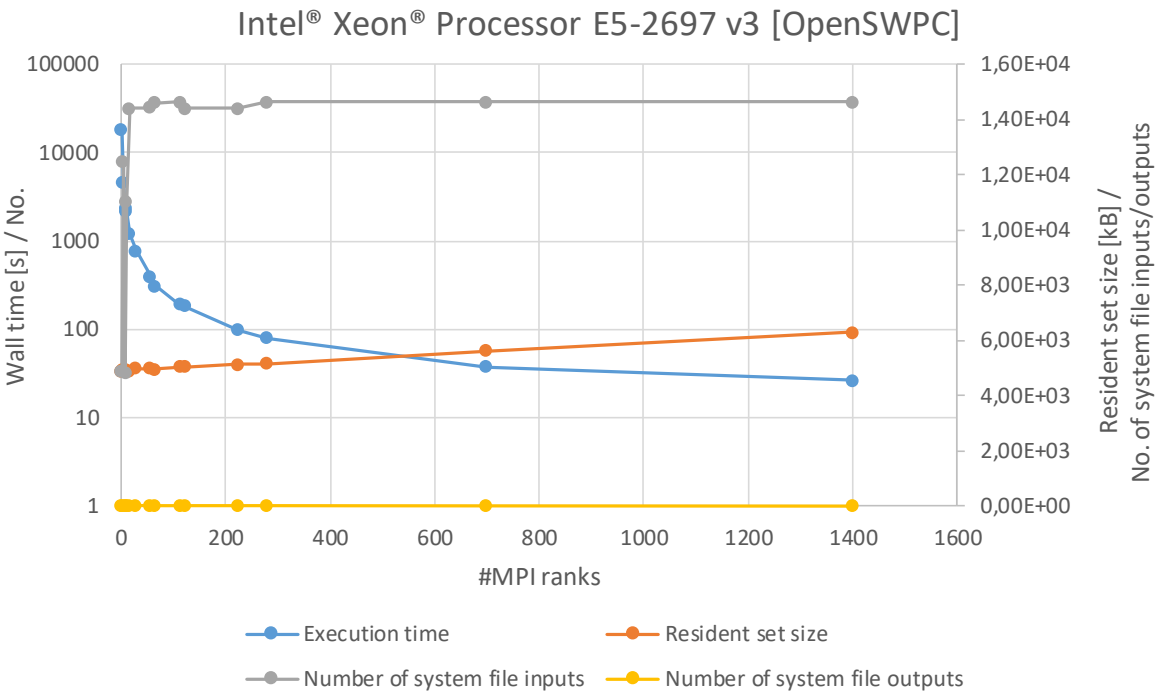

Fig. 34 OpenSWPC results for Intel® Xeon® E5-2697 v3

3. IPF

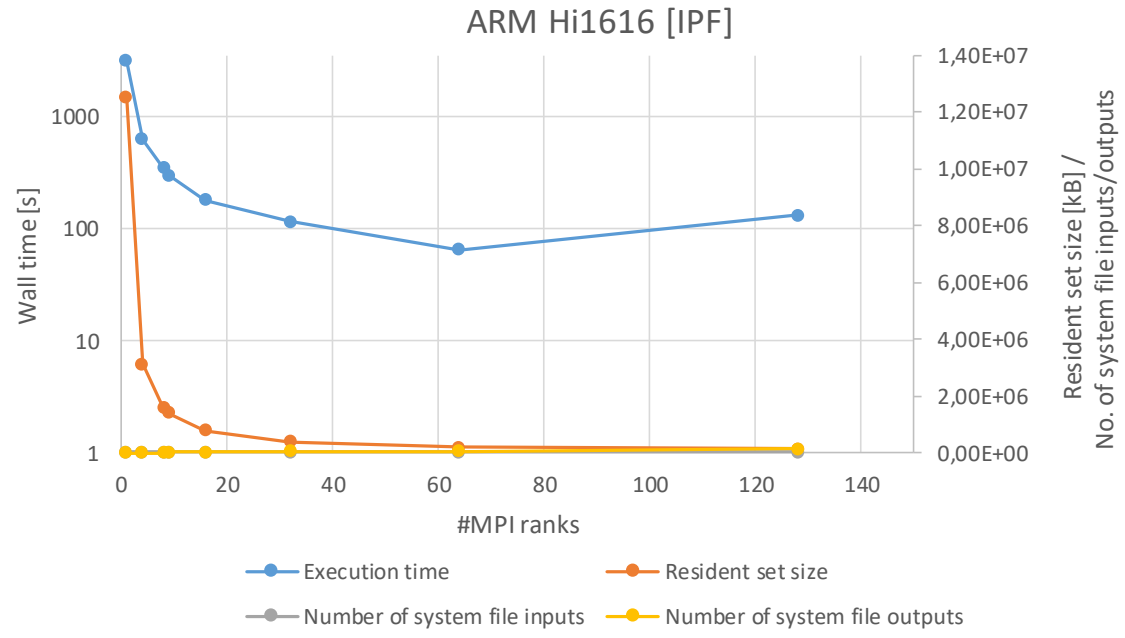

Fig. 35 IPF results for ARM Hi1616

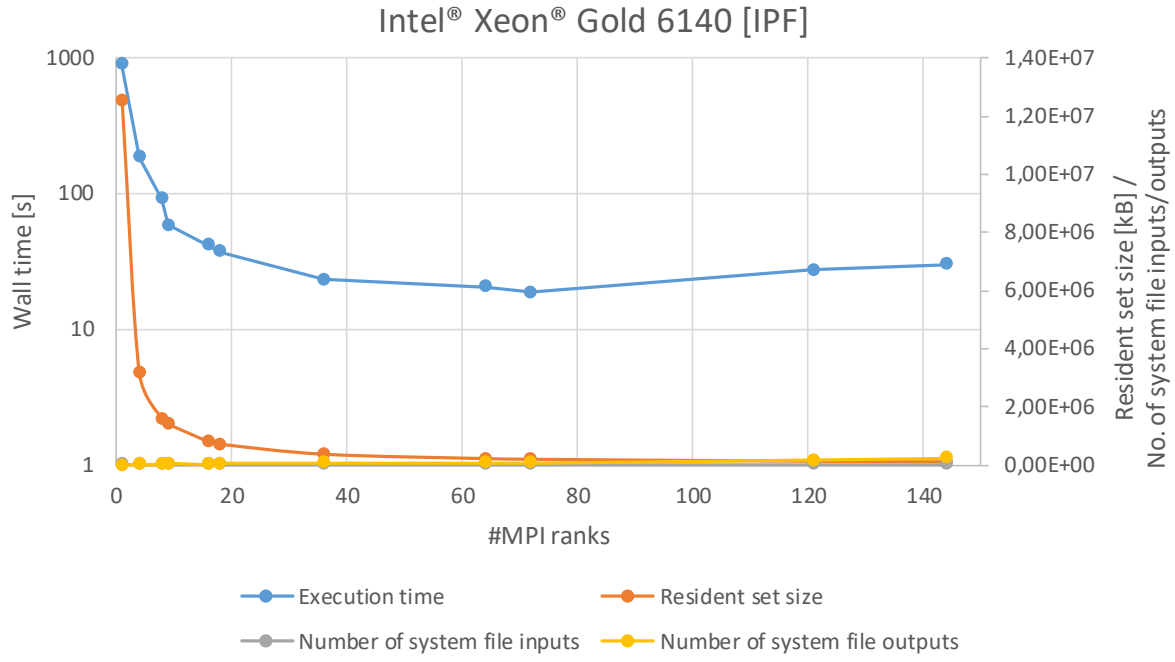

Fig. 36 IPF results for Intel® Xeon® Gold 6140

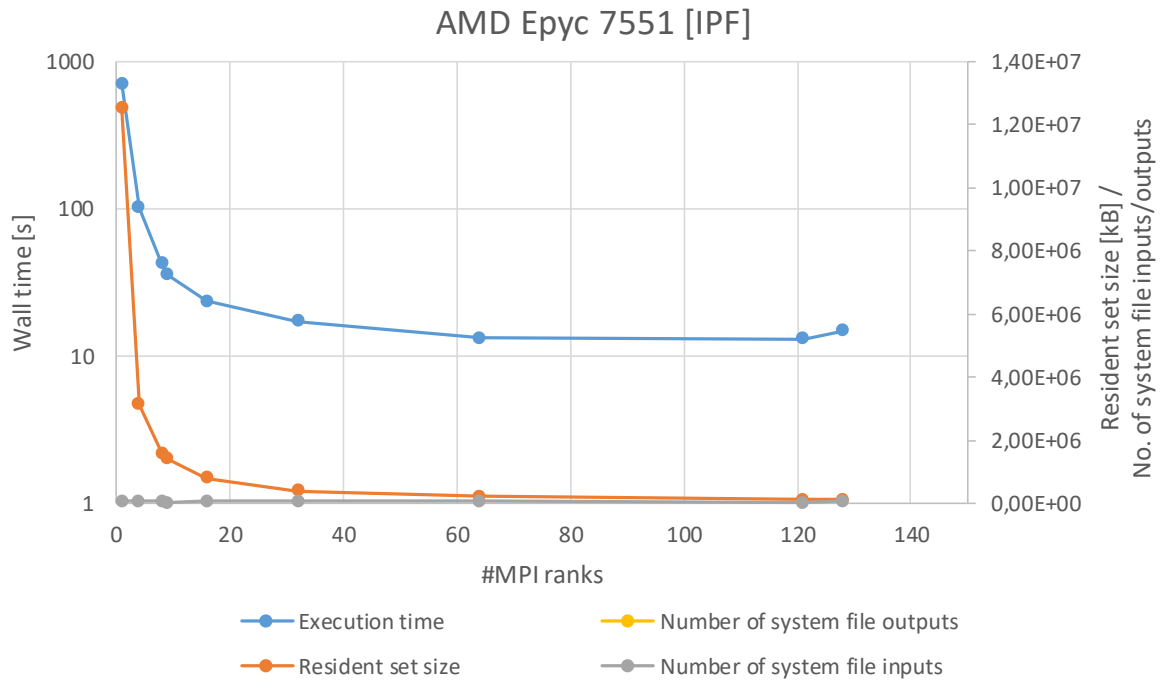

Fig. 37 IPF results for AMD Epyc™ 7551

