DOI: 10.14529/jsfi190202

HPC Processors Benchmarking Assessment for Global System Science Applications

Damian Kaliszan[1]***, Norbert Meyer***[1]***, Sebastian Petruczynik***[1]***,***
Michael Gienger[2]***, Sergiy Gogolenko***[2]

The work undertaken in this paper was done in the Centre of Excellence for Global Systems Science (CoeGSS) – an interdisciplinary project funded by the European Commission. CoeGSS project provides a computer-aided decision support in the face of global challenges (e.g. development of energy, water and food supply systems, urbanisation processes and growth of the cities, pandemic control, etc.) and tries to bring together HPC and global systems science. This paper presents a proposition of GSS benchmark which evaluates HPC architectures with respect to GSS applications and seeks for the best HPC system for typical GSS software environments. The outcome of the analysis is defining a benchmark which represents the average GSS environment and its challenges in a good way: spread of smoking habits and development of tobacco industry, development of green cars market and global urbanisation processes. Results of the tests that have been run on a number of recently appeared HPC platforms allow comparing processors' architectures with respect to different applications using execution times, TDPs[3] and TCOs[4] as the basic metrics for ranking HPC architectures. Finally, we believe that our analysis of the results conveys a valuable information to the broadened GSS audience which might help to determine the hardware demands for their specific applications, as well as to the HPC community which requires a mature benchmark set reflecting requirements and traits of the GSS applications. Our work can be considered as a step into direction of development of such mature benchmark.

Keywords: Global Systems Science, HPC benchmarks, parallel applications, e-Infrastructure evaluation.

Introduction

Global Systems Science (GSS) is a branch of science which uses specific knowledge and techniques to evaluate the impact of policies and people's relation on various global phenomena such as climate change, financial crises, pandemic spread, growth of the cities, human migration, etc. This document addresses the question of "which HPC architectures among the recently introduced are best to run GSS applications most effectively?". Such aliases as "the best" or "most effectively" may obviously have different meanings for different people. While some people might consider it to be the fastest execution time, others might be interested in the price-performance ratio 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 study, the authors acquired cutting-edge processors from four major vendors – Intel, AMD, HiSilicon, and IBM. In particular, we benchmarked GSS applications on Intel®Xeon®Gold 6140 [31] 2-node cluster, AMD Epyc^{TM} 7551 single node, ARM Hi1616 2-node cluster, IBM Power8+ [28] single node, and – as a reference testbed – Eagle cluster located at Poznan Supercomputing and Networking Center (Poland) equipped with Intel®Xeon®Haswell E5-2697 v3 processors. The set of tested applications (called a *GSS benchmark* here) covers many research areas from the entire GSS field. The rest of the paper is organised as follows. Section 1 describes benchmark and the experimental setup. More specifically, we introduce applications chosen

[1]Poznan Supercomputing and Networking Center, Poznań, Poland
[2]High-Performance Computing Center Stuttgart, Stuttgart, Germany
[3]Thermal Design Power
[4]Total Cost of Ownership

for the benchmark and substantiate this choice, present a technical overview of the testbeds where the tests were launched, and discuss the approach selected for measuring the performance metrics. In Section 2, we shortly overview the main benchmarking results. In particular, we emphasise applicability of those results for the hardware/software co-design. Section 3 contains statements for the most relevant conclusions. Finally, we end our paper with a formulation of the directions for future research.

1. Benchmark and Experimental Setup

1.1. Representative GSS Applications Selected for the Benchmark

Functionally, the benchmarked applications can be categorized into two groups: HPC- compliant social simulation software and large-scale CFD (Computational Fluid Dynamics) applications. From the perspective of programming languages, GSS benchmark covered applications are written in C++ and Python, which are the most popular programming languages among GSS experts who use HPC. This subsection contains a short description of tested applications – ABM4Py/GG, Pandora/GG, IPF, OpenSWPC, CCTM, CM1, HWRF – and explains why these particular applications were selected for the benchmark. We refer the interested reader for further technical details to the reports [23, 24].

1.1.1. HPC-compliant Social Simulation Software

Since GSS deals with evaluating impact of different policies on the society, social simulations constitute a relevant part for the majority of workflows in large-scale GSS applications. Typical social simulation component consists of pre-processing, simulation, and post-processing steps. Agent-based modelling and simulation (ABMS) is the most widely accepted tool for the simulation step. Pre-processing step takes care of collecting and wrangling real-word data, as well as synthesising inputs for ABMS. As long as fine-grained data about society are rarely available for public use, large-scale agent-based models usually require synthetic input data – synthetic populations and synthetic social networks – prepared on the base of partial and marginal information. Post-processing includes visualisation, data analytics, model verification, validation and uncertainty quantification.

In our benchmark, the group of social simulation software covers applications for pre-processing and simulation of agent-based models. In the benchmark definition, we intentionally skipped post-processing step as it frequently heavily depends on the concrete GSS problems in hand, so it is hard to determine typical use-cases and computational kernels for post-processing. We selected two ABMS frameworks – ABM4Py is implemented in Python and follows graph-based ABMS methodology particularly suitable for large-scale social simulation, while Pandora is written in C++ and uses regular meshes to simulate the environment. The pre-processing step is represented by HPC-compliant implementation of the IPF method for generating synthetic populations. The text below shortly describes each application and the specific inputs used into benchmark.

IPF (iterative proportional fitting) is a "technique that can be used to adjust a distribution reported in one data set by totals reported in others. IPF is used to revise tables of data where the information is incomplete, inaccurate, outdated, or a sample" [6]. This procedure reconstructs a contingency matrix based on the known marginals in an unbiased way. Nowadays, IPF and

its derivatives (e.g., IPU) constitute the computational core of the most popular techniques for generating synthetic data which serve as an input for agent-based social simulations [18].

Despite its popularity, we are not aware of any HPC-compliant open-source implementation of IPF. In order to overcome this obstacle, we coded a simple IPF process using linear algebra kernels from ScaLAPACK. This simple implementation was a baseline for our IPF benchmark.

Note: The above-mentioned IPF codes are not published with open access on the Internet yet.

ABM4Py/GG (ABMS for Python) is a distributed agent-based modelling and simulation framework for fast prototyping agent based components of GSS models [17]. Agent-based models implemented in ABM4Py follow the graph-based representation [17, 29]. Namely, agents and loci of interactions are interconnected into a complicated dynamic network, and agent-based simulation reflects possible temporal evolution of this network. This network is further split into subgraphs with graph partitioning software and distributed between MPI processes.

