First Experiences in Performance Benchmarking with the New SPEChpc 2021 Suites

Holger Brunst[‡], Sunita Chandrasekaran[*], Florina M. Ciorba[†], Nick Hagerty[‖], Robert Henschel[††], Guido Juckeland[¶], Junjie Li[**], Verónica G. Melesse Vergara[‖], Sandra Wienke[§], and Miguel Zavala[*],
[*] University of Delaware, Newark, DE, USA. [†]University of Basel, Basel, Switzerland
[‡]Technische Universität Dresden, Dresden, Germany. [§]RWTH Aachen University, Aachen, Germany
[¶]Helmholtz-Zentrum Dresden-Rossendorf, Dresden, Germany. [‖]Oak Ridge National Laboratory, Oak Ridge, TN, USA
[**] Texas Advanced Computing Center, The University of Texas at Austin, Austin, TX, USA.
[††]Indiana University, Bloomington, IN, USA

2022 22nd International Symposium on Cluster, Cloud and Internet Computing (CCGrid) | 978-1-6654-9956-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/CCGRID54584.2022.00077

***Abstract*—Modern High Performance Computing (HPC) systems are built with innovative system architectures and novel programming models to further push the speed limit of computing. The increased complexity poses challenges for performance portability and performance evaluation. The Standard Performance Evaluation Corporation (SPEC) has a long history of producing industry-standard benchmarks for modern computer systems. SPEC's newly released SPEChpc 2021 benchmark suites, developed by the High Performance Group, are a bold attempt to provide a fair and objective benchmarking tool designed for state-of-the-art HPC systems. With the support of multiple host and accelerator programming models, the suites are portable across both homogeneous and heterogeneous architectures. Different workloads are developed to fit system sizes ranging from a few compute nodes to a few hundred compute nodes. In this work we present our first experiences in performance benchmarking the new SPEChpc2021 suites and evaluate their portability and basic performance characteristics on various popular and emerging HPC architectures, including x86 CPU, NVIDIA GPU, and AMD GPU. This study provides a first-hand experience of executing the SPEChpc 2021 suites at scale on production HPC systems, discusses real-world use cases, and serves as an initial guideline for using the benchmark suites.**

***Index Terms*—HPC, SPEC, HPG, SPEChpc 2021, benchmarks, performance benchmarking and analysis, heterogeneity, offloading, MPI, MPI+X, OpenMP, OpenACC**

I. Introduction

Evaluating the performance of computing systems using carefully designed benchmarks supports comparisons between different systems. Performance benchmarks have contributed to improvements in successive generations of systems, which are important for pushing the speed limit of computing, purchasing investments, development of research software and performance analysis tools, system maintenance, and others.

Authors listed in alphabetical order. Following [1] to list contributions: **H. Brunst** (Investigation, Writing - Review & Editing), **S. Chandrasekaran** (Conceptualization, Investigation, Supervision, Project administration, Writing - Original Draft), **F. M. Ciorba** (Methodology, Writing - Review & Editing), **N. Hagerty** (Software, Writing - Review & Editing), **R. Henschel** (Writing - Review & Editing), **G. Juckeland** (Data Curation, Writing - Review & Editing), **J. Li** (Software, Investigation, Writing - Original Draft), **V. Vergara** (Software, Investigation, Writing - Review & Editing), **S. Wienke** (Investigation, Writing - Original Draft), **M. Zavala** (Software)

The SPEC High Performance Group (HPG) [2] has been designing benchmarks for the last three decades, releasing the first benchmark suite, SPEC HPC96, in 1996. Over the years, SPEC HPG released various benchmark suites that target all parallel execution layers of modern HPC systems. Nevertheless, each suite focused on individual parallelism layers: SPEC MPI2007 on inter-node communication, SPEC OMP2012 on intra-node CPU parallelism, and SPEC ACCEL 2017 on the performance of accelerator devices.

Motivation. The design and release of SPEC ACCEL raised the question as to *how to measure performance of a system with multiple accelerator devices*. To answer this question, contributors of SPEC HPG discussed how to better reflect ***overall system performance***, specifically considering the increasing heterogeneity in system architectures and diversity in programming models. The outcome materialized as the launch of a new application search program [3] in late 2017 to gather input from the HPC community on potential benchmark applications that are, among others, characterized by *more than one form of parallelism* (cross-node, node-level, with/out accelerator offloading). The SPEChpc 2021 benchmark suites are comprised of those applications that fulfill the selection criteria (see Section III).

Most of the existing benchmarks either focus on low-level hardware performance features and provide microbenchmarks and small application kernels, or on higher level code features and provide mini- or proxy apps selected to prepare the hardware and software stacks of upcoming large systems. In contrast, SPEChpc 2021 comprises real-world applications solicited from the broader HPC community, provides a set of execution and reporting rules, and adopts a peer-review process before publishing benchmarking results online.

To the best of our knowledge, SPEChpc 2021 is a *one of a kind benchmark suite* that offers a harness to handle the process from installation to ensuring the correctness of the results and providing a performance score (called the SPEC score) to enable ranking, that explores hybrid programming models as well as MPI-only, and facilitates benchmarking on university clusters and large HPC center systems.

Contributions. This work brings forward the following

978-1-6654-9956-9/22/$31.00 ©2022 IEEE
DOI 10.1109/CCGrid54584.2022.00077

contributions: (1) Presents the new SPEChpc 2021 benchmark suites, including code statistics, instruction mix, MPI call percentages, and roofline models. (2) Provides a first study of how well the suites meet the search requirements by evaluating the performance results on a variety of hardware systems as well as exploring different system configurations. To the best of our knowledge, this work is among the first few, with respect to evaluating scientific applications on a pre-exascale system, namely Spock, equipped with AMD MI100 GPUs. (3) Compares the performance of employing MPI+X programming models with various X, namely OpenMP host, OpenACC and OpenMP target offloading.

Impact. The significance and impact of this performance benchmarking study are multi-fold: (i) Added value through extensive testing on a wide range of platforms, going beyond classical building and compilation or error testing. (ii) The performance numbers obtained across devices allow identifying unsuspected system configuration bugs. (iii) Engaged the HPC community by enlisting the help of testers for the beta release candidate of the suites.

II. Motivation

HPC application benchmarks are of value for researchers and system managers who have used previous SPEC HPG benchmark suites, such as SPEC MPI, SPEC OMP, or SPEC ACCEL, in various scenarios. The new SPEChpc 2021 benchmark suites also encourage similar uses.

Procurement: HPC system procurement is an important flagship. During the preparation of the tendering documents, managers compare SPEC scores across different hardware and software setups, available as submitted results online [4]. This helps define limits and thresholds in the tendering documents. Given SPEC's application benchmark characteristics, the benchmark suites are highly valuable to extrapolate performance of complex scientific applications for various and especially future hardware architectures. To this end, it has been beneficial to further integrate SPEC scores into acceptance tests for HPC system procurement. vendors need to demonstrate that their products deliver high performance with scientific applications and not just on highly-optimized microbenchmarks. In the past, numerous organizations have extensively used SPEC HPG benchmarks for procurement purposes.