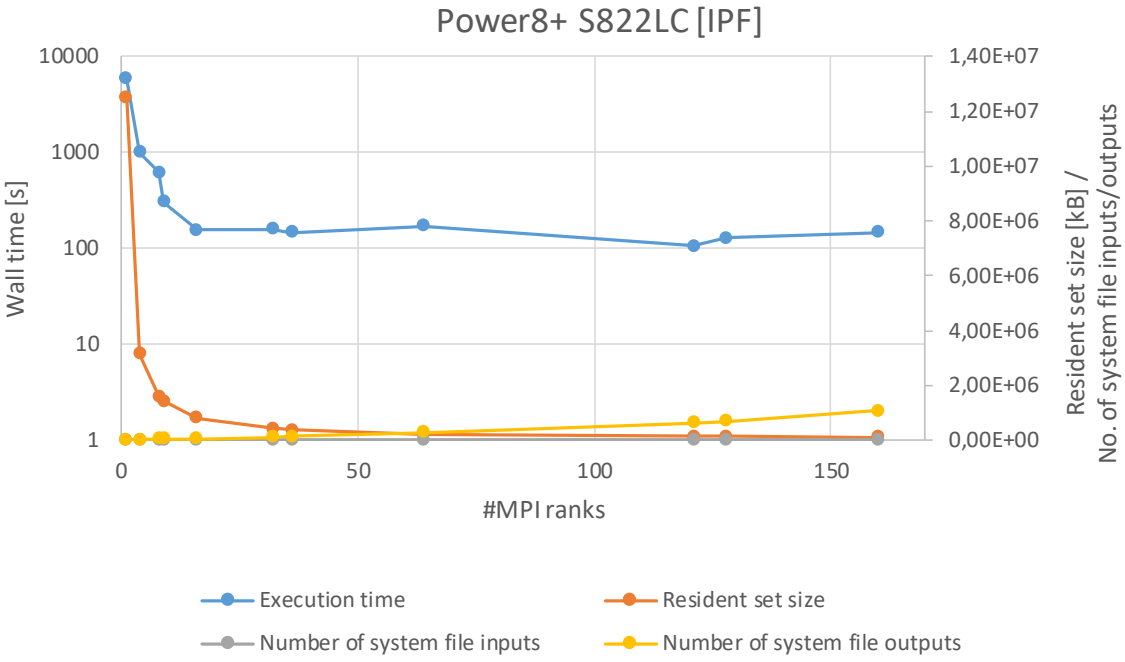

Fig. 38 IPF results for IBM Power8+ S822LC

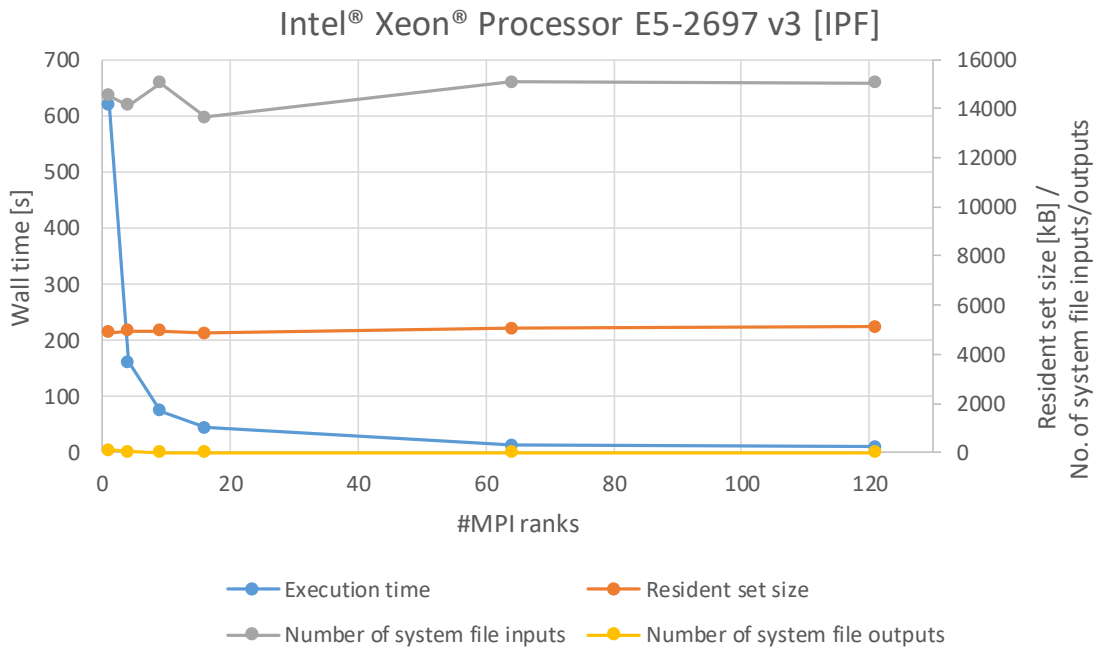

Fig. 39 IPF results for Intel® Xeon® E5-2697 v3

4. ABMS

Layer shape 64x64

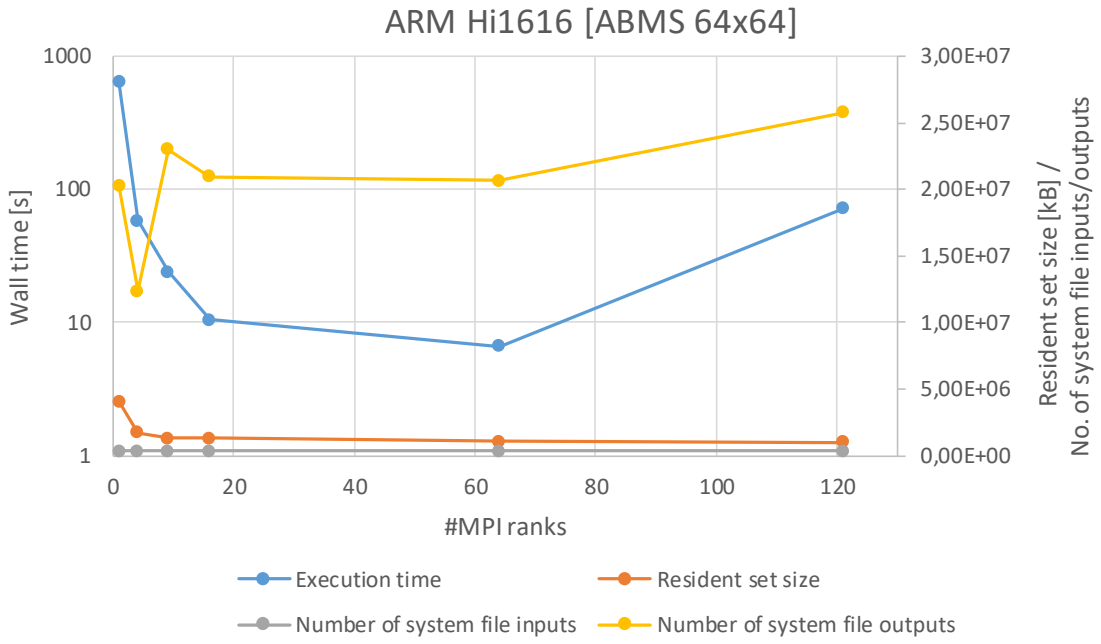

Fig. 40 ABMS results for ARM Hi1616 using layer shape 64x64

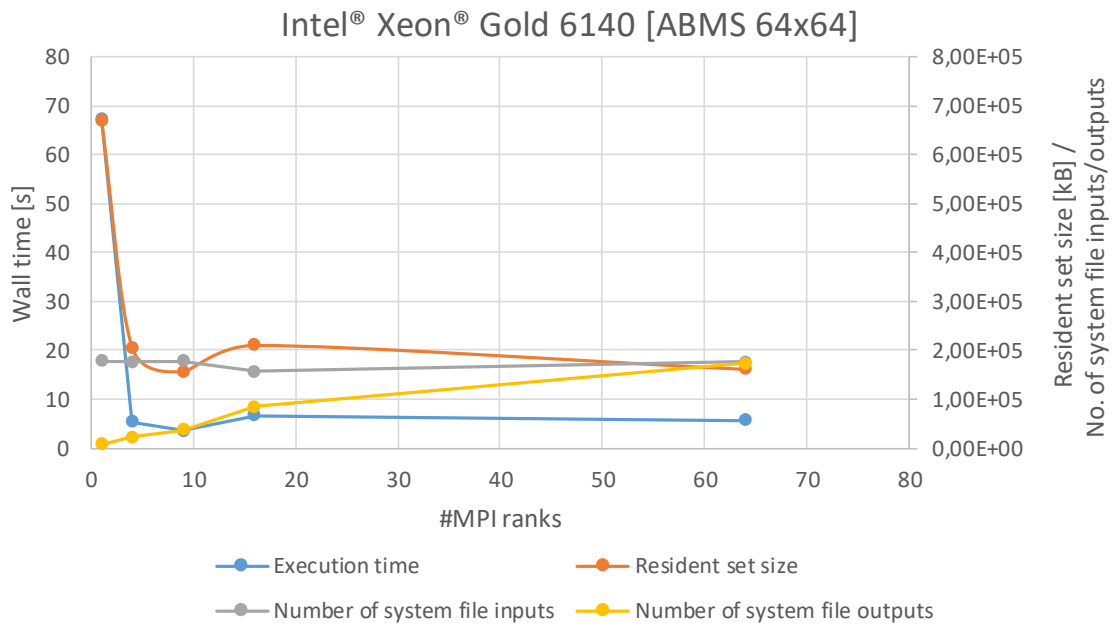