In order to benchmark this framework, we implemented the green growth agent-based model, first proposed in [7], within ABM4Py framework. The green growth model is an innovation-diffusion model for electric cars with a global scope and a fine-scale spatial data resolution. This model allows measuring the most relevant performance metrics for ABM4Py, including elapsed time, I/O, waiting time, and synchronisation time. In order to enable comparison of the ABM4Py framework with Pandora frameworks (see below), our toy implementation uses 2D mesh as a topology of the environment. More specifically, we test against two cases – the one where layer-shape size is set to 64x64, and the second with a size of 128x128. The project's repository is available at [19].

Pandora/GG application serves for benchmarking of Pandora agent-based modelling and simulation framework. In contrast to ABM4Py, Pandora only supports raster inputs, which restricts this framework to agent-based models with 2D mesh as a topology of the environment. Pandora parallelises the simulation process via splitting of rasters on even pieces and distributing them between MPI processes. On the other hand, Pandora is implemented in C++, which allows reaching higher performance compared to Python-based ABM4Py framework.

Similarly to the ABM4Py use-case, our Pandora/GG application implements the green growth agent-based model from [7]. The project's repository is available at [12].

1.1.2. Large-scale CFD Applications

A wide variaty of GSS applications – from evacuation planning [21] to air quality control [15] – rely on coupling of ABMS with CFD. The group of benchmarked CFD applications includes large scale tools that simulate GSS-related scenarios like natural disasters (hurricanes, earthquakes), spread of air pollution, and weather forecast. We selected 4 exemplar open-source codes for large-scale CFD simulations:

- OpenSWPC – an integrated parallel simulation code for modelling seismic wave propagation in 3D heterogeneous viscoelastic media which is applicable for evacuation planning in case of earthquakes;
- CMAQ – a community multiscale air quality modelling system, which was successfully used for conducting large scale air quality simulations and policy making for air pollution control [15];

- CM1 – a model for studying processes in the Earth's atmosphere, which can be used for weather forecast in different GSS applications where behaviour of agents in ABMS component of GSS model depends on weather (e.g., predicting refugee destinations [30]);
- HRWF – parallel implementation of the hurricane weather research and forecasting which is suitable for evacuation planning in case of hurricanes.

OpenSWPC (Open-source Seismic Wave Propagation Code) is an open-source software from large-scale simulation of seismic waves propagation (2D or 3D) by solving motion equations using the finite difference method (FDM) [8]. OpenSWPC is widely used in seismology. It ports easily and delivers good performance on different distributed systems varying from small PC clusters to large-scale supercomputers. The project's repository is available at [10].

CMAQ/CCTM (Community Air Multiscale Quality Modelling System) is an active open-source project of the U.S. EPA (*Environmental Protection Agency*) that delivers a suite of programs for conducting the air quality model simulations. CCTM (*The CMAQ Chemistry-Transport Mode*) is a parallel implementation of the advanced chemical transport model in CMAQ which is often used in computer-aided policy making for improving air quality [15]. It is the only CMAQ program that can be run in parallel.

CCTM runs require large input datasets with a complicated file structure. In our study, we used the official single day simulation benchmark dataset distributed by EPA [5]. Both the project description and the application are available at [1].

CM1 is a three-dimensional, time-dependent, non-hydrostatic numerical model. 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 [11]. Both the project description and the application are available at [9].

WRF (Weather Research and Forecasting model) is an example of a well-scalable application, which motivated us to add it to the set of tests in order to increase the variety of requirements of the GSS benchmark. The Hurricane Weather Research and Forecasting (HWRF) model is a specialised version of the WRF model [16]. It is used to forecast the track and intensity of tropical cyclones. Both the project description and the application are available at [2].

This document does not go further into theory as it is beyond its main subject.

1.2. Configuration of Testbeds Used for the Benchmark

We executed our benchmark on four testbeds using cutting-edge processors recently introduced by four major processor vendors: Xeon®Gold 6140 from Intel [31], AMD Epyc™ 7551 from AMD [3, 25], ARM Hi1616 from HiSilicon [4, 20], and Power8 from IBM [13, 14]. While Tab. 1 summarises relevant technical characteristics of our testbeds, the following paragraphs describe important architectural improvements introduced into the processors.

Intel® Xeon® Gold 6140 (SkyLake SP). The new core for Skylake-X, technically called the Skylake-SP core architecture, delivers new improvements compared to the previous Broadwell-E platform. One of those "upgrades" has been targeted at a better SIMD performance: clustering multiple data entries into a single element and performing the same operation to each of them at

Table 1. Testbed characteristics

	Intel® Xeon® Gold 6140	AMD Epyc™ 7551	ARM Hi1616	Power8+ S822LC
No. of nodes	2	1	2	1
Cores per node	36	64	64	20
CPU Frequency [GHz]	2.3	2	2.4	2.92
L1 (data) cache	1.125 MB	2 MB	2 MB	1.25 MB
L2 cache	36 MB	32 MB	16 MB	10 MB
L3 cache	49.5 MB	128 MB	64 MB	160 MB
RAM type (channels)	DDR4 (6)	DDR4 (8)	DDR4 (4)	DDR3 (4)
RAM frequency [MHz]	2666	2400	2400	1333
RAM capacity [GB]	192	512	128	512
I/O and disks type	SSD	SSD	SSD	SSD
Network	10Gb Ethernet	–*	10Gb Ethernet	–*
TDP [W]	140	180	70	190
Processor price [USD]	2450	3743	300	1500
OS version	Ubuntu 16.04.03 LTS	CentOS 6.9	EulerOS release 2.0 (SP2), Ubuntu 16.04.3 LTS	Ubuntu 16.04.2 LTS
Total bandwidth estimates (per node) [GBps]				
L1 (data) cache	15897	6144	19660	2803
L2 cache	5299	8192	19660	2803
L3 cache	5299	8192	19660	3738
Total memory	119.21	158.95	71.53	230
SMP interconnect	62.4	37.92	48	38.4
I/O (maximum theoretical, simplex**)	96	128	92	64

* These testbeds have only one node, therefore, network is not used in this case;

** All testbeds use PCIe interconnect, thus, for total I/O bandwidth at full-duplex simply multiply by 2.

once in one go. This has evolved in many forms, from SSE and SSE2 through AVX and AVX2 and now into AVX-512-F. Other important changes available in Intel® Xeon® Gold are presented separately in [31].