Research Software: Academic HPC systems provide users with a varied HPC-based software stack, including research compilers and runtime systems. Academic researchers use the SPEC benchmarks to analyze whether the software stack is mature enough to compile and correctly execute these applications (using the verification feature of the SPEC harness), and whether the research compilers deliver high performance [5]–[10]. The SPEC HPG benchmarks have also been used during the prototype implementation of OMPT in LLVM's OpenMP runtime as part of the OpenMP tools committee work before the final specification of OMPT [11].

Performance Analysis of Tools: Development of software tools, such as performance analysis tools for parallel programs, is another HPC-related activity with high relevance for academic HPC systems. The SPEC HPG benchmark suites are often used to measure the overhead of such tools, i.e., executing the SPEC benchmarks with and without the HPC software tool under development. For example, the MUST tool [12] provides runtime correctness and deadlock analysis of parallel programs. MUST has been evaluated with SPEC MPI L2007 v2 to assess its overhead and the influence of specific changes in the tool infrastructure. Similarly, specific parts of tools can be assessed using SPEC benchmarks, such as the OpenMP measurement adapters of Score-P [13], an instrumentation and measurement infrastructure for profiling and event tracing. The SPEC OMP benchmark suite has also been used to evaluate the existing Opari2 adapter against a prototype measurement adapter based on OMPT [14].

System Regression Testing: HPC centers perform regular systems maintenance. This includes security and software updates as well as performance optimizations. In the past, RWTH Aachen and TU Dresden observed that performance-relevant changes and errors arise unintentionally during this process. Regression tests following maintenance intervals with well-defined SPEC benchmarks make it possible to detect such unintentional changes in the system. The exact same application scenario is executed regularly, and the results are automatically compared against results from previous measurements. The development and testing of the SPEChpc 2021 benchmark suites help identify non-performing HPC nodes among other issues. To elaborate further, we present two motivational scenarios at sites RWTH Aachen and TU Dresden.

Case Study 1: At RWTH Aachen University, tests with the new SPEChpc 2021 benchmark suites (*medium* suite) on 50 compute nodes of the system showed significant negative performance differences for some of the benchmarks compared to other HPC systems with a very similar setup, e.g., available as SPEC results [4]. Via a deep dive into the performance data, the execution times were found to differ mostly in the amount of MPI time, specifically, in MPI_Allreduce collective operations. This cross-node execution time imbalance is caused by: (1) dropping memory bandwidth and (2) noise (which leads to desynchronization [15]). While the memory DIMMs did not completely malfunction and were not detected in the system's health check, some of them delivered roughly 20 % less bandwidth than expected. Hence, additional job-based bandwidth checks have been implemented and are regularly reported to the vendor responsible for replacing those DIMMs. Although (system and network) noise is a known issue, its high impact for the worst-case scenarios as triggered by the SPEC benchmarks was surprising, underscoring the usefulness of the new benchmark suites. To improve and achieve comparable performance for such bulk-synchronous parallel programs using MPI collectives, users at RWTH Aachen University will now be advised to leave one core empty per NUMA domain per compute node instead of fully occupying the nodes with MPI processes. Given these important findings, the performance of other applications will also be investigated

and, if necessary, improved.

Case Study 2: At TU Dresden, several undetected system problems were discovered on the TU Dresden system, even though system health checks were implemented. The SPEChpc 2021 benchmarks revealed significant performance variations when executed multiple times across a changing subset of equal nodes of the system. For example the tealeaf benchmark is very sensitive to process scheduling due to its intensive use of collective MPI operations. We observed a performance degradation of a factor of 2. The reasonable cases had an MPI_Allreduce time contribution of 7% and the bad cases had a time contribution of 52%. These cases led to the kernel bug (referring to (2) below) that stole 50% of cycles of a small group of pinned MPI processess thereby slowing down all the others. The suites have shown its high suitability to detect system problems (by means of comparison with reference data) that did not lead to a partial or complete failure of the system, but slowed down a subset of nodes significantly. Once the partial slow downs were recognized as such, their causes were found quickly to be: (1) a faulty BIOS configuration of a number of computing nodes, (2) a kernel bug [16] occurring infrequently, and (3) an unfavorable configuration of the SLURM daemon. The commonality between all these issues is that no crashes occurred and the entire system was 100% available at all times. Nevertheless, benchmarks from the SPEChpc 2021 suites showed performance degradation of up to 50% in the BIOS setup and kernel bug cases. While the degradation must have occurred due to one or two application, it still shows how the SPEChpc2021 suites were able to highlight those. It is pertinent to have a good/right mix of applications in a suite that can highlight such issues in the system. The above-mentioned kernel bug, was discovered due to the fact that the execution time of the Tealeaf benchmark (see Section III) occasionally doubled inexplicably, even though the CPU resource configuration was unaltered. Increasing the number of compute nodes also increased the frequency of observation of this phenomenon.

A profiler-assisted [13] analysis of the core cycle counters over time revealed that a very small random fraction of MPI ranks were only processed at half speed because they received only half of the theoretically possible processor cycles. This in turn led to the load imbalance at execution time of the actually very well statically balanced solver. Deploying an additional system profiler [17] revealed that errant kernel threads on the affected nodes were responsible for the absence of CPU cycles. The root cause was determined to be a missing kernel patch [18], now installed.

III. Overview of SPEChpc 2021 benchmark suites

The search program [3] for SPEChpc 2021 benchmarks from 2017 gathered applications from the HPC community with the following characteristics:

- **Support for MPI+X parallelism**, where X takes one of three values: OpenMP host (denoted as OMP), OpenMP target (denoted as TGT), and/or OpenACC (denoted as ACC). They offer hybrid execution using all potentially available parallelism within and across compute nodes. An *MPI-only* version of the application is needed for baseline comparisons.
- **Origin in various science domains** to reflect the diversity of typical and real HPC workloads.
- **Fortran or C/C++** programming language.
- **Support for strong scaling** of work distribution for multiple data set sizes, as one SPEC suite will always distribute the same workload of a benchmark regardless of the actual number of process/threads used.
- **Predictable code paths** with no algorithmic differences depending on the computing platform
- **Limited time spent in I/O** as this not an I/O benchmark.
- **Numerically verifiable output** to check for correctness within a definable margin of error.

A. SPEChpc 2021 Benchmark Composition

The first official release (version 1.0.3) of the SPEChpc 2021 suites consists of *nine* applications [4]. The applications and their basic properties are summarized in Table I. Not all application benchmarks are included in all the suites. This is the case when the maximum problem size that represents a realistic problem has already been reached in a smaller suite or when the problem does not naturally scale to all suites (see Section III-B). Other applications did not support all three levels of intra-node parallelism, namely 'X' in MPI+X, before submission in response to the search program. During the benchmark preparation phase, those have been refactored accordingly to support the three 'Xs'.

B. SPEChpc 2021 Suites and Metrics

The SPEChpc 2021 benchmark suites support *strong scaling* workloads by offering **four suites**: ***tiny***, ***small***, ***medium***, and ***large***, which represent common workload sizes for all benchmarks in one suite. Thus, the SPEC performance measurement harness (Section IV) can be used to execute and verify the results where everyone solves the same problem(s).

The **SPEC score** is the ratio of the execution time of the benchmarks on the reference system (RS) to the execution time on the system under test (SUT) that is also reported in the public SPEC results repository [4].