Fig. 41 ABMS results for Intel® Xeon® Gold 6140 using layer shape 64x64

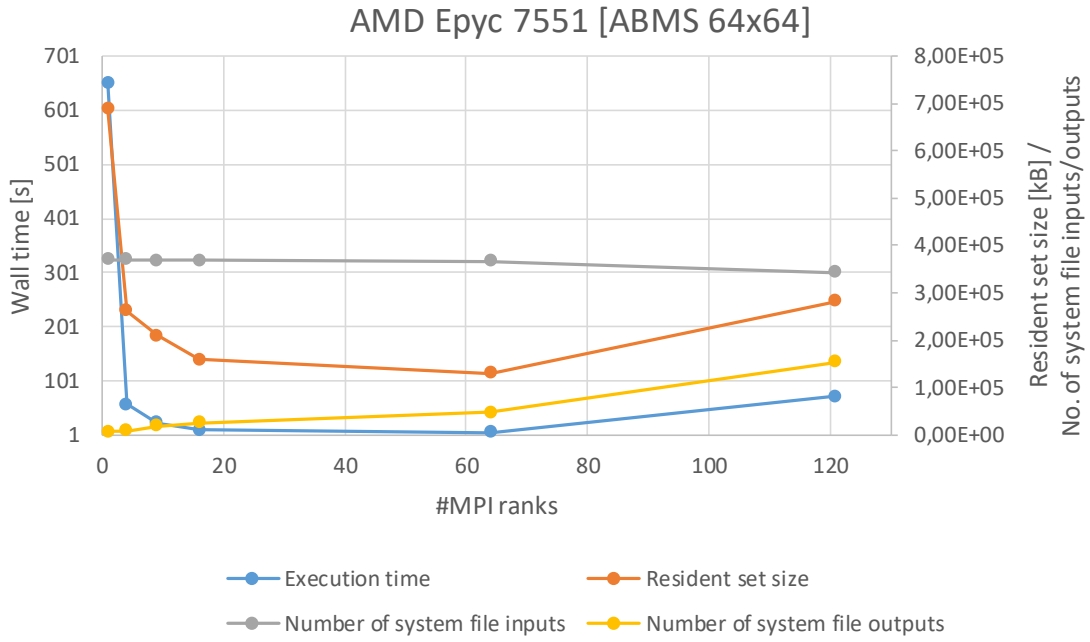

Fig. 42 ABMS results for AMD Epyc™ 7551 using layer shape 64x64

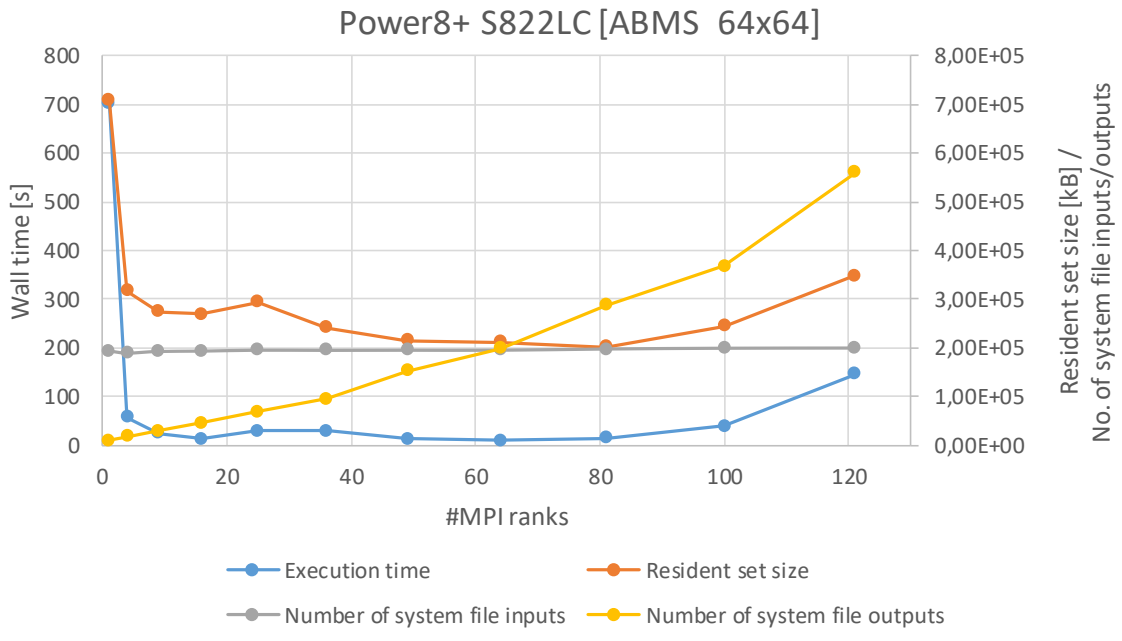

Fig. 43 ABMS results for IBM Power8+ S822LC using layer shape 64x64

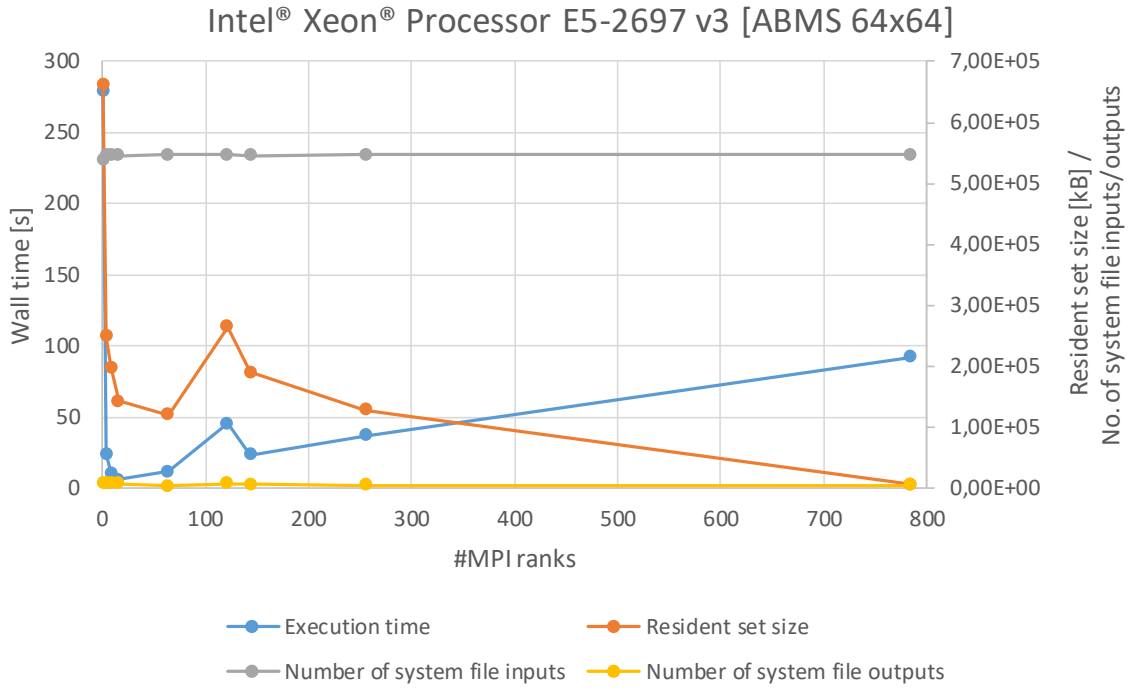

Fig. 44 ABMS results for Intel® Xeon® E5-2697 v3 using layer shape 64x64