AMD Epyc™7551. This 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. The new core design can process four x86 assembler instructions per cycle and introduces Simultaneous Multithreading (SMT). Zen microarchitecture also introduces a considerable number of improvements and design changes over Bulldozer including wider instruction set, larger cache system, 2x higher bandwidth, better branch predictions, etc [3, 25].

ARM Hi1616. 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. Hi1616 features several major microarchitectural improvements in memory performance, as well as in integer and floating point arithmetics that build on top of the current generation of ARMv8-A cores [4, 20].

Power8+ S822LC. Being designed for "accelerated workloads in high-performance computing (HPC), high-performance data analytics (HPDA), enterprise data centers, and accelerated cloud deployments" [13, 14], IBM 8335 Power System S822LC for High Performance Computing server Model GTB (8335-GTB) perfectly suits for all kinds of GSS applications. S822LC brings together two POWER8 CPUs with four NVIDIA Tesla P100 GPUs through novel NVLink Technology.

1.3. Measuring and Reporting the Performance Technique

In the process of preparing and launching the benchmark, we faced a number of technical difficulties, which significantly influenced the approach we have chosen to measure and report performance. This subsection highlights our approach to tackling those difficulties.

Since the analysed testbeds represent brand-new architectures, we encountered a limited number of tools for measuring metrics of interest which were strait available for all testbeds. In particular, many popular performance measuring tools – VampirTrace, Scalasca, etc – were not ported to the ARM Hi1616 by the time of performing benchmark. Moreover, tested applications are written in different languages – C/C++, Fortran, and Python – which reduces the range of performance measuring tools suitable for all applications at once. On the other hand, our study is focused on overall performance of the code and does not require instrumentation to measure and analyse performance. Thus, we decided to omit specialised benchmarking libraries (e.g., LibSciBench [22]) and performance analysis tools (e.g., VampirTrace) in favour of standard Linux toolset available out-of-the-box. The metrics of interest were measured by means of `/usr/bin/time` Linux utility. This utility allows measuring the following metrics for the application as a whole and for each separate MPI process: total elapsed real time, the number of filesystem inputs and outputs, maximum resident set size of the process, average total (data+stack+text) memory use of the process, etc. We used measurements of the above-mentioned metrics and the data from Tab. 1 to compute dependent metrics (like TDP and TCO) and to build all charts and plots in this paper. In order to keep conditions of the experiments even, the system caches were flushed by calling `sync; echo 3 > /proc/sys/vm/drop_caches` before each experiment. All C/C++ and Fortran codes were compiled with gcc version 5.4.0 using -O3. Python 3.5.2 was used as a Python interpreter for the ABM4Py application.

In addition, we experienced issues with different orders of magnitude in elapsed times of the applications. Even though we adjusted the input files to keep the elapsed time scales closer for all application, we still had significant differences in total elapsed times between social simulation and CFD software: each individual benchmark for social simulation codes consumed less than 3 hours on any testbed, while benchmarks for some CFD application required more than 30 hours. The latter fact prevented us from performing many repetitions of time-consuming CFD application runs. More specifically, in order to reduce the number of repetitions, we started with 2 runs and repeated the experiment until the ratio of the sample standard error to the sample mean of the elapsed time was less than 5% in each test configuration. Since the number of measurements generated with this approach, it is sometimes insufficient to construct meaningful confidence interval as suggested in the performance reporting guidelines [22], Fig. 1 and 2 report sample means of the measurements without confidence interval.

Last but not least, our testbeds were limited by 1–2 nodes, which is not enough to conduct full scalability experiments with some of the selected applications. In order to overcome this obstacle, we used the Eagle cluster equipped with 50 nodes containing 2, 14-core Intel Haswell E5-2697 v3 CPUs each, as a reference testbed, where we performed tests up to reaching the scalability bound. This allowed us to get an impression on the scalability of the applications reflected in Tab. 2.

2. Benchmarking Results

Figures 1 and 2 summarise major benchmarking results for social science and CFD applications respectively. For each application, we include two plots: the first one – scalability plot –

illustrates the change in the total elapsed time with the grows of the number of MPI processes for all testbeds, whereas the second one – the metrics plot – presents all metrics measured for Intel Xeon Gold 6140 with different number of MPI processes. Intel Xeon Gold 6140 has been chosen to report details on measured metrics since it demonstrated the best performance compared to other testbeds (see Section 3). Besides the plots for Intel Xeon Gold 6140, reports [24] and [23] contain similar plots and supplementary information for the remaining testbeds. Both Fig. 1 and 2 share identical legends. Note that scalability plot for HWRF application (Fig. 2g) lacks information about ARM Hi1616 and IBM Power8+. We failed to port HWRF on those architectures.

2.1. HPC-compliant Social Simulation Software

Our benchmarks demonstrate high performance of IPF on different architectures. The application scales to a number of available cores on all testbeds in out study (see Fig. 1a). On the reference testbed, we observe the scalability bound for more than 1400 cores. Neither RAM, nor I/O of modern architectures are the limiting factors for IPF performance (see Fig. 1b). The reason of such good results is in a heavy use of highly optimised ScaLAPACK kernels in the IPF implementation. In contrast to other applications, in case of IPF, the least elapsed time is observed for AMD Epyc^{TM} 7551 testbed (see Fig. 1a).