The maximum memory requirements for each workload size is defined to reflect typical HPC system sizes. The benchmarks have also been tested with MPI rank counts typical of these system sizes. Due to communication buffers, the maximum memory requirements can easily be exceeded for very large MPI rank counts. All these design limits are shown in Table II. Each suite is assigned a prefix number (Table II) and each benchmark a postfix number (Table I). For example, **5**05 denotes the LBM benchmark in the ***tiny*** suite.

IV. SPEC harness

The SPEChpc 2021 suites reside within the SPEC harness which has been maintained by SPEC for more than 15 years. Users of the popular SPEC CPU benchmarks [19], [20] will be able to readily use SPEChpc 2021 benchmarks as well,

TABLE I: SPEChpc 2021 benchmark application properties

Name	Application Area	Language	Suite	approx. # LOC	# MPI calls	# OMP dir.	# ACC dir.
LBM D2Q37 (x05)	Computational Fluid Dynamics	C	T/S/M/L	9,000	118	50	66
SOMA (x13)	Physics / Polymeric Systems	C	T/S/-/-	9,500	90	192	185
Tealeaf (x18)	Physics, High Energy Physics	C	T/S/M/L	5,400	22	86	40
Cloverleaf (x19)	Physics, High Energy Physics	Fortran	T/S/M/L	12,500	23	827	886
Minisweep (x21)	Nuclear Engineering, Radiation Transport	C	T/S/-/-	17,500	41	39	43
POT3D (x28)	Solar Physics	Fortran	T/S/M/L	(incl. HDF5) 495,000	88	124	77
SPH-EXA (x32)	Astrophysics and Cosmology	C++14	T/S/-/-	3,400	82	36	18
HPGMG-FV (x34)	Cosmology, Astrophysics, Combustion	C	T/S/M/L	16,700	53	206	127
miniWeather (x35)	Weather	Fortran	T/S/M/L	1,100	11	36	20

TABLE II: Design limits for the SPEChpc 2021 suites

Suites	max memory	min ranks	max ranks
tiny (T) (5xx)	64 GB	1	256
small (S) (6xx)	480 GB	64	1,024
medium (M) (7xx)	4 TB	256	4,096
large (L) (8xx)	14.5 TB	2,048	32,768

TABLE III: Overview of the experimental setup

System	Dominant chips	Parallelism	Suite	min | max ranks
Frontera	Intel Xeon Platinum 8280	MPI-only, MPI+OMP	T, S, M, L	56 | 57,344
JUWELS Booster	NVIDIA A100	MPI+ACC, MPI+TGT	M, L	100 | 1,400
Spock	AMD MI100	MPI+TGT	S	16 | 32
Summit	NVIDIA V100	MPI+ACC, MPI+TGT	M, L	1,050 | 16,800

TABLE IV: Execution Time of the *Large* Suite on Frontera* (in seconds)

Benchmark	140 nodes	256 nodes	384 nodes	512 nodes	1024 nodes
805.lbm_l	998.6	517.1	391.2	265.9	149.6
818.tealeaf_l	828.1	448.2	298.6	223.7	123.6
819.clvleaf_l	1113.2	612.9	405.8	304.8	172.2
828.pot3d_l	2593.0	1497.2	984.7	716.1	382.4
834.hpgmgfv_l	1045.4	596.9	423.8	305.6	183.9
835.weather_l	1207.5	644.2	408.9	271.3	122.1

*Results collected on Frontera for MPI+OpenMP executions.

as the harness used with both is the same and its usage very similar. The SPEC harness is involved in all aspects of running the benchmark, from installation, correctness to submission and publication of results [4]. It can be tuned to all usage scenarios, supports different compilers and compiler environments, as well as batch systems, code verification and ensures code source integrity. The harness supports publishing of benchmark results that include **all** the information required to reproduce the benchmark results. Therefore, the harness promotes performance reproducibility, which is of increasing importance to the HPC community.

V. Results

We present here the results of executing the SPEChpc 2021 suites on four HPC systems: (a) Frontera at Texas Advanced Computing Center (TACC) [21], (b) Summit [22] at ORNL, (c) JUWELS Booster module at Forschungszentrum Jülich [23], and (d) Spock [24], a pre-exscale system also at ORNL. We have chosen these systems to show results and findings on a range of homogeneous (Frontera Intel Cascade Lake Xeon) and heterogeneous (Summit NVIDIA V100, JUWELS NVIDIA A100 and Spock AMD MI100) systems utilizing **MPI-only** and **MPI+X** programming paradigms. All experiments used 1 MPI rank/GPU.

We present **strong scaling** results. The SPEC benchmark suites are traditionally evaluated using a SPEC score (defined in Section IV) which facilitates comparison between systems. Given that performance comparisons of systems falls outside the scope of this work, we will report the traditional execution time in seconds. Table III describes the experimental setup. We have also populated Zenodo [25] (a general purpose open-access repository to share and maintain data) with performance data that was generated to build the plots and tables in this manuscript. The plots in Zenodo ishow that the comparisons may be interpreted in different ways drawing ambiguous conclusions, hence we do not focus on them in this paper but leave the interpretation of the comparisons to the different stakeholders including the procurement managers, HPC system operators, tools developers and application developers.

A. Performance Results on Frontera

Experimental setup. Frontera consists of 8,368 Cascade Lake based Dell PowerEdge C6420 compute nodes with dual socket Intel Xeon Platinum 8280 28-core CPU and 192 GB DDR4 memory. The system's interconnection network is a fat tree topology with a blocking factor of 22:18. The interconnect is Infiniband HDR technology with full HDR connectivity between switches and HDR100 connectivity to the compute nodes. Since this is a CPU-only platform, MPI and MPI+OMP are the programming models used. For MPI-only executions, 56 MPI ranks per node were used. For MPI+OMP executions, 2 MPI ranks with 28 OpenMP threads per node were used, each of the MPI ranks being placed on separate NUMA domains to minimize cross-NUMA OpenMP memory traffic. For timing data collection, median is taken from 3 iterations of *Tiny* and *Small*, while *Medium* and *Large* were run for only 1 iteration. For all cases, the Intel Fortran/C/C++ compilers and Intel MPI from Intel Parallel Studio 2020 Update 4 were used. Codes are compiled with the *-O3 -no-prec-div -fp-model fast=2 -xCORE-AVX512 -ipo* flags. All profiles

TABLE V: SPEChpc 2021 Instruction Mix*

Benchmark	FP32 (% of uOps)	FP64 (% of uOps)	Non-FP (% of uOps)	Vectorization of FP (% of uOps)
605.lbm_s	0.00	51.98	48.02	86.80
613.soma_s	0.20	23.43	76.17	1.18
618.tealeaf_s	0.00	42.20	57.80	2.67
619.clvleaf_s	0.00	21.93	78.08	86.65
621.miniswp_s	0.00	8.92	91.07	57.90
628.pot3d_s	0.00	17.70	82.30	97.90
632.sph_exa_s	0.00	36.27	63.70	49.75
634.hpgmgfv_s	0.00	22.30	77.70	81.22
635.weather_s	0.00	26.32	73.67	3.45

*Results collected on Frontera for MPI-only executions using 4 nodes and 56 MPI ranks/node. Only results for the *small* suite are shown; the codes in other suites exhibit nearly identical characteristics.