Layer shape 128x128

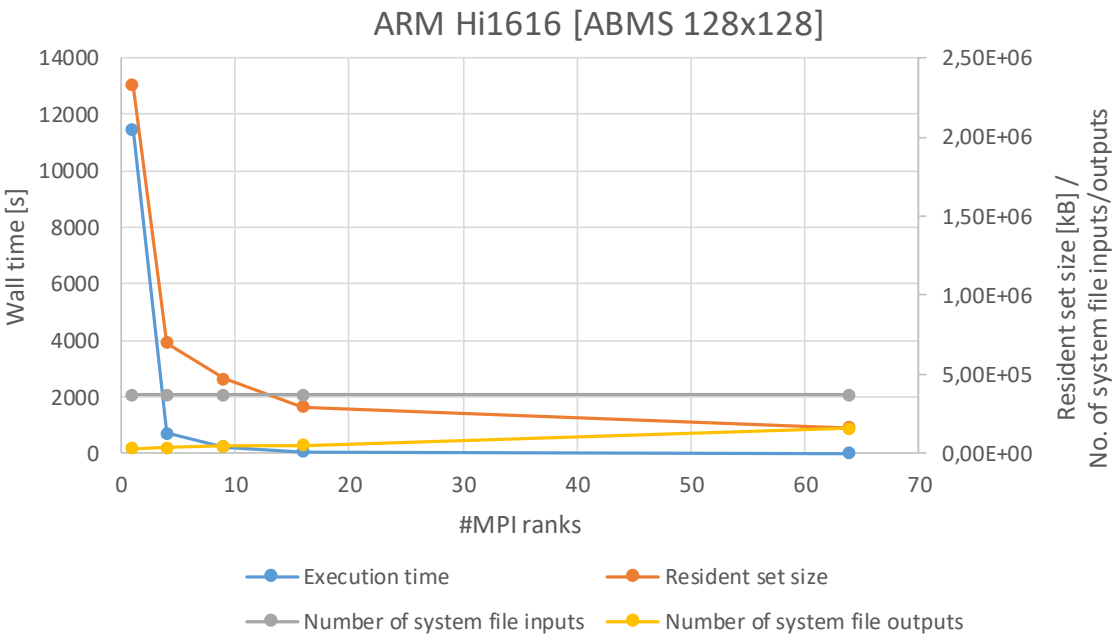

Fig. 45 ABMS results for ARM Hi1616 using layer shape 128x128

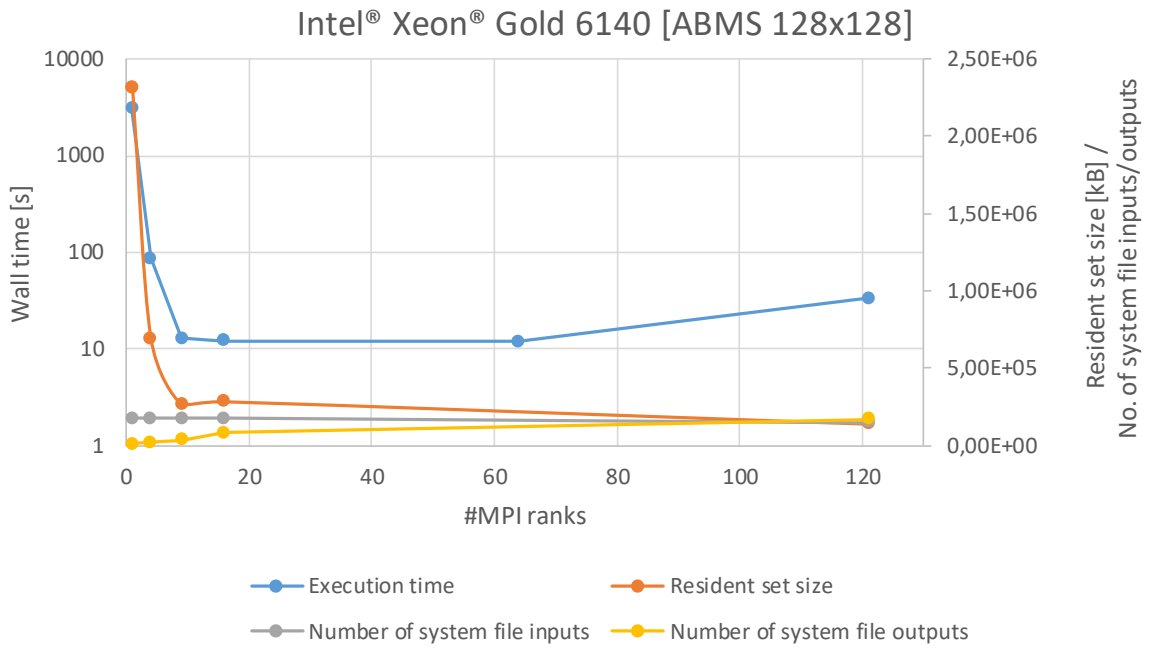

Fig. 46 ABMS results for Intel® Xeon® Gold 6140 using layer shape 128x128

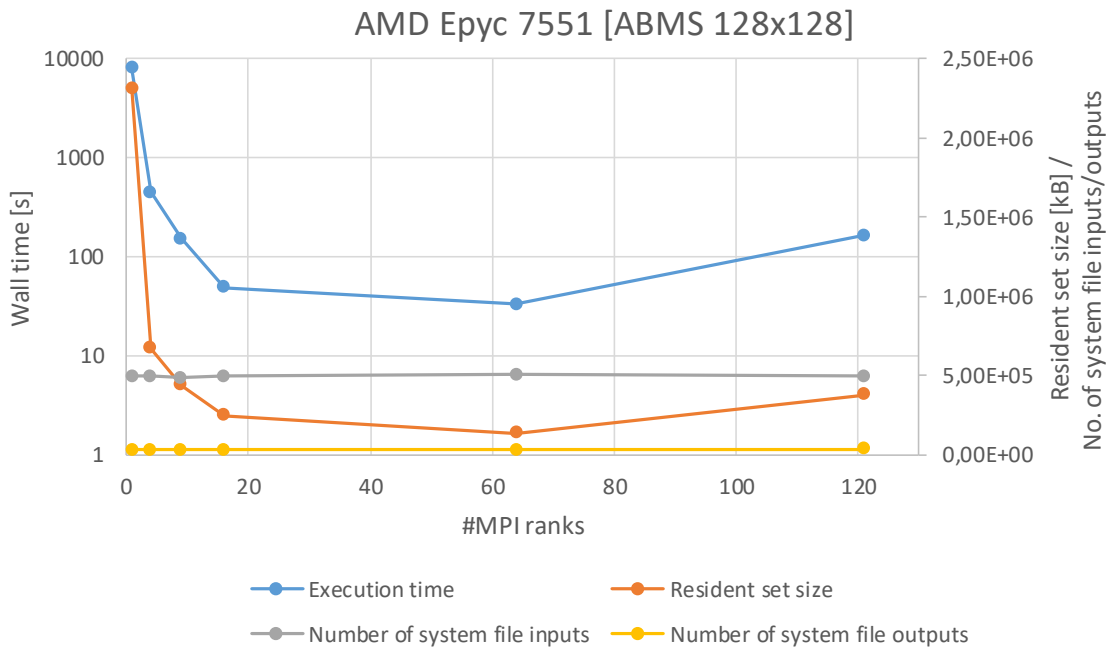

Fig. 47 ABMS results for AMD Epyc™ 7551 using layer shape 128x128