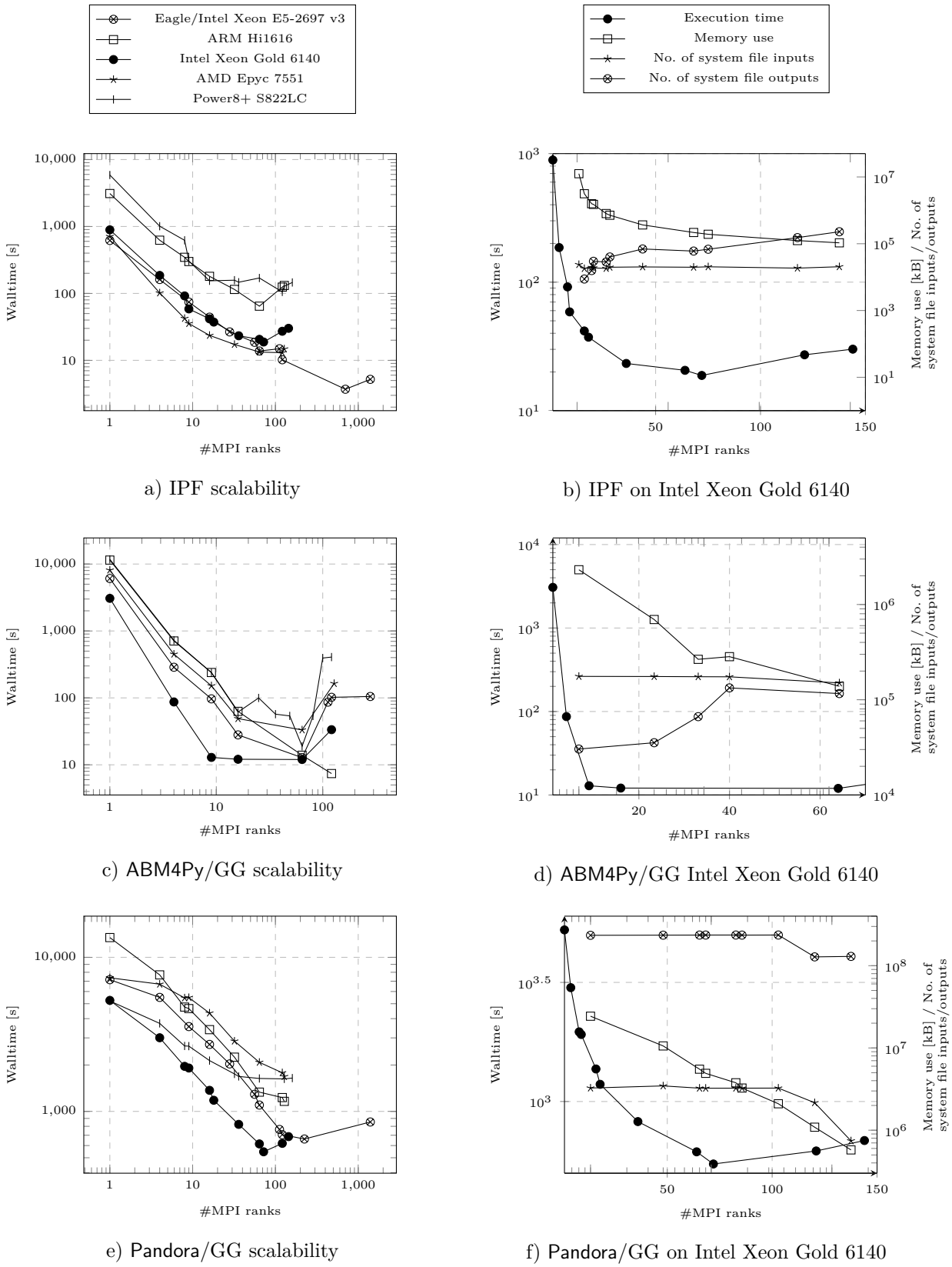
Along with IPF, we benchmarked a green growth agent-based model (ABM) of diffusion implemented in ABM4Py and Pandora frameworks.

In both cases, despite a strong difference in parallelization strategies, we observe the same pattern: ABMS applications produce a big amount of output which has a strong negative impact on application performance (see Fig. 1d and 1f). As a consequence, according to our green growth ABM, being I/O bound, current ABMS frameworks for HPC have moderate requirements to CPU performance. Nevertheless, we must emphasise that the results can look differently for more complex models with sophisticated agent activities and for simpler models which can be reduced to iterative applying of sparse matrix-vector or matrix-matrix operations (e.g., random surfer model and PageRank). Thus, our benchmarks for ABMS frameworks are not very illuminative and must be extended with more sophisticated and more simple models to provide more evidences and draw stronger conclusions. But discussion of the new representative ABMS models for benchmarking goes beyond the scope of this text.

2.2. Large-scale CFD Applications

Our measurements demonstrate that CFD applications are in general CPU-bound (see Fig. 2). Nevertheless, we observed that at some architectures, memory is also a bottleneck for some specific choices of the number of MPI processes. In particular, we noticed that OpenSWPC is memory-bound for a small number of MPI processes and CPU is bound for a large number of utilised cored, as the memory consumption monotonically decreases with the number of MPI processes in this application (see Fig. 2b). All benchmarked applications except for HWRF demonstrate the same monotonic decrease in memory consumption (see Fig. 2). At the same time, system files' outputs make a solid contribution to the total elapsed time for such applications as CCTM and CM1 (see Fig. 2d and 2f), which, in turn, imposes additional performance constraints on architectures with low I/O bandwidth.

Note: Both figures 1 and 2 share identical legends



a) IPF scalability

b) IPF on Intel Xeon Gold 6140

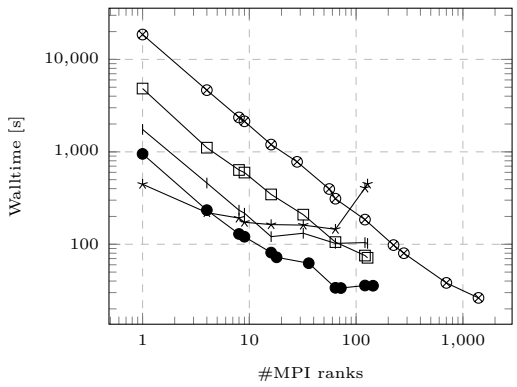
c) ABM4Py/GG scalability

d) ABM4Py/GG Intel Xeon Gold 6140

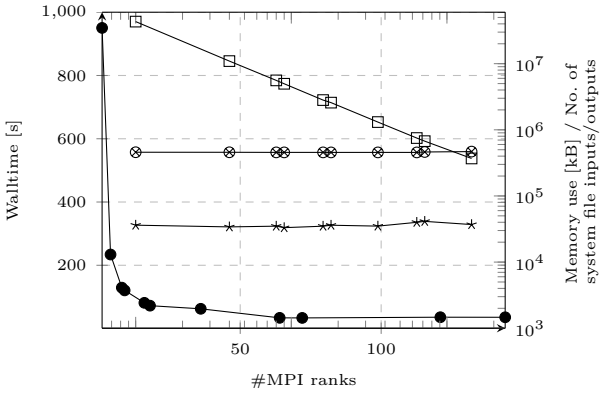
e) Pandora/GG scalability

f) Pandora/GG on Intel Xeon Gold 6140

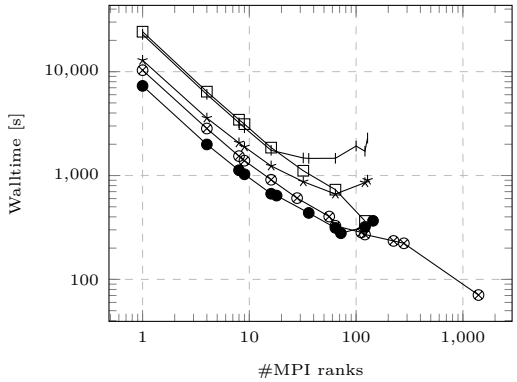
Figure 1. Results for social simulation applications