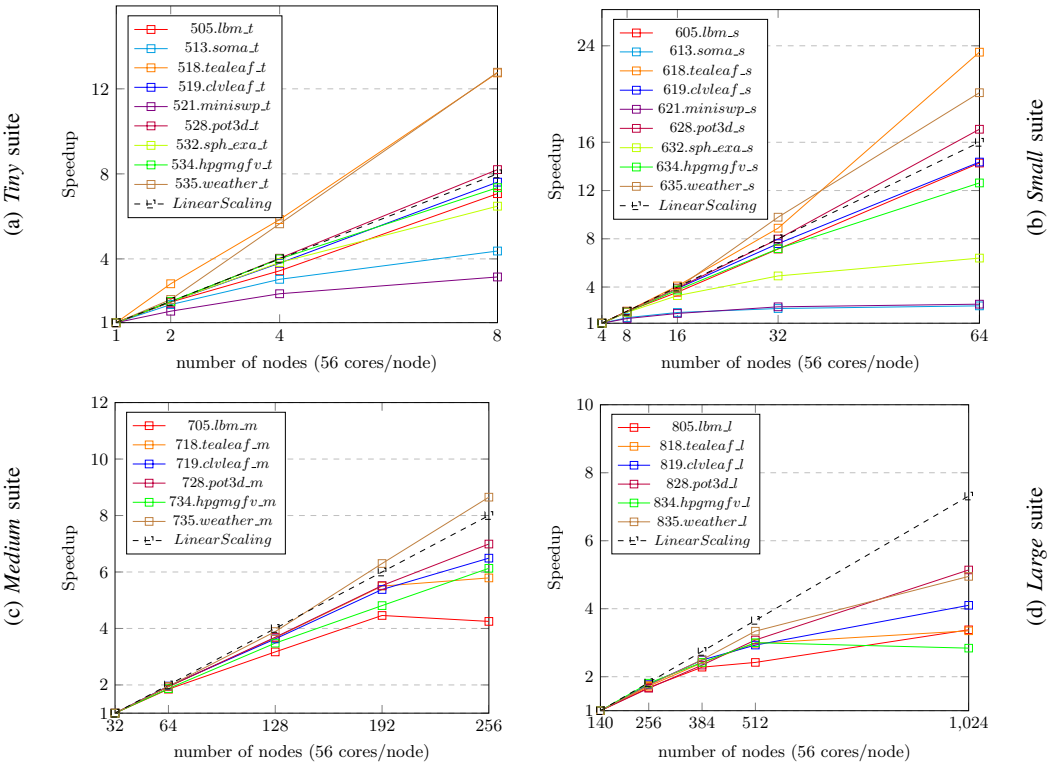


Fig. 1: Speedup for MPI-only. *Tiny*, *Small*, *Medium*, and *Large* suites on Frontera

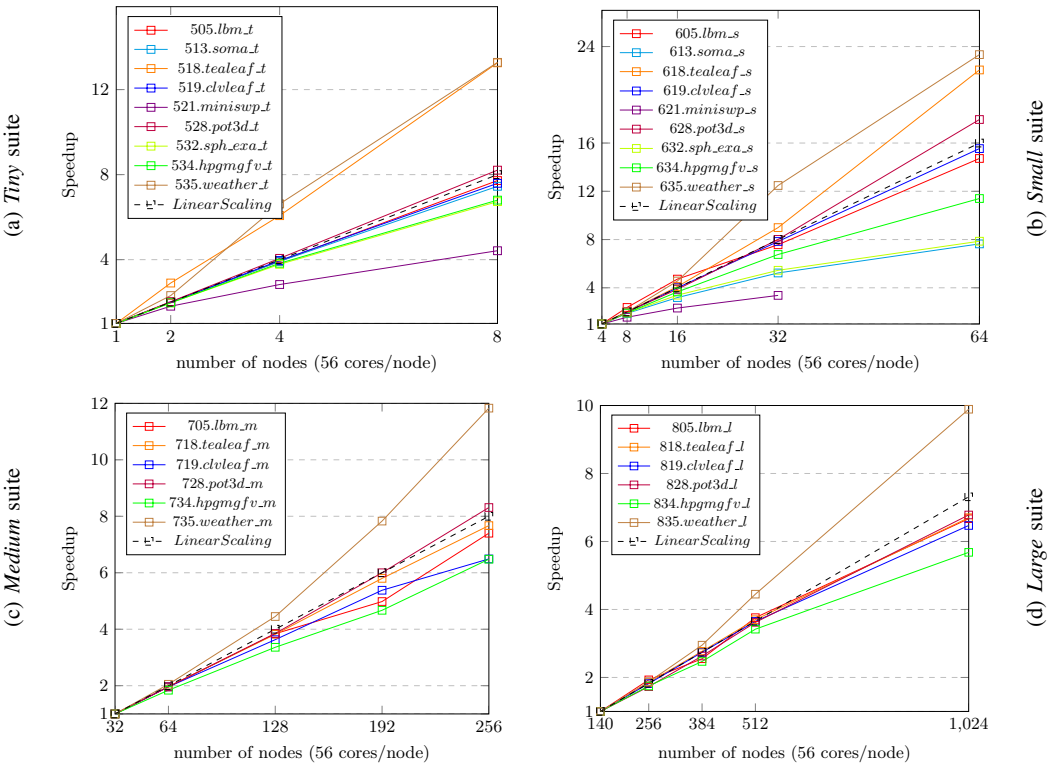
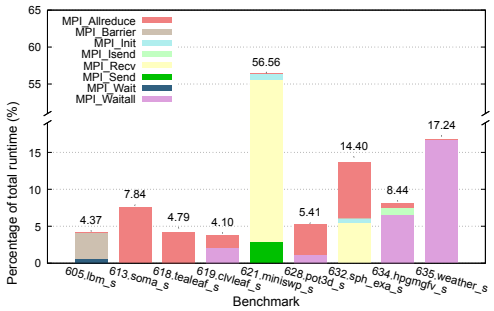
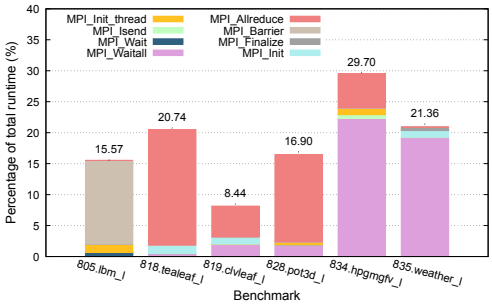


Fig. 2: Speedup for MPI+OMP. *Tiny*, *Small*, *Medium*, and *Large* suites on Frontera. *621.miniswp_s* unable to run on 64 nodes due to a potential OpenMP compiler bug.



(a) *Small* suite



(b) *Large* suite

Fig. 3: MPI functions distribution for small and large suites. Data collected on Frontera for MPI-only versions using 4 nodes and 56 MPI ranks/node for the *small* suite, and 140 nodes and 56 MPI ranks/node for the large suite. Plot shows MPI functions that contribute more than 0.5% of total execution time.