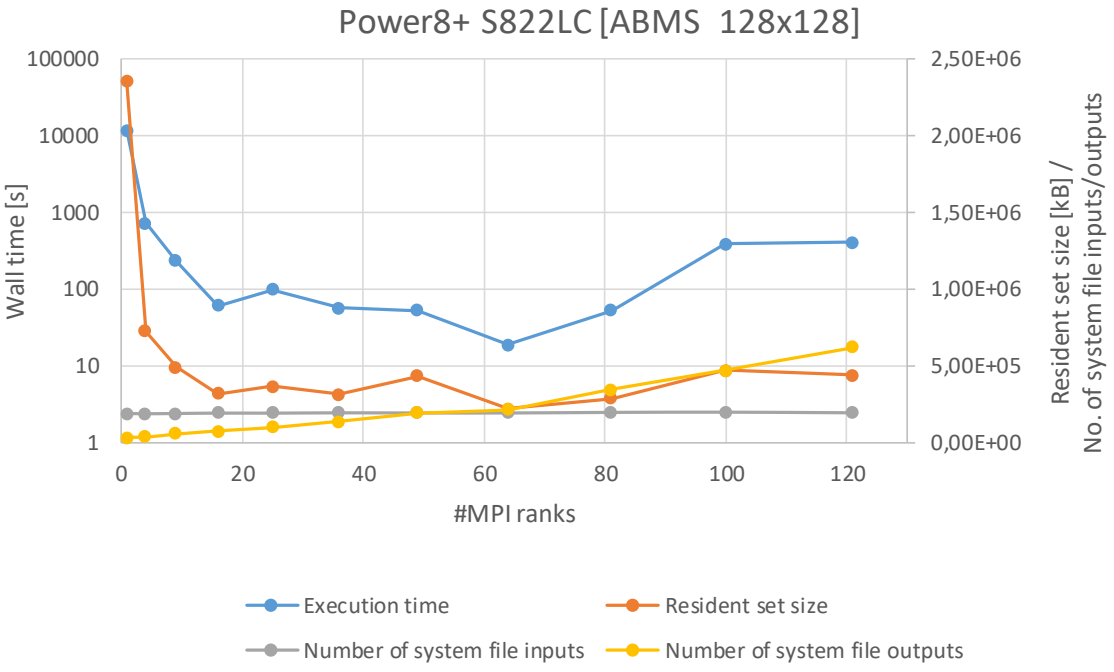

Fig. 48 ABMS results for IBM Power8+ S822LC using layer shape 128x128

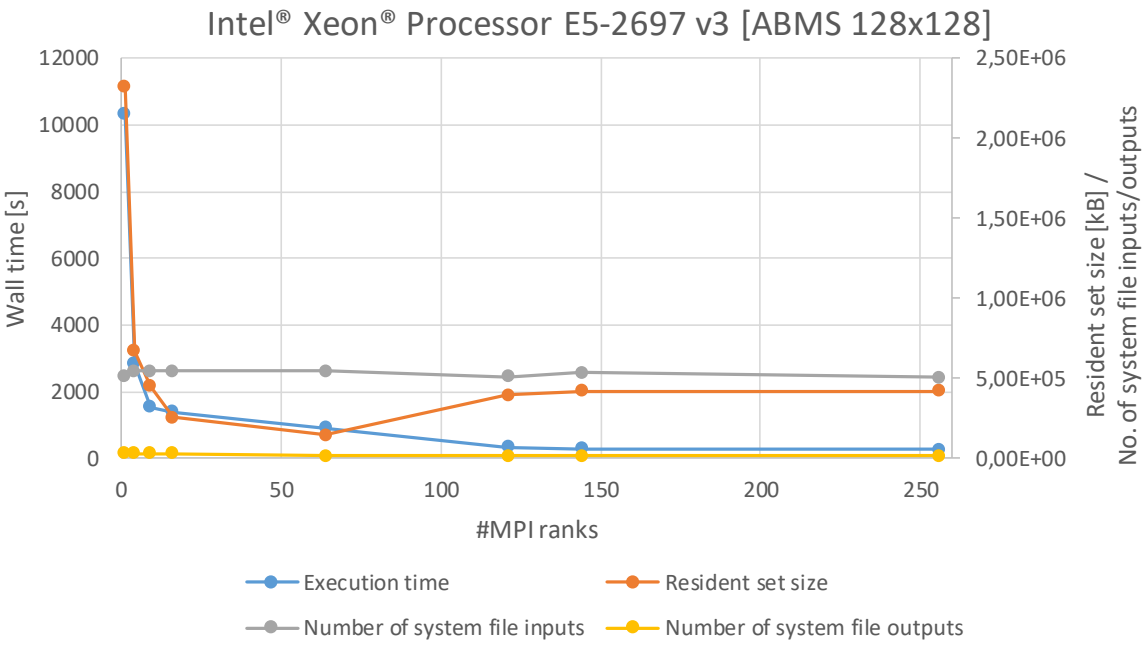

Fig. 49 ABMS results for Intel® Xeon® E5-2697 v3 using layer shape 128x128

5. CMAQ/CCTM

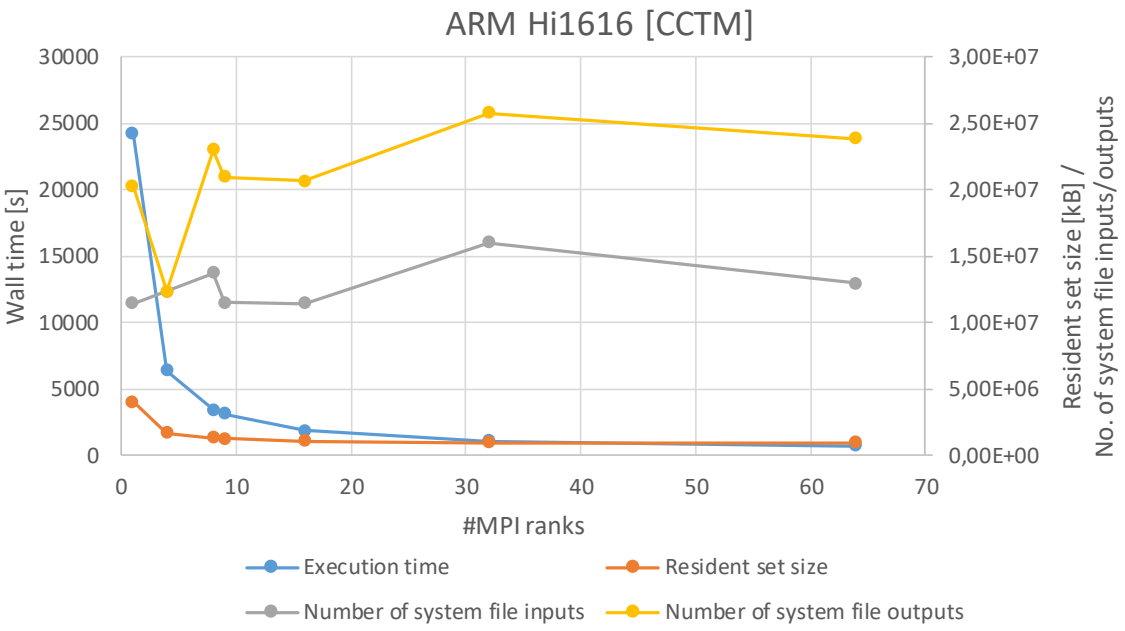

Fig. 50 CMAQ/CCTM results for ARM Hi1616

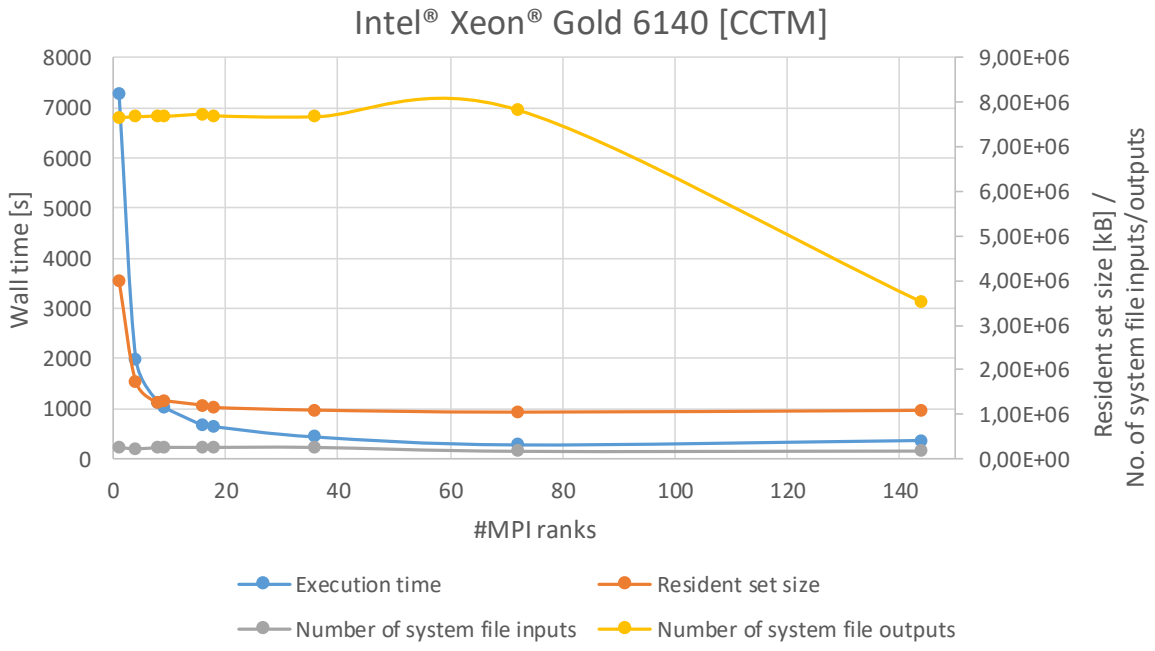