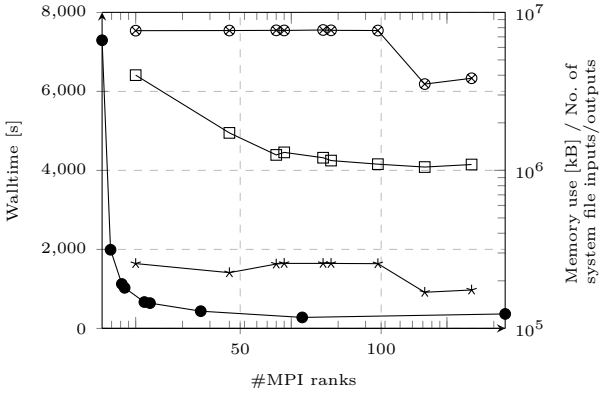
a) OpenSWPC scalability



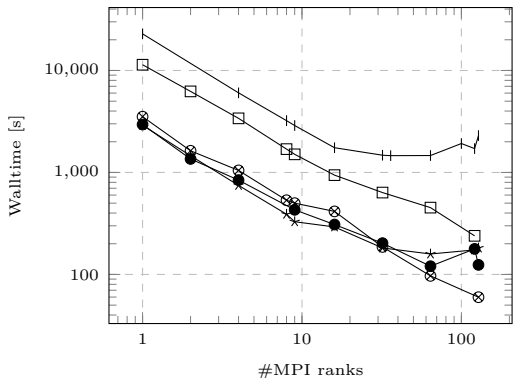
b) OpenSWPC on Intel Xeon Gold 6140



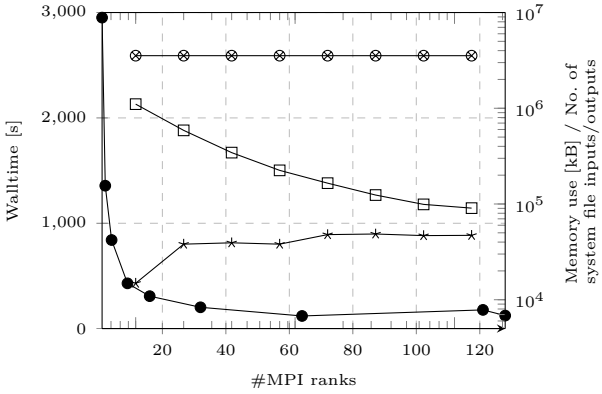
c) CMAQ/CCTM scalability



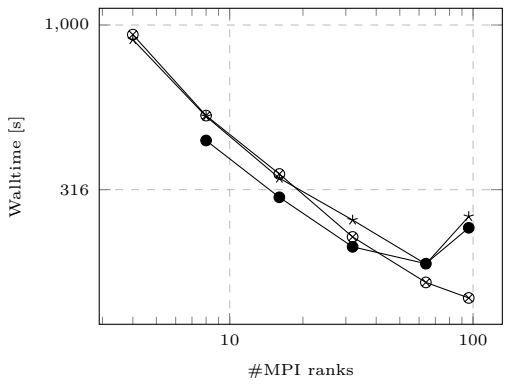
d) CMAQ/CCTM on Intel Xeon Gold 6140



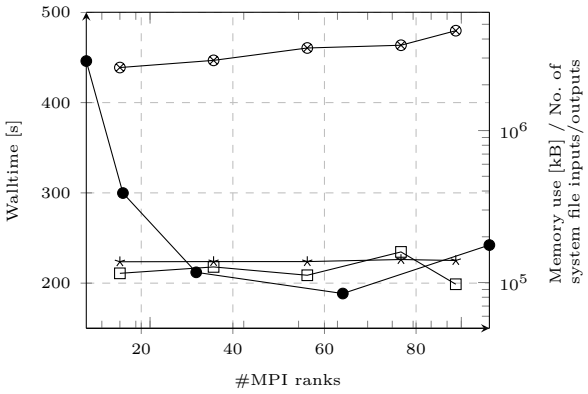
e) CM1 scalability



f) CM1 on Intel Xeon Gold 6140



g) HWRF scalability



h) HWRF on Intel® Xeon® Gold 6140

Figure 2. Results for CFD applications

Intel® Xeon® Gold 6140 provides the least elapsed time for all CFD applications from our benchmark with one minor exception (see Fig. 2). AMD Epyc™ 7551 beats Intel® Xeon® Gold 6140 in OpenSWPC for the small number of cores (see Fig. 2a). Nevertheless, performance of AMD Epyc™ 7551 degrades quickly compared to Intel® Xeon® Gold 6140 if the number of utilised cores grows.

2.3. Implications for the Hardware/Software Co-design

Table 2 shortly summarises information about scalability of the benchmarked social simulation software and about hardware bottlenecks revealed in the previous subsection. In this table, rows from group "Bottlenecks" reflect which of the following – CPU, memory consumption, system files' inputs or system files output – appeared to be limiting factors for performance of the benchmarked applications, whereas the "Scalability" row presents the maximum number of utilised CPU cores where we observed a decrease in the total elapsed time on the reference testbed – the Eagle cluster.

Table 2. Bottlenecks in the hardware and scalability of the benchmarked applications

		Social simulation software				CFD software			
		IPF	Pandora		ABM4Py	HWRF	OpenSWPC	CMAQ CCTM	CM1
			Europe	World	128x128				
Bottlenecks	CPU	✓				✓	✓	✓	✓
	RAM						✓		
	Inputs					✓			
	Outputs		✓	✓	✓			✓	✓
Scalability*		≈700	≈128	≈700	≈64	≥128	≥128	≥128	≥128

* maximum number of utilised cores of Xeon E5-2697 v3 cluster that leads to reduction of the total elapsed time.