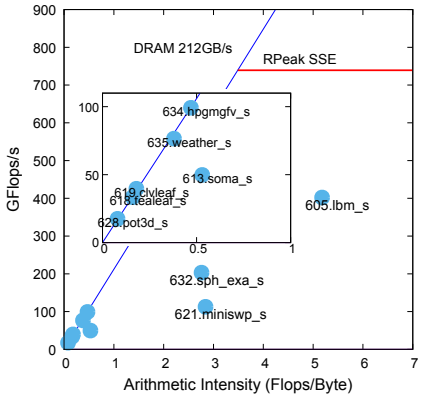


Fig. 4: Roofline plot for the *small* suite. Data collected for MPI-only versions using 4 nodes (224 ranks on Frontera). The roofline plots for the tiny, medium, and large suites are similar. Arithmetic intensity and memory bandwidth are collected for the entire duration of each program.

were collected using Intel Application Performance Snapshot representing basic tuning. Fine tuning of MPI parameters, such as rank/thread distribution, can certainly further improve the performance. Results on scalability and performance statistics from the experiments on Frontera are next discussed.

Results- Scalability: To understand the parallel performance of the benchmark, a set of **strong scaling** tests were performed for all four suites with both MPI-only and MPI+OMP programming models. Fig. 1 shows MPI-only scalability, and Fig. 2 demonstrates the MPI+OMP scalability. The execution times of the large suite with MPI+OpenMP are presented in Table IV. For additional information on execution times of other suites, please refer to performance data collected via Zenodo available at [**?**].

For the MPI-only runs, all suites scale well within their design limit, and have their appropriate applicable ranges. From the *tiny* suite, Minisweep and SOMA scale relatively poorly and scaling efficiency drops below 50% when more than 4 nodes are used. For SOMA, this is due to a high volume of all-to-all communication which is the nature of the code. Regarding Minisweep, the poor scaling beyond certain node count is due to a combination of inherently high amount of MPI communication and relatively small data set being used to fit the designed memory limit of the benchmark. As soon as the *tiny* suite no longer scales well, the *small* suite should be considered unless benchmarking MPI and interconnect is the main goal. From the *small* suite, Minisweep and SOMA also scale relatively poorly compared to other codes for the same reasons outlined above. Another code, SPH-EXA, joins the list of non-ideal scaling at 32 nodes also due to high volume of MPI traffic and reduced compute work per rank. Beyond 32 nodes on Frontera, the *medium* suite works best on up to 192 nodes, beyond which the *large* suite should be favored.

In the hybrid MPI+OMP case, performance is significantly better for several codes due to reduced MPI all-to-all communication and/or reduced memory traffic given the shared memory model. It is also worth mentioning that two codes, Tealeaf and miniWeather, will often achieve super-linear scaling. These two codes have the smallest memory footprint in the suites, yet are highly memory-bound. As more nodes are used to solve the same problem, the problem size per node reduces, resulting in decreased memory traffic and fewer memory stalls.

For example, miniWeather in the *small* suite (635.weahter_s) used 220.03 GB/s as peak memory bandwidth and 202.05 GB/s as average bandwidth when 4 nodes are used; this is close to the upper limit of the achievable memory bandwidth, while at 16 nodes, the memory bandwidth utilization was reduced to 197.52 GB/s as peak and 146.44 GB/s as average. On 32 nodes, the metrics further reduced to 155.29 GB/s and 118.11 GB/s, respectively. The resulting DRAM stall was 33.43%, 11.27%, 7.31% at 4, 16, and 32 nodes, respectively. Such a change in performance characteristics alleviates performance bottleneck and yields the super-linear scaling behavior. Demonstrated by all the scalability results, with the four different suites, the SPEChpc 2021 benchmark covers the entire spectrum from a few nodes to few hundreds. The *tiny* suite is suitable for a single node or a few nodes, the *small* suite is bigger and will be best suited to test a handful of nodes with a few hundred cores. The *medium* suite works well on a small cluster with a few thousand cores, while the *large* suite is large enough to test a medium-sized cluster with tens of thousands cores.

Performance statistics: MPI profiles were collected for all the codes in all four suites. In this manuscript, profiles of

the *small* suite executed on 4 nodes (224 ranks), representing the performance characteristics at small scale, and of the *large* suite executed on 140 nodes (7840 ranks), illustrating the performance characteristics at large scale, are shown. For additional information please refer to performance data collected via Zenodo [**?**]. The MPI functions that consume more than 0.5% of total execution time are shown in Fig. 3(a) and Fig. 3(b) for the *small* and *large* suites, respectively. Clearly, *MPI_Allreduce* plays a big role in many codes, such as SOMA, Tealeaf, Cloverleaf, POT3D, and SPH-EXA. Point-to-point communications are a key component for Minisweep, SPH-EXA, HPGMG-FV, and miniWeather. The purple bar denotes *MPI_Waitall* and indicates codes that rely on large amount of non-blocking communication. In addition, the floating point metrics were inspected. The instruction mix of single precision floating point (FP32) operations, double precision floating point (FP64) operations, and non-floating point (non-FP) operations, along with the SIMD vectorization rate are shown in Table V. The statistics are identical for different workloads; thereby, only the data for the *small* suite are presented here. With the -xCORE-AVX512 flag, the compilers will choose to the most efficient level of vectorization up to 512 bit, and in this particular case, it is mostly the 256-bit AVX2 instruction being used. Since the amount of AVX512 or AVX2 vectorization depends on the compiler switches and hardware, we only show the total percentage of FP operations being vectorized and do not distinguish the underlying instruction set in the table. These codes have a healthy mix of FP and non-FP operations, and most codes have their FP operations very well vectorized. The codes are FP64-heavy, with only SOMA having a tiny percentage of FP32 operations. Memory bandwidth is the limiting factor for many HPC codes. Most SPEChpc 2021 codes are also memory-bound. The roofline analysis in Fig. 4 reveals that five out of the nine codes land on the DRAM bandwidth boundary. This is for the *small* suite executed on 4 nodes (224 ranks); other workloads executed at the lower range of the design limits (Table II) show very similar characteristics. As more resources are added, codes like TeaLeaf and miniWeather become less memory-bound as mentioned in the above super-linear scaling discussion. All floating point operations of the the most compute-intensive code, LBM, in the suite can be vectorized; this code will benefit most from a long SIMD instruction set.

B. Performance Results on Summit

Experimental setup. These results use ORNL's Summit supercomputer which consists of over 4,600 nodes each with two 22-core IBM POWER9 CPU and six NVIDIA V100 GPUs. On Summit, the MPI+ACC and MPI+TGT models were executed using NVHPC 21.7 and IBM XL 16.1.1-10, respectively. For MPI+ACC, the benchmarks were compiled using *-O3 -acc=gpu*. For MPI+TGT, the benchmarks were compiled using *-O3 -qarch=pwr9 -qtune=pwr9 -qsmp=omp -qoffload -qtgtarch=auto*. For the experiments using MPI+ACC at-scale, two iterations were used. For MPI+TGT, however,