Fig. 51 CMAQ/CCTM results for Intel® Xeon® Gold 6140

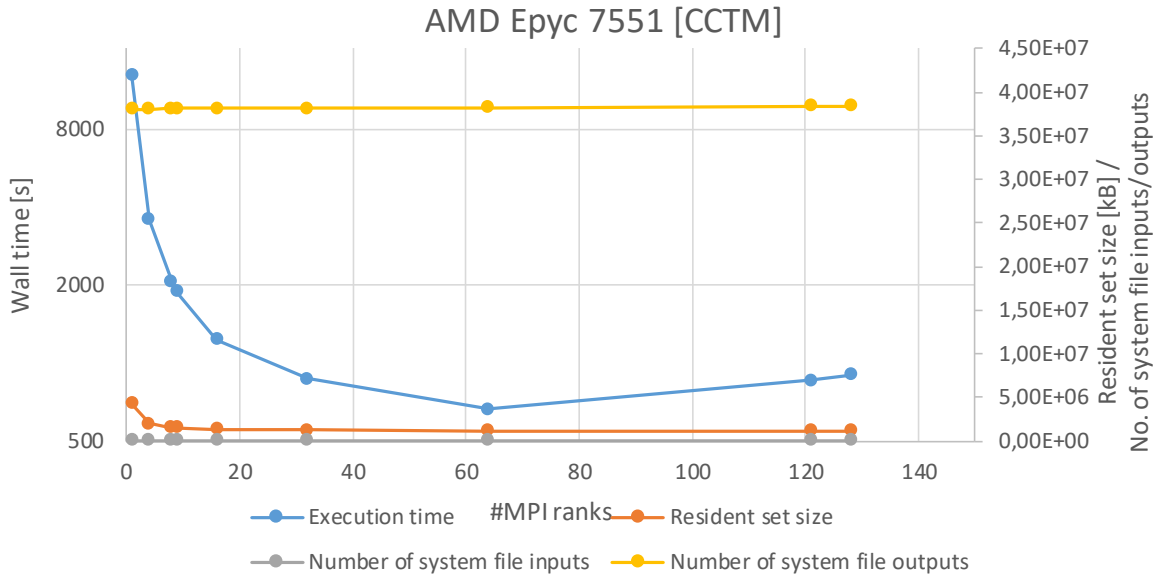

Fig. 52 CMAQ/CCTM results for AMD Epyc™ 7551

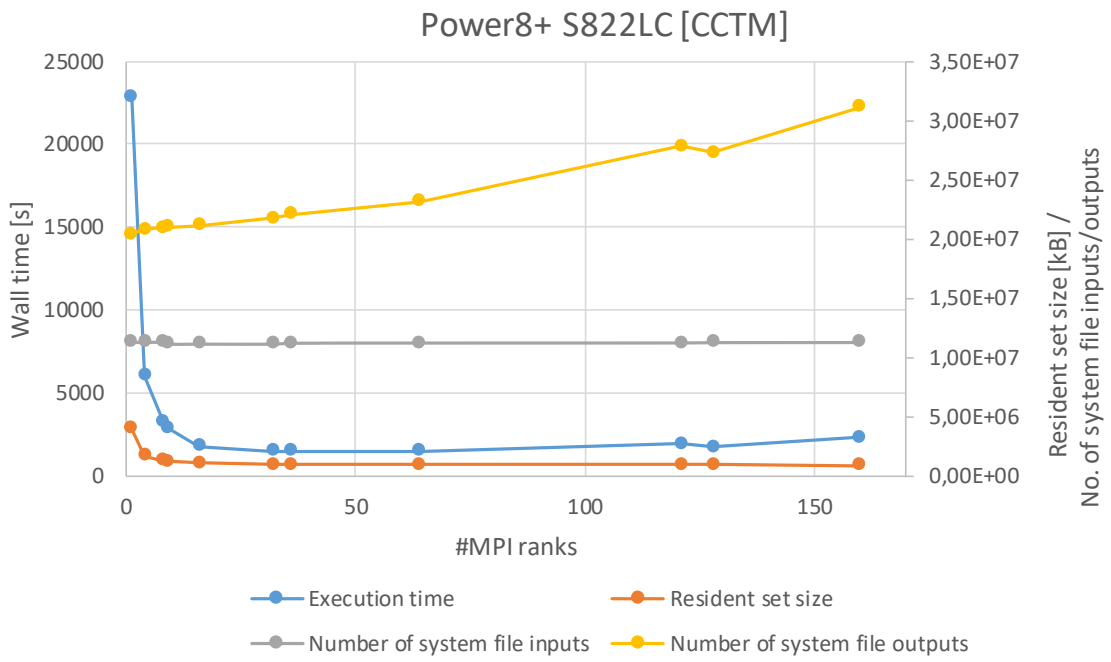

Fig. 53 CMAQ/CCTM results for IBM Power8+ S822LC

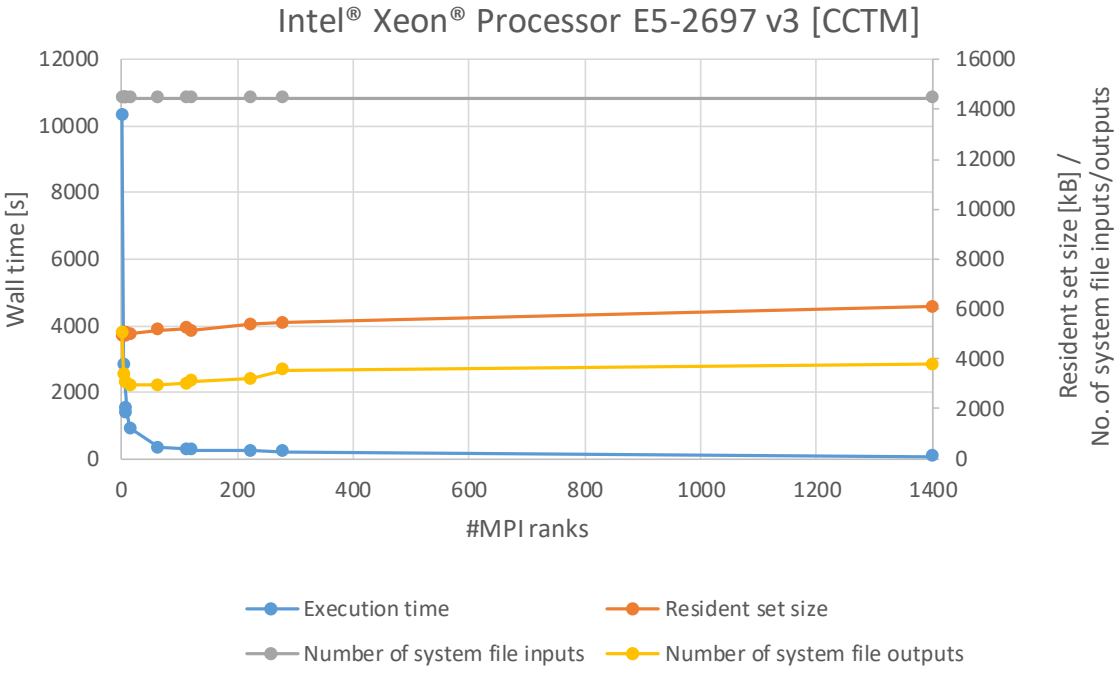

Fig. 54 CMAQ/CCTM results for Intel® Xeon® E5-2697 v3

6. CM1