As Tab. 2 illustrates, most of the distributed GSS applications are memory-bound. Even large-scale CFD codes can be bound by I/O and RAM under special circumstances. It allows us to conclude that the fast memory is an essential requirement to HPC clusters for GSS applications whereas high CPU clock frequency plays a less important role. Moreover, since many state-of-the-art GSS applications deal with large input and output files, we believe that GSS software developers should invest more time into design of file-avoiding applications. Our scalability tests show that hyper threading provides little performance improvements for most of the GSS applications. Therefore, it makes little sense to invest money in expensive massively multithreaded chips (like Power8) for GSS users. We also recommend avoiding clusters with GPU accelerated nodes since only a few popular GSS applications benefit from GPUs. In particular, among widely used general-purpose ABMS frameworks and problem-specific ABMS codes for HPCs, only the FLAME-GPU (Flexible Larg-scale Agent Modelling Environment) framework utilises GPUs [26, 27]. Seldom use of GPUs is also partially related to the fact that most social science applications are memory-bound. Being more specific, among the architectures used in benchmarking, we recommend to build clusters upon ARM Hi1616 in case that energy efficiency is a crucial requirement, or upon Intel® Xeon® Gold 6140 in case that performance is a first priority while relatively high operating expense and capital expenditure are not an issue.

According to our benchmarks, the scalability of GSS applications is rather diverse. All applications from the social simulation software stack demonstrate poor scalability with one notable exception – the IPF implementation. Moreover, even though our benchmarks do not demonstrate this explicitly, it is also known that social simulation software scales are worse than the large-scale

CFD codes. On the other hand, due to stochastic nature of ABMs, a typical social simulation workflow assumes many simultaneous simulation runs, whereas the fitting step in reconstruction of a synthetic population should normally be performed only once for a given dataset. Therefore, the optimal number of nodes for the state-of-the-art should be defined by scalability of the synthetic population and CFD codes (if the latter ones are of interest for the target GSS audience). We can always bypass the gap in scalability of the synthetic population and ABMS codes and reach full utilisation of clusters by making several simultaneous simulation runs (and treating simulation results in a file-avoiding way).

Unfortunately, our results do not allow for drawing solid conclusions about node interconnections since the benchmarks had been done on the testbeds with only one or two nodes.

3. Discussion

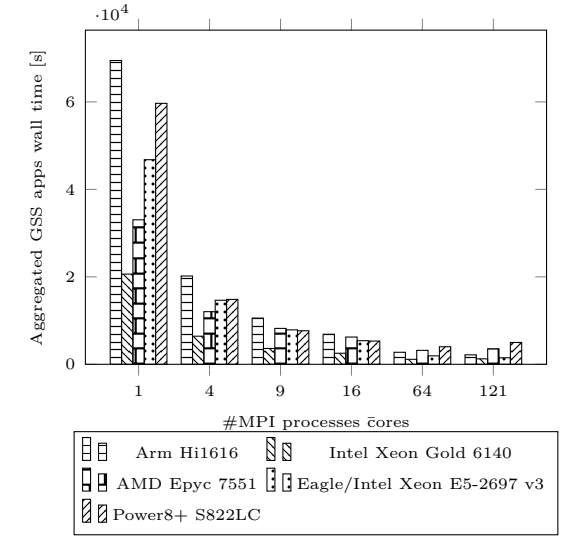
The findings of this work allowed the authors to formulate the following conclusions:

- Among all tested applications, IPF is the least I/O demanding. For the reference architecture it shows, the scalability is up to 700 cores. On the other hand, other selected processors scale in the range of a number of physical cores, so we expect that using them in a multinode configuration will result in a behaviour similar to the reference testbed.
- Green Growth-pilot applications are dominated by I/O operations (mostly output) where a large HDF5 file is created and to which all processes save data;
- Results obtained for ABMS4py – another social simulation application-prove that it should be subject to major improvements. For instance, the best execution time on Intel® Xeon® 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)
- 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 the number of cores used until the maximum number of 1400 is used. Other processors show similar behaviour. Using hyper threads (where possible) does not provide any further time improvements.
- CCTM is I/O dominated (especially output) application and, thus, the impact on the execution times is high
- 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.
- HWRF demonstrates very similar results to CM1 in terms of scalability on processors equipped with the implementation of simultaneous multithreading
- Best execution time gets 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^{TM}, 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.

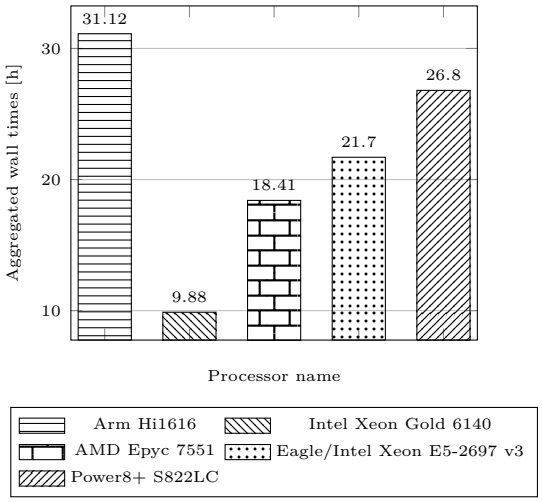
For the final comparative analysis, two additional metrics have been introduced:

- *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.

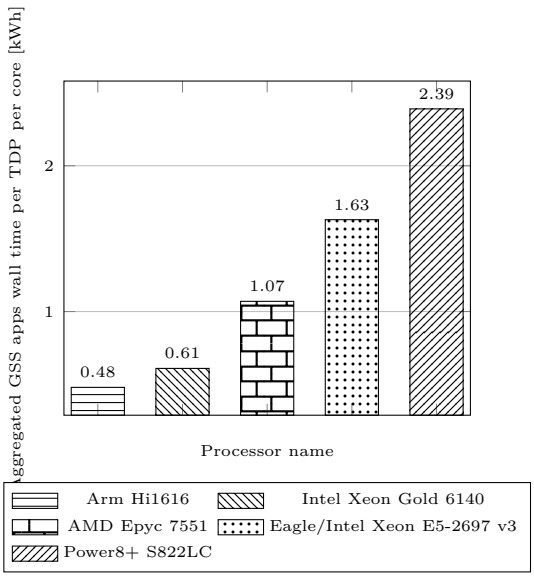
- *Cost efficiency* using scaled timing results by the cores price falling on the given number of cores (cores price is calculated by dividing processor price by a given number of processor cores).