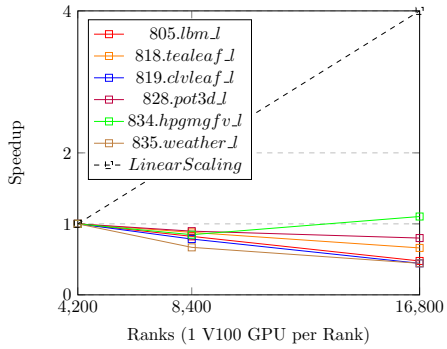


Fig. 5: Speedup for MPI+ACC, *Large* Suite on Summit

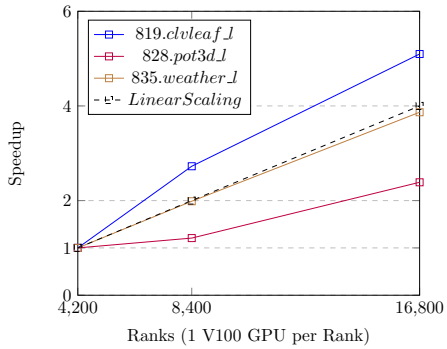


Fig. 6: Speedup for MPI+TGT, *Large* Suite on Summit

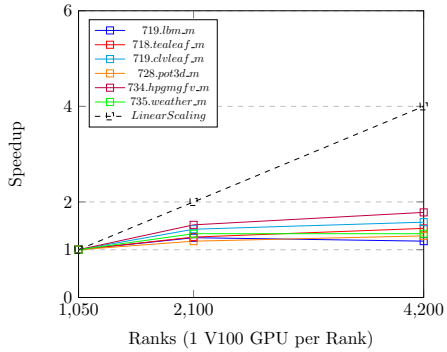


Fig. 7: Speedup with MPI+ACC, *Medium* Suite on Summit

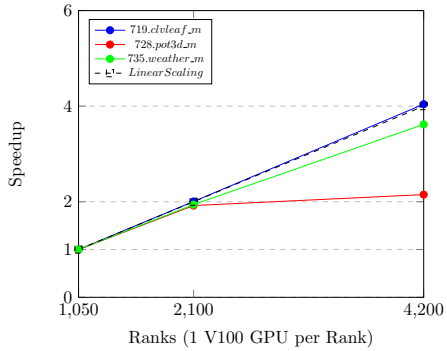


Fig. 8: Speedup with MPI+TGT, *Medium* Suite on Summit

TABLE VI: Execution Time of the *Medium* and *Large* Suites on Summit (in seconds)

Medium Suite						
Benchmark	**1050 Ranks**		**2100 Ranks**		**4200 Ranks**	
	ACC	**TGT**	**ACC**	**TGT**	**ACC**	**TGT**
705.lbm_m	20.4	CE	16.3	CE	17.3	CE
718.tealeaf_m	71.3	RE	56.4	RE	49.3	RE
719.clvleaf_m	32.0	4154	22.4	2072	20.3	1028
728.pot3d_m	95.7	3159	81.2	1645	74.2	1470
734.hpgmgfv_m	187	RE	123	RE	105	RE
735.weather_m	27.1	5416	20.3	2792	20.3	1497
Large Suite						
Benchmark	**4200 Ranks**		**8400 Ranks**		**16800 Ranks**	
	ACC	**TGT**	**ACC**	**TGT**	**ACC**	**TGT**
805.lbm_l	31.8	CE	38.6	CE	66.6	CE
818.tealeaf_l	60.1	RE	68.3	RE	90.9	RE
819.clvleaf_l	29.4	3415	37.4	1253	66.2	670
828.pot3d_l	140	3618	156	2996	175	1516
834.hpgmgfv_l	128	RE	151	RE	116	RE
835.weather_l	26.3	4887	39.3	2461	59.2	1264

due to the long execution time, a single iteration was used. All experiments on Summit used 1 MPI rank per V100 GPU.

Results. The experiments on Summit use both MPI+ACC and MPI+TGT programming models. Fig. 5 shows that running the *large* suite with more than 4,200 MPI ranks, each offloading to a single V100 GPU, results in poor scalability. The problem sizes in the *large* suite likely need to be increased to accommodate systems of Summit's scale. Only three of the six benchmarks successfully compiled with IBM XL 16.1.1-10 compiler. As shown in Fig. 6, although performance improves as the number of MPI ranks increases, overall, the execution time is over 200x slower than that obtained with MPI+ACC using the same number of GPUs (see Table VI). An important point to take away is that performance largely depends on the implementations. In addition, performance of the *medium* suite was investigated using 1,050 to 4,200 ranks, each offloading to a single V100 GPU. Both programming models at this scale show increasing speedup. However, as shown in Table VI, the execution time in these cases is 30-200X slower. CE and RE in Table VI refer to Compiler and Runtime errors. Additional studies are necessary to understand this degradation. Fig. 7 and Fig. 8 show strong scaling results for the *medium* suite.

C. Performance Results on JUWELS Booster

TABLE VII: Execution time of the *Large* Suite on JUWELS Booster (in seconds)

Benchmark	**400 Ranks**		**800 Ranks**		**1400 Ranks**	
	ACC	**TGT**	**ACC**	**TGT**	**ACC**	**TGT**
805.lbm_l	56.7	70.9	32.5	40	22.2	26.6
818.tealeaf_l	154	70.7	106	53.5	73.4	38.8
819.clvleaf_l	61.9	236	33.6	128	20.4	84.2
828.pot3d_l	200	280	150	205	95	136
834.hpgmgfv_l	209	539	141	332	133	229
835.weather_l	59.9	73.4	37.4	48.3	23.7	34.1

Experimental setup. The JUWELS Booster module consists of NVIDIA's A100 GPUs and AMD EPYC Rome CPUs. For both MPI+TGT and MPI+ACC runs, the GCC 10.3.0 and NVHPC 21.5 compilers are used, respectively, while ParaStation MPI/5.2.9-1 is used for MPI. Each of the booster module consists of 4 NVIDIA A100 GPUs. For both MPI+ACC and MPI+TGT versions, we use *-w -Mfprelaxed -Mnouniform -Mstack_arrays -fast*. For all the experiments, due to long execution time, one iteration was used.

Results. The **strong scaling** results on JUWELS for the *large* suite using both MPI+ACC and MPI+TGT versions

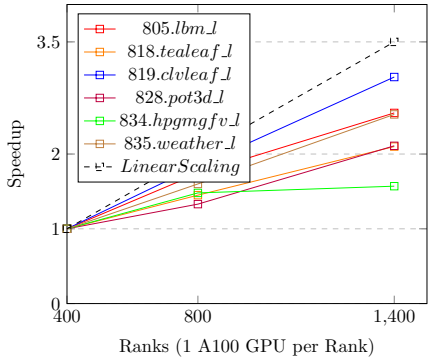


Fig. 9: Speedup for MPI+ACC, *Large* Suite on JUWELS Booster

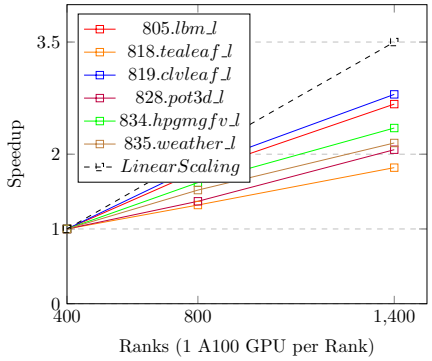


Fig. 10: Speedup for MPI+TGT, *Large* Suite on JUWELS Booster

are shown in Figures 9 and 10. These show that almost all six benchmarks achieve close to linear scaling with room for improvement. For both MPI+ACC and MPI+TGT versions, Cloverleaf shows the best scaling behavior followed by LBM and the rest. Table VII presents the execution time in seconds for the *large* suite for both versions spanning 400 to 1,400 MPI ranks. We observe that for LBM, Cloverleaf, POT3D, HPGMG-FV and miniWeather, the MPI+ACC version performs better than the MPI+TGT version. Cloverleaf is evidently over 3.9x faster on an average across ranks and HPGMG-FV over 2.2x faster on an average across ranks. However for Tealeaf, the MPI+TGT version is about 2x better than the MPI+ACC version. The observations above were similar for the *medium* suite as well. Future work will include investigation of these discrepancies in more detail. For additional information on any of these experiments along with data on the *medium* suite, please refer to performance data collected in Zenodo [**?**]. While comparing the parallel efficiency of the *large* suite with that of the *medium* suite, we observe that the efficiency of medium workload drops below 75% with 400 ranks for almost all the benchmarks, indicating that for ranks above 400, it is better to use larger workload to achieve the best parallel efficiency especially on large clusters such as JUWELS. The trend seems to hold for both MPI+ACC and MPI+TGT versions.