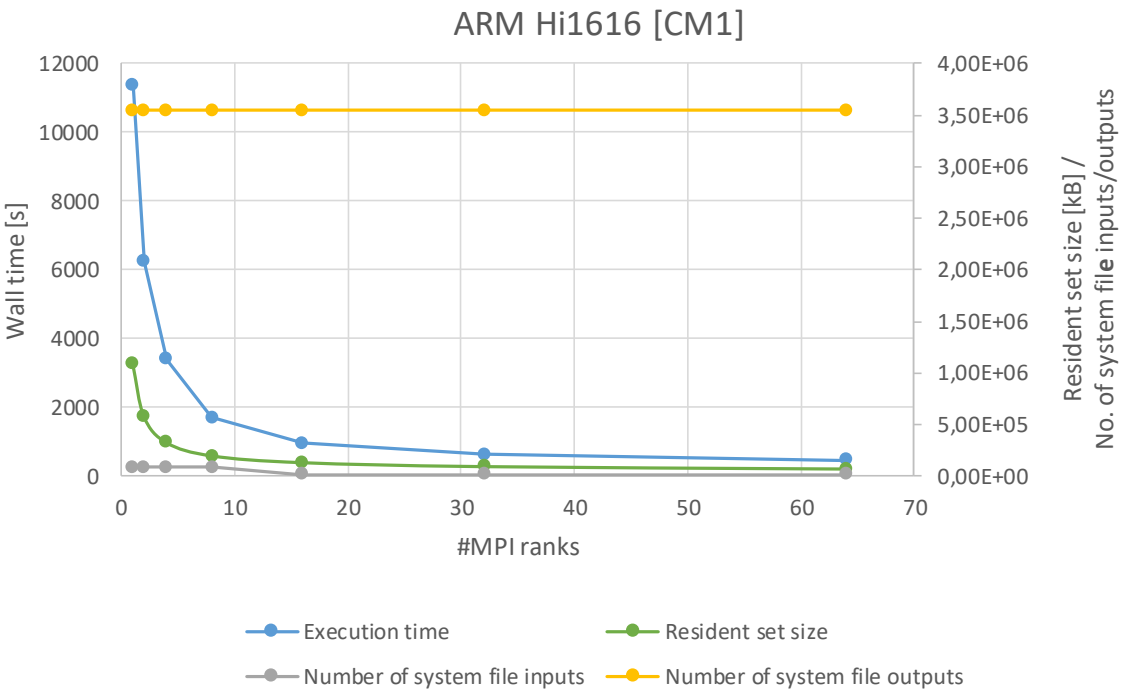

Fig. 55 CM1 results for ARM Hi1616

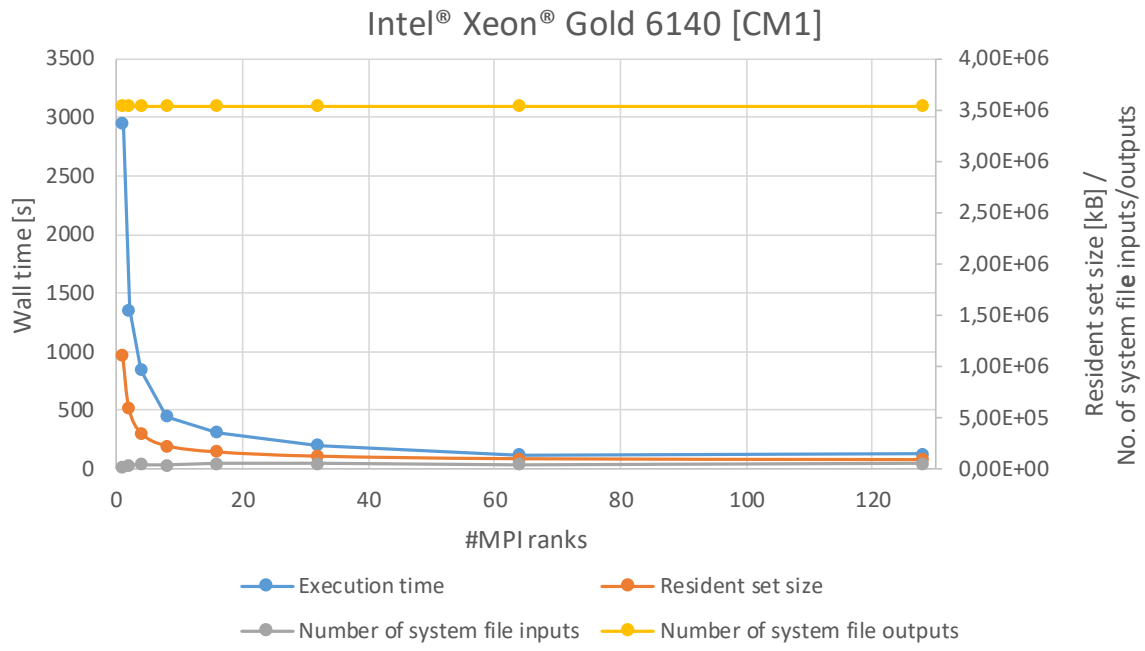

Fig. 56 CM1 results for Intel® Xeon® Gold 6140

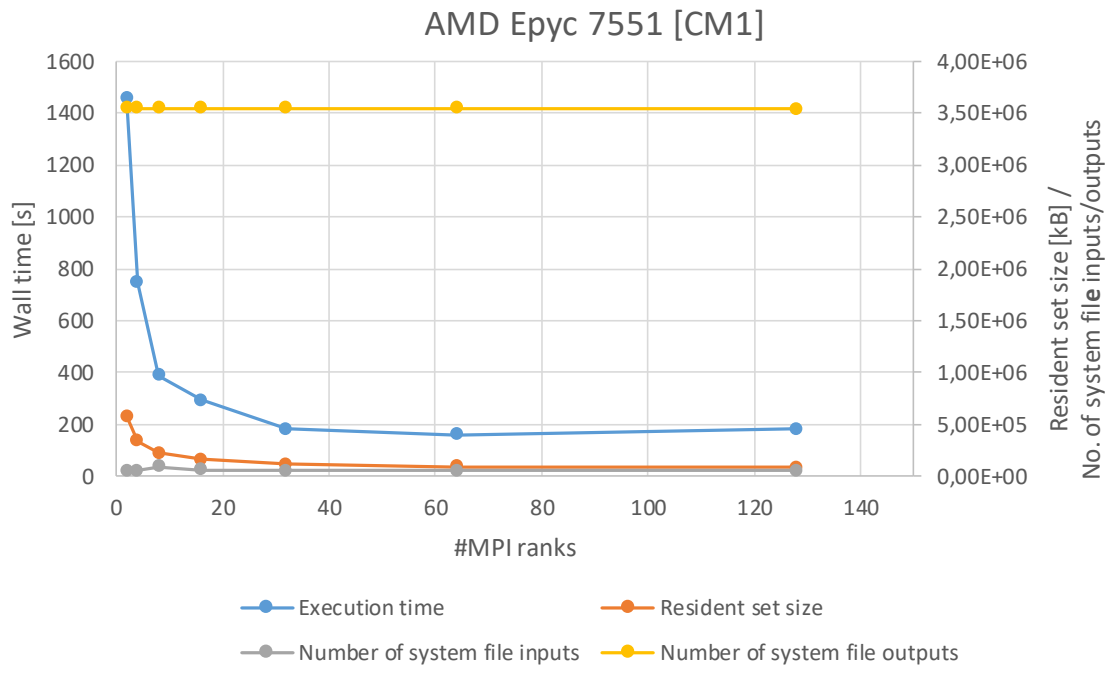

Fig. 57 CM1 results for AMD Epyc™ 7551

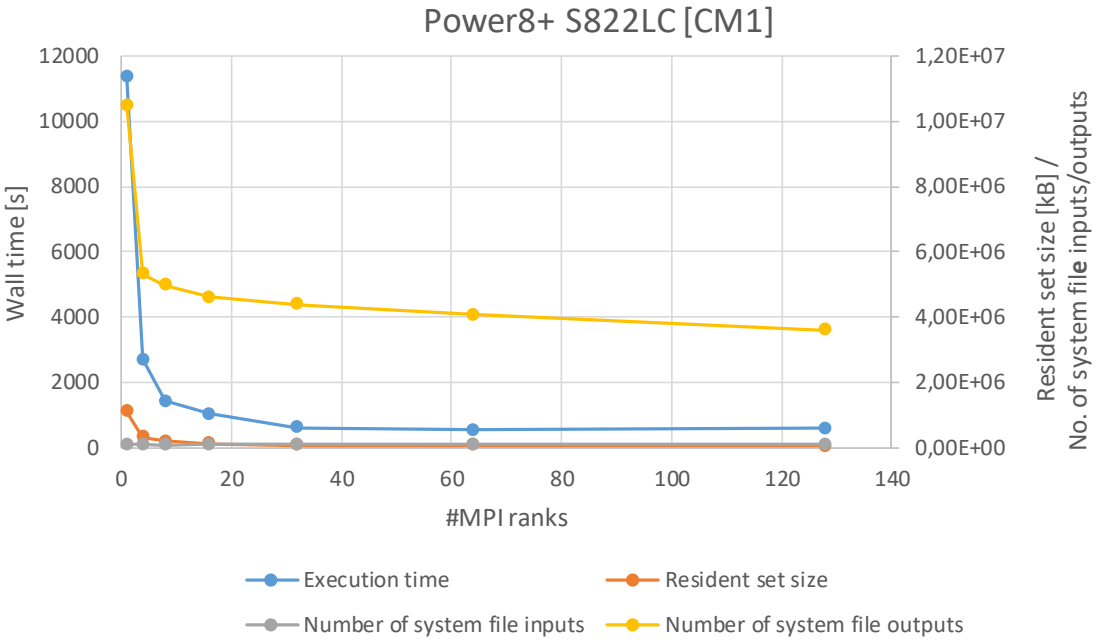

Fig. 58 CM1 results for IBM Power8+ S822LC

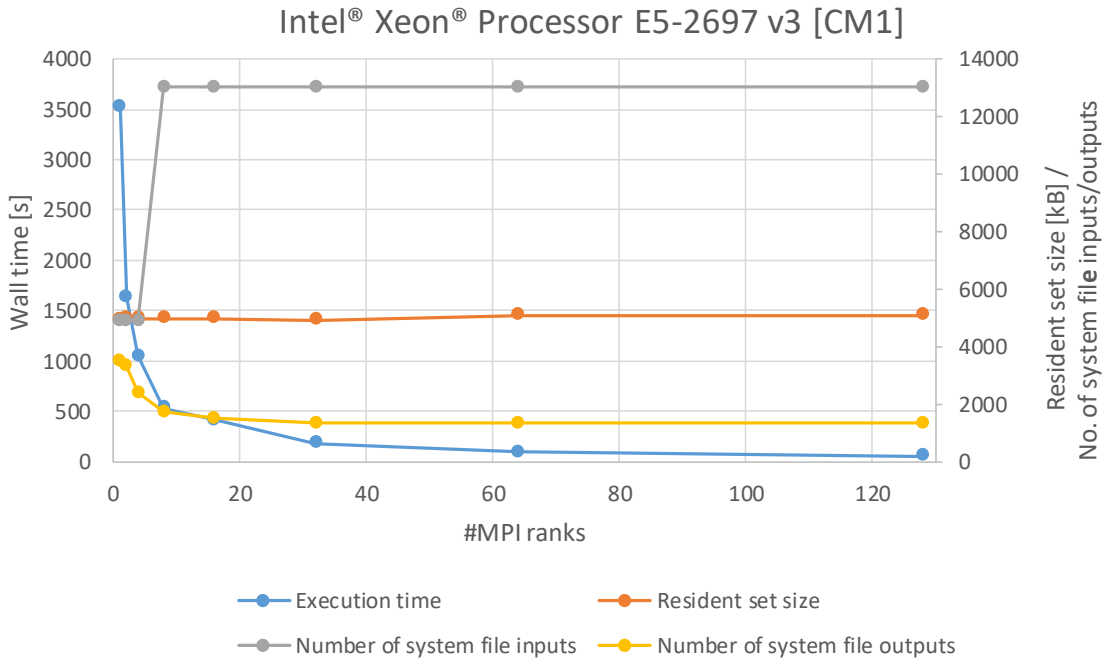