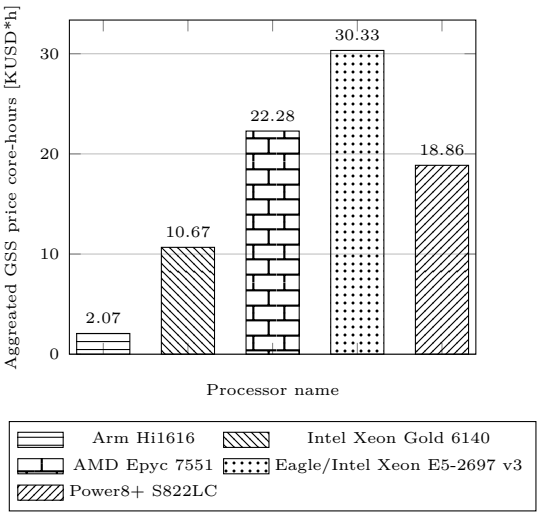
a) Ranking of architectures across the number of cores (processes)



b) Aggregated GSS benchmark walltimes



c) Ranking of architectures based on estimated energy consumption (total power needed by CPU to finish all tests) - the lower the better



d) Ranking of architectures based on cost efficiency

Figure 3. Benchmarking summary results

In the scope of *aggregated execution times* for the given number of cores, it can be noted that among the whole range of the number of cores, the winner is Intel® Xeon® Gold 6140 (Fig. 3a). Surprisingly, the AMD Epyc™ is slower than the ARM Hi1616 when 64 or more cores are used (it is mostly because hyperthreading degrades the performance in some cases) and is also slower than the reference Intel®Xeon® E5-2697 when 9 or more cores are used.

By looking at the aggregated execution time across all tested applications, the winner turned out to be Intel® Xeon® Gold 6140 (Fig. 3b).

For the *estimated cumulative energy consumption* (calculated as a sum of walltimes and TDP per used cores products expressed in kilowatt-hours) metric, the winning processor is ARM Hi1616, representing the current tendency in HPC, where attention generally speaking is turned to energy-efficient technologies. The second-place holder is Intel®Xeon® Gold 6140 and Power8+ brings up the rear (Fig. 3c).

The estimated power consumption chart is a good point of view when talking about green HPC computing. The presented results are not exact because they are only dictated by the estimated values based on the CPU's TDP. Nevertheless, assuming the fact that almost all architectures use the same memory model - DDR4, - it can be considered that most of the energy consumed is the energy utilised by the processor. In this scope, the best energy consumption ratio is characterised by the ARM architecture, which is absolutely designed for energy-saving solutions which are 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™ demonstrated 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.

Another valuable finding is based on GSS cost efficiency analysis (Fig. 3d). ARM Hi1616 turned out to be approximately 9 times better than for Power8+ (equipped with DDR3, NVLinks were not utilised) and 15 times better than the reference Intel® Xeon® E5-2697 processor, mostly due to its small number of expensive cores and relatively average timing results. Additionally, when talking about general processor 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® E5-2697, 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 a relatively small amount of output. This is the case of 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 hyper threading is not utilised properly. At the same time, it is quite often that 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.

When analysed simultaneously, all the abovementioned results proved that the most promising ARM processor in the context of cost and energy consumption is the slowest one (mostly due to the low clock frequency). Other competitor, Intel® Xeon® Gold 6140, is 5 times less cost-efficient for GSS benchmark and slightly worse in energy consumption but it is approximately 3 times quicker regarding the aggregated execution time. In other words, the future HPC investors with the above information in place have the ability to decide which direction to follow: whether to reach high compute intensity, minimise costs, or try to find the golden middle.

Table 3 presents the summarised results for each individual architecture in three separate domains: walltime, energy efficiency and cost efficiency. The numbers reflect the weighted points referring to the overall applications results in a given category (in this case, the less the better). In general, the most promising processor is Intel® Xeon® Gold 6140 but those individuals for whom the cost and environmental aspects are the most important should look closely at ARM Hi1616.

Table 3. Ranking of all tested architectures (*the less the better*)

	Walltime	**Energy efficiency**	**Cost efficiency**
ARM Hi1616	1.0	0.2	0.1
Intel® Xeon® Gold 6140	0.3	0.3	0.4
AMD Epyc™ 7551	0.6	0.4	0.7
Intel® Xeon® E5-2697 v3	0.7	0.7	1.0
Power8+ S822LC	0.9	1.0	0.6

In general, it can be said, the most promising processor is Intel® Xeon® Gold 6140 but those individuals for whom the cost and environmental aspects are the most important should look closely at ARM Hi1616.

Conslusions and Future Work

The proposed benchmark gives a good evaluation tool for a relatively automatic way of proceeding with tests and receiving results which will directly allow using the best HPC architecture. It means that the end user or the resource owner may finally have different criteria to fulfil their requirements. The resource owner will focus on parameters which are globally efficient (all applications running on the machine), cost-efficient (TCO shown as CAPEX and OPEX, i.e. the investment costs vs. maintenance costs of the HPC). The end user will, however, concentrate on the fastest way to receive results and the most efficient way of parallelisation.

From that point of view, the benchmark could be extended by testing the scalability of the e-Infrastructure and the energy consumption of the running benchmark automatically. A final step of the benchmark could interpret the results for both groups of users and propose the best HPC system in terms of size and architecture (CPU, memory size, aggregated speed to external memory, if necessary).

Acknowledgements

This work has been supported by the CoeGSS (Centre of Excellence for Global System Science) project and has been partly funded by the European Commission's ICT activity of the H2020 Programme under grant agreement number: 676547.