D. Performance Results on Spock

Experimental setup. The Spock pre-production AMD [24] system at ORNL is used to collect scalability and profiling data

TABLE VIII: Execution Time of the *Small* Suite on Spock (in seconds)

Benchmark	16 Ranks	24 Ranks	32 Ranks
605.lbm_s	234.2	169.4	142.3
618.tealeaf_s	624.2	507.6	461.8
621.miniswp_s	1319.6	967.9	828.1
628.pot3d_s	1426.1	968.7	897.4
632.sph_exa_s	1065.5	747.2	631.6
635.weather_s	5154.7	3305.4	2226.4

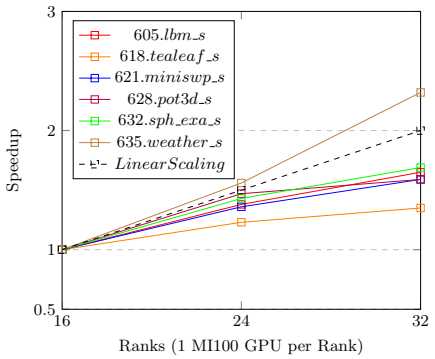


Fig. 11: Speedup for MPI+TGT, *Small* Suite on Spock

using the SPEChpc 2021 benchmark suites. Spock consists of 36 compute nodes, each with one AMD EPYC 7662 64-Core processor, 256 GB DDR4 memory, and four AMD Instinct MI100 GPUs. These experiments used 16, 24, and 32 MPI ranks with 4 ranks per node corresponding to 4, 6 and 8 nodes. Two iterations were used for the 4- and 8-node runs and 3 iterations were used for the 6-node run. The reported time is the average over all iterations. The benchmark suites were compiled with the LLVM Clang v12.0.0 compiler suite packaged with ROCm v4.2.0, acceleration libraries for the MI100 architecture, and Cray MPICH v8.1.8. The benchmarks were compiled using *-O3 -fopenmp -target x86_64-pc-linux-gnu -fopenmp-targets=amdgcn-amd-amdhsa -Xopenmp-target=amdgcn-amd-amdhsa -march=gfx908*.

Results. We ran SPEChpc 2021 *small* suite on Spock. SOMA could not compile because of an internal compiler error while processing the TGT directives. HPGMG-FV did not run to completion within the time allowed for jobs on Spock; this is potentially caused by the TGT or MPI implementations. The exact cause is unknown and investigation is ongoing. We present **strong scalability** plots for the *small* suite on Spock in Fig. 11. The execution times for these codes (shown in Table VIII) decreased for all codes as more nodes were added.

VI. Related Work

Numerous benchmarks focus on hardware performance features. For instance, SHOC [26] that includes both the OpenCL and CUDA implementations provides microbenchmarks and small application kernels. The classic NASA parallel benchmarks (NPB) [27] are a popular multi-zone, hybrid (MPI+X) benchmark suite [28], [29] that are used for evaluating prototypes of programming features, runtime or compiler implementations. A benchmark suite [30] consisting of a set of low-level operations to test raw performances compilers and hardware is developed by EPCC. The HPC community also widely uses HPL [31] for floating-point focused operations, HPMG [32] for multigrid methods, and HPCG [33] for bandwidth-focused operations. The benchmarking landscape has also been influenced by so-called mini or proxy applications that abstract a specific (performance) behaviour from the real-world application they mimic. Examples are Mantevo [34] and Exascale Computing Project's (ECP) Mini-apps [35]–[37].

The SPEChpc 2021 suites differ from the above benchmarking effort in a way that the suites are incorporated into the SPEC harness (detailed in Section IV) adding benefits such as reproducibility, results publication, and ranking of systems utilizing the SPEC scores. Other SPEC related benchmarking efforts include the release of the following suites for the HPC community: SPEC HPC96 [38] benchmark suites, improved and replaced by SPEC HPC2002 [39], SPEC OMP2001 [40], superseded by SPEC OMP2012 [41] for shared-memory parallel systems using CPUs, SPEC MPI2007 [42] for distributed-memory parallel systems using CPUs, and SPEC ACCEL [43] with applications using OpenACC and OpenMP 4.5 in a performance portable manner, thus, targeting different (single) accelerator devices. SPEChpc 2021 suites complement the above SPEC suites by enabling scalability and efficiency studies at large scale, using large homogeneous and heterogeneous HPC clusters that are recently increasingly being built with multiple-GPUs per node.

VII. Conclusions and Future Work

This work illustrates first experiences in performance benchmarking with the new SPEChpc 2021 suites on 4 HPC systems: Frontera, Summit, JUWELS Booster, and Spock, that have diverse hardware architectures: Intel Cascade Lake CPUs, NVIDIA V100/A100 GPUs, and AMD Instinct MI100 GPUs. The SPEChpc 2021 suites (tiny, small, medium, and large) employ MPI-only, MPI+OMP, MPI+ACC, and MPI+TGT as parallel programming APIs. Analysis of SPEChpc's application properties on Frontera reveals that (1) most codes are memory-bound, (2) they utilize mainly FP64 as in traditional HPC applications, and (3) 8-30% of the total execution time of the *large* suite corresponds to MPI, in particular *MPI_Allreduce*. We presented strong-scaling results for all suites on all systems with up to 16,800 MPI ranks. Selecting an appropriate suite for performance benchmarking is critical for trading off poor scalability with superlinearity. The best and the worst parallel efficiencies were obtained on the state-of-the-art NVIDIA's A100 GPUs. On the same accelerator-based systems, the performance of MPI+TGT implementations lags behind that of MPI+ACC implementations. These performance differences are due to the OpenMP offloading features' implementations in compilers (e.g., NVHPC vs IBM XL vs LLVM) which require further in-depth investigation. These results have been motivating compiler vendors to invest efforts towards closing the performance gaps. All performance measurements presented here will be published on the SPEC website [4] upon the official release of the SPEChpc 2021 suites by SPEC HPG. Due to the growing importance of application scaling and of scalable benchmarking

in HPC, weak scaling benchmarks are needed that will prove to be beneficial for the HPC community, and the SPEChpc 2021 benchmarks could be extended in this direction.

VIII. Acknowledgements

The authors would like to acknowledge a number of facilities and grants for their support, but first and foremost, we would like to thank the full SPEC HPG group for the tremendous effort behind developing and releasing the SPEChpc 2021 benchmark suites. Then, the Center for Information Services and HPC at TU Dresden for providing its facilities for high throughput calculations; RWTH Aachen University under project rwth0663 for supporting simulations on their computing resources; supported by NSF under grant no. 1814609; Gauss Centre for Supercomputing e.V. for funding this project by providing computing time through the John von Neumann Institute for Computing (NIC) on the GCS Supercomputer JUWELS at Jülich Supercomputing Centre (JSC); Oak Ridge Leadership Computing Facility, a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725 for resources used; the Frontera supercomputer at TACC funded by NSF for large scaling calculations and profiling; the Swiss PASC initiative via the SPH-EXA project; the Swiss State Secretariat for Education, Research and Innovation (SERI).