Fig. 59 CM1 results for Intel® Xeon® E5-2697 v3

7. HWRF

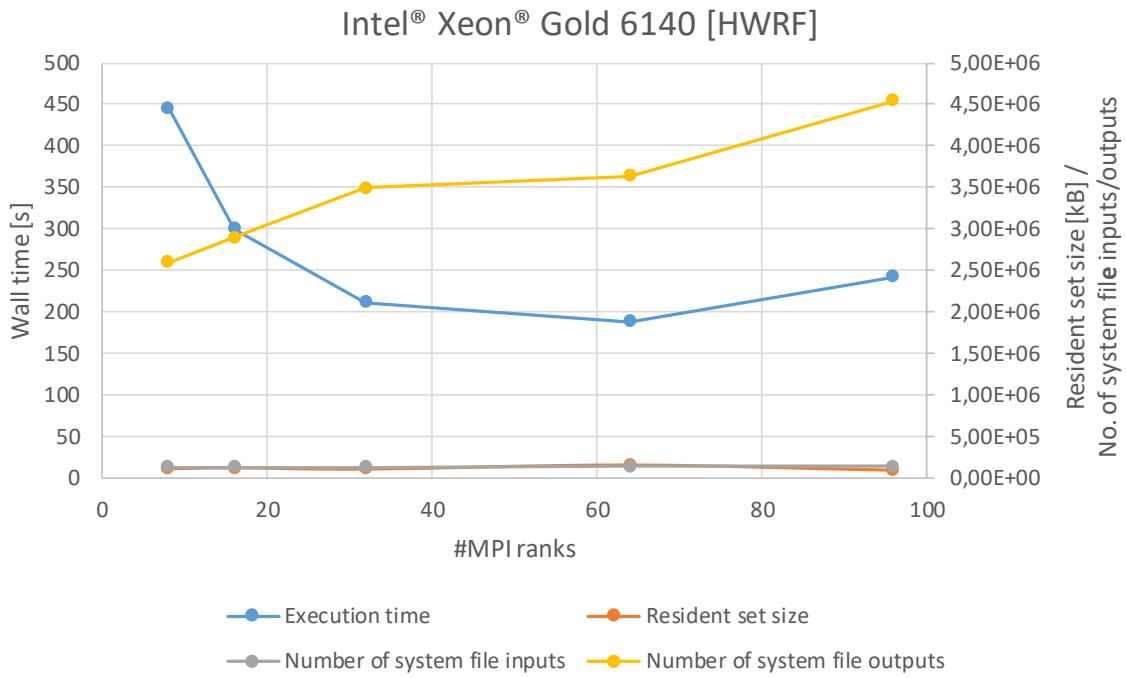

Fig. 60 HWRF results for Intel® Xeon® Gold 6140

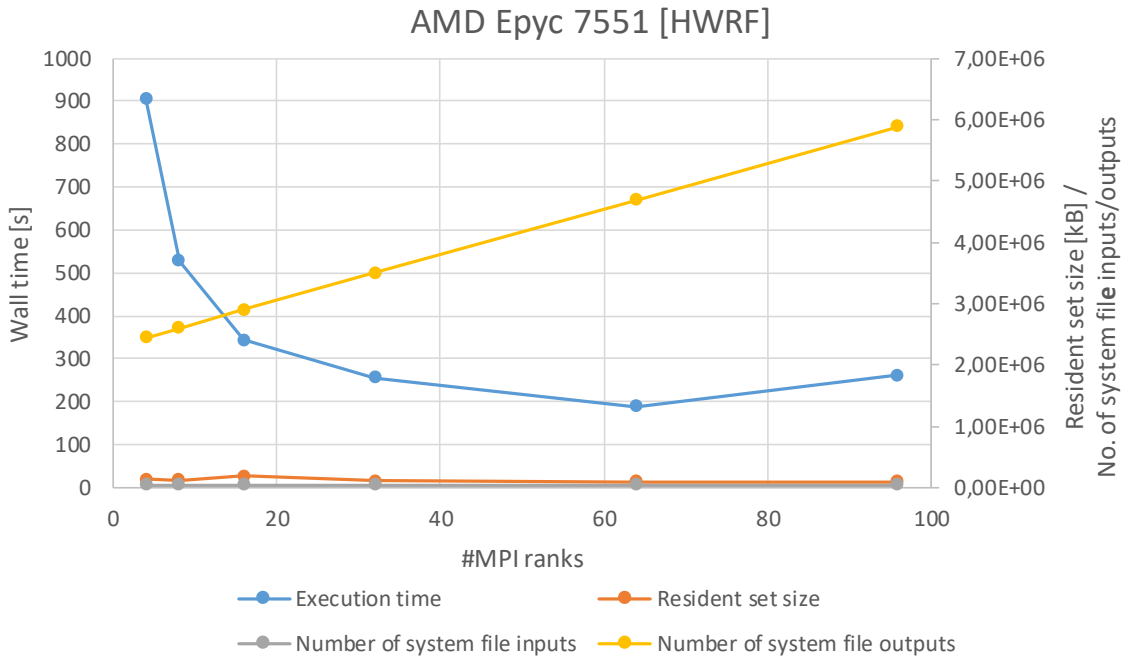

Fig. 61 HWRF results for AMD Epyc™ 7551

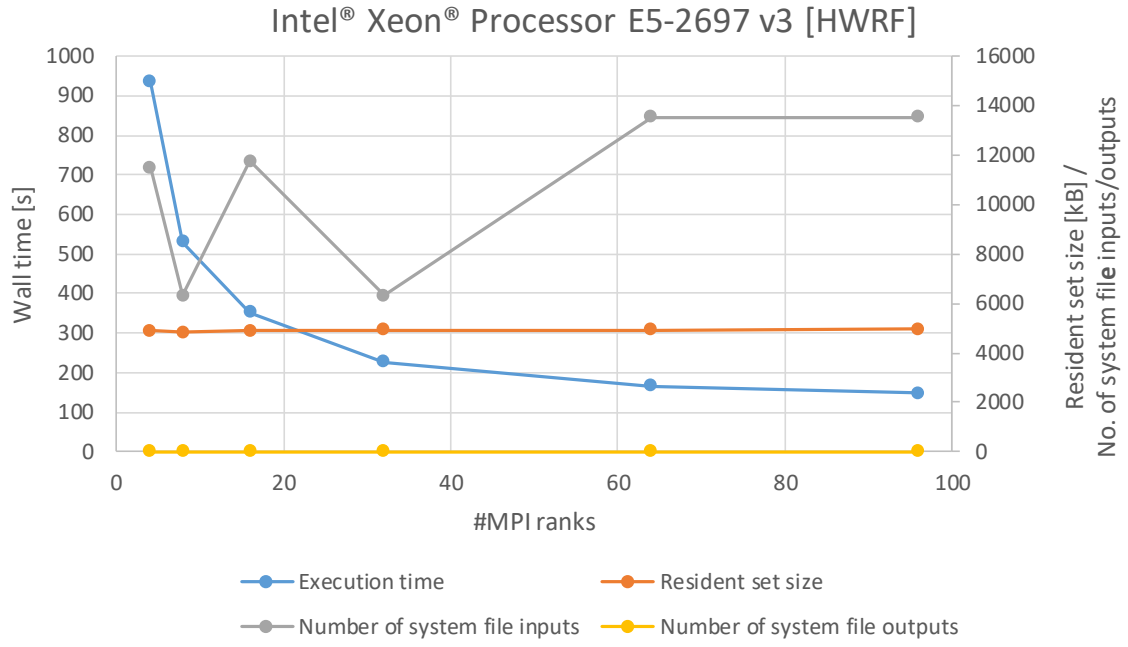

Fig. 62 HWRF results for Intel® Xeon® E5-2697 v3

List of figures

List of tables