References

1. EPA: Cmaq: The community multiscale air quality modeling system. `https://www.cmascenter.org/cmaq`, accessed: 2018-09-21

2. DTC: The users page on hurricane wrf. `https://dtcenter.org/HurrWRF/users`, accessed: 2019-05-30

3. AMD: AMD EPYC 7000 series processors: Leading performance for the Cloud era. https://www.amd.com/system/files/2017-06/AMD-EPYC-Data-Sheet.pdf (2019), accessed: 2019-05-28

4. Guo, X., Morales, C., Saastad, O.W., Shamakina, A., Rijks, W., Weinberg, V.: Best practice guide – ARM64. http://www.prace-ri.eu/best-practice-guide-arm64/ (2019), accessed: 2019-05-28

5. Cmaq inputs and test case data. https://www.epa.gov/cmaq/cmaq-inputs-and-test-case-data (2018), accessed: 2018-09-21

6. Yule, G.G.U.: On the methods of measuring association between two attributes. Journal of the Royal Statistical Society 75(6), 579–652 (1912)

7. Wolf, S., Fuerst, S., Geiges, A., Steudle, G.A., von Postel, J., Jaeger, C.C.: Electric mobility in view of green growth. https://globalclimateforum.org/wp-content/uploads/2018/12/GCFWorkingPaper3-2017.pdf (2017), accessed: 2019-05-28

8. Maeda, T., Takemura, S., Furumura, T.: OpenSWPC: An open-source integrated parallel simulation code for modeling seismic wave propagation in 3D heterogeneous viscoelastic media. Earth Planets Space 69(102) (2017), DOI: 10.1186/s40623-017-0687-2

9. Bryan, G.: Cm1 homepage. http://www2.mmm.ucar.edu/people/bryan/cm1, accessed: 2019-05-30

10. Maeda, T.: OpenSWPC - an open-source seismic wave propagation code. https://github.com/takuto-maeda/OpenSWPC (2018), accessed: 2018-09-17

11. Bryan, G.H.: The governing equations for CM1 (2013)

12. Rubio-Campillo, X.: C++/Python Agent-Based Modelling framework for large-scale distributed simulations. https://github.com/xrubio/pandora, accessed: 2018-09-17

13. Bicas Caldeira, A., Haug, V., Vetter, S.: IBM power systems S822LC for high performance computing: Introduction and technical overview. https://www.redbooks.ibm.com/redpapers/pdfs/redp5405.pdf (2016), accessed: 2019-05-28

14. Bicas Caldeira, A., Kahle, M.E., Saverimuthu, G., Vearner, K.: IBM power systems S822LC: Technical overview and introduction. https://www.redbooks.ibm.com/redpapers/pdfs/redp5283.pdf (2015), accessed: 2019-05-28

15. Chemel, C., Fisher, B., Kong, X., Francis, X., Sokhi, R., Good, N., Collins, W., Folberth, G.: Application of chemical transport model CMAQ to policy decisions regarding PM2.5 in the UK. Atmospheric Environment 82, 410 – 417 (2014), DOI: 10.1016/j.atmosenv.2013.10.001

16. Biswas, M.K., Bernardet, L., Ginis, I., Kwon, Y., Liu, Q., Marchok, T., Sheinin, D., Tallapragada, V., Thomas, B., Tong, M., Trahan, S., Wang, W., Yablonsky, R., Zhang, X.: Hurricane weather research and forecasting (HWRF) model: 2017 scientific documentation (2018), DOI: 10.5065/D6MK6BPR, accessed: 2019-05-28

17. Geiges, A., Wolf, S., Steudle, G., Fuerst, S.: Report of framework for prototyping of parallel agent based modelling systems. http://coegss.eu/wp-content/uploads/2018/11/D3.8.pdf (2018), accessed: 2019-05-28

18. Swarup, S., Marathe, M.V.: Generating synthetic populations for social modeling. http://people.virginia.edu/~ss7rs/synthetic_population_tutorial_2/slides.php (2017), accessed: 2019-05-28

19. CoeGSS-Project: Agent-based modelling framework for python. https://github.com/CoeGSS-Project/abm4py, accessed: 2019-05-30

20. Newburn, C.J., Abdurachmanov, D., Kaplan, L., McIntosh-Smith, S., McLean, M., Summoto, S., Van Hensbergen, E., Vergara Larrea, V.: The ARM software ecosystem: Are we there yet? https://arm-hpc.gitlab.io/presentations/SC17-Arm-Ecosystem.pdf (2017), accessed: 2019-05-28

21. Epstein, J.M., Pankajakshan, R., Hammond, R.A.: Combining computational fluid dynamics and agent-based modeling: A new approach to evacuation planning. PLOS ONE 6(5), 1–5 (2011), DOI: 10.1371/journal.pone.0020139

22. Hoefler, T., Belli, R.: Scientific benchmarking of parallel computing systems: Twelve ways to tell the masses when reporting performance results. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. pp. 73:1–73:12. SC '15, ACM, New York, NY, USA (2015), DOI: 10.1145/2807591.2807644

23. Gienger, M., Gogolenko, S., Geiges, A., Kaliszan, D., Petruczynik, S., Januszewski, R., Wolniewicz, P.: Second report on provided testbed components for running services and pilots. http://coegss.eu/wp-content/uploads/2018/11/D5.8.pdf (2018), accessed: 2019-05-28

24. Gienger, M., Petruczynik, S., Januszewski, R., Kaliszan, D., Gogolenko, S., Fuerst, S., Palka, M., Ubaldi, E.: First report on provided testbed components for running services and pilots. http://coegss.eu/wp-content/uploads/2018/02/D5.7.pdf (2017), accessed: 2019-05-28

25. Wikichip: Hi1616 - HiSilicon. https://en.wikichip.org/wiki/hisilicon/hi16xx/hi1616, accessed: 2019-05-28

26. Richmond, P.: FLAME GPU: Technical report and user guide. https://media.readthedocs.org/pdf/flamegpu/latest/flamegpu.pdf (2018), accessed: 2019-05-28

27. Richmond, P., Romano, D.: Template-Driven Agent-Based Modeling and Simulation with CUDA, pp. 313–324. GPU Computing Gems Emerald Edition, Elsevier (2011), DOI: 10.1016/b978-0-12-384988-5.00021-8

28. Family 8335+03 IBM power system S822LC for high performance computing. https://ibm.co/2cMMb8B (2018), accessed: 2018-09-18

29. Gogolenko, S.: Software for agent based social simulation with raster inputs in distributed HPC environments. https://fs.hlrs.de/projects/teraflop/26thWorkshop_talks/WSSP26-24_Gogolenko.pdf (2017), accessed: 2019-05-28

30. Suleimenova, D., Bell, D., Groen, D.: A generalized simulation development approach for predicting refugee destinations. Scientific Reports 7(1), 13377 (2017), DOI: 10.1038/s41598-017-13828-9

31. Intel®Xeon®Gold 6140 processor specification. https://ark.intel.com/products/120485/Intel-Xeon-Gold-6140-Processor-24_75M-Cache-2_30-GHz (2018), accessed: 2018-09-18