References

[1] "CCRedit author statement," 2021. [Online]. Available: https://www.elsevier.com/authors/policies-and-guidelines/credit-author-statement

[2] SPEC, "SPEC HPG: High Performance Group," 2021. [Online]. Available: https://www.spec.org/hpg/

[3] SPEC, "SPEC MPI accelerator benchmark search program," 2021. [Online]. Available: http://spec.org/hpg/search/

[4] "SPEChpc 2021 homepage," 2021. [Online]. Available: https://www.spec.org/hpc2021/

[5] S. Boehm *et al.*, "Evaluating performance portability of accelerator programming models using SPEC ACCEL 1.2 benchmarks," in *ISC*. Springer, 2018, pp. 711–723.

[6] G. Juckeland *et al.*, "From describing to prescribing parallelism: Translating the SPEC ACCEL OpenACC suite to OpenMP target directives," in *ISC*. Springer, 2016, pp. 470–488.

[7] S. Pophale *et al.*, "Comparing high performance computing accelerator programming models," in *International Conference on High Performance Computing*. Springer, 2019, pp. 155–168.

[8] T. Huber *et al.*, "Impact of virtualization and containers on application performance and energy consumption," *IU Scholar Works*, 2018.

[9] T. Cramer *et al.*, "Evaluating the performance of OpenMP offloading on the NEC SX-Aurora TSUBASA vector engine," *Supercomputing Frontiers and Innovations*, vol. 8, no. 2, p. 59–74, Aug. 2021.

[10] T. Cramer *et al.*, "Performance analysis for target devices with the OpenMP tools interface," in *2015 IEEE IPDPSW*, 2015, pp. 215–224.

[11] OpenMP ARB, "OpenMP 5.1," 2020. [Online]. Available: https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5-1.pdf/

[12] T. Hilbrich *et al.*, "MPI runtime error detection with MUST: Advances in deadlock detection," in *SC12*, 2012, pp. 1–10.

[13] A. Knüpfer *et al.*, "Score-P: A Joint Performance Measurement Run-Time Infrastructure for Periscope,Scalasca, TAU, and Vampir," in *Tools for High Performance Computing 2011*, H. Brunst *et al.*, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 79–91.

[14] C. Feld *et al.*, "Score-P and OMPT: navigating the perils of callback-driven parallel runtime introspection," in *IWOMP*. Springer, 2019, pp. 21–35.

[15] A. Afzal *et al.*, "Desynchronization and wave pattern formation in mpi-parallel and hybrid memory-bound programs," in *High Performance Computing*, P. Sadayappan *et al.*, Eds. Springer International Publishing, 2020, pp. 391–411.

[16] Red Hat Customer Portal, "Kernel: Race condition in hashtables causes kworker thread to loop into rht_deferred_worker() function," 2020. [Online]. Available: https://access.redhat.com/solutions/5337251

[17] T. Ilsche *et al.*, "lo2s — Multi-core System and Application Performance Analysis for Linux," in *2017 IEEE CLUSTER*, 2017, pp. 801–804.

[18] lkml.org: Josh Elsasser, "rhashtable: avoid reschedule loop after rapid growth and shrink," 2019. [Online]. Available: https://lkml.org/lkml/2019/1/23/789

[19] J. Bucek *et al.*, "SPEC CPU2017: Next-generation compute benchmark," in *ICPE*, 2018, pp. 41–42.

[20] C. D. Spradling, "SPEC CPU2006 benchmark tools," *ACM SIGARCH Computer Architecture News*, vol. 35, no. 1, pp. 130–134, 2007.

[21] "Frontera at the Texas Advanced Computing Center TACC," 2021. [Online]. Available: https://www.tacc.utexas.edu/systems/frontera

[22] "Summit at ORNL," 2021. [Online]. Available: https://docs.olcf.ornl.gov/systems/summit_user_guide.html

[23] Forschungszentrum Jülich, "JUWELS booster overview," 2021. [Online]. Available: https://apps.fz-juelich.de/jsc/hps/juwels/booster-overview.html

[24] "SPOCK a pre-exascale system at ORNL," 2021. [Online]. Available: https://docs.olcf.ornl.gov/systems/spock_quick_start_guide.html

[25] H. Brunst *et al.*, "First Experiences in Performance Benchmarking with the New SPEChpc 2021 Suites - Measurement Data," Oct. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5753669

[26] A. Danalis *et al.*, "The scalable heterogeneous computing (SHOC) benchmark suite," in *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*, 2010, pp. 63–74.

[27] D. Bailey *et al.*, "The NAS parallel benchmarks 2.0," Technical Report NAS-95-020, NASA Ames Research Center, Tech. Rep., 1995.

[28] X. Wu and V. Taylor, "Performance characteristics of hybrid MPI/OpenMP implementations of NAS parallel benchmarks sp and bt on large-scale multicore supercomputers," *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, no. 4, pp. 56–62, 2011.

[29] H. Jin and R. F. Van der Wijngaart, "Performance characteristics of the multi-zone NAS parallel benchmarks," *JPDC*, vol. 66, no. 5, pp. 674–685, 2006.

[30] J. M. Bull and D. O'Neill, "A microbenchmark suite for openmp 2.0," *ACM SIGARCH Comp. Arch. News*, vol. 29, no. 5, pp. 41–48, 2001.

[31] J. J. Dongarra *et al.*, "The LINPACK benchmark: past, present and future," *Concur. and Comput.: Practice and Experience*, vol. 15, no. 9, pp. 803–820, 2003.

[32] M. Adams, "HPGMG 1.0: A benchmark for ranking high performance computing systems," 2014.

[33] M. A. Heroux *et al.*, "HPCG benchmark technical specification," SNL, Tech. Rep., 2013.

[34] M. A. Heroux *et al.*, "Improving performance via mini-applications," *SNL, Tech. Rep. SAND2009-5574*, vol. 3, 2009.

[35] J. Cook *et al.*, "Proxy app prospectus for ECP application development projects," LLNL, Tech. Rep., 2017.

[36] V. Dobrev *et al.*, "Identify initial kernels, bake-off problems (benchmarks) and miniapps wbs 1.2. 5.3. 04, milestone ceed-ms6," *ECP Milestone Report*, 2017.

[37] N. Sultana *et al.*, "Understanding the usage of MPI in exascale proxy applications," in *2018 SC Conference Workshop*, 2018.

[38] R. Eigenmann and S. Hassanzadeh, "Benchmarking with real industrial applications: the SPEC High-Performance Group," *IEEE Computational Science and Engineering*, vol. 3, no. 1, pp. 18–23, 1996.

[39] R. Eigenmann *et al.*, "SPEC hpc2002: The next high-performance computer benchmark," in *International Symposium on High Performance Computing*. Springer, 2002, pp. 7–10.

[40] H. Saito *et al.*, "Large system performance of SPEC OMP2001 benchmarks," in *International Symposium on High Performance Computing*. Springer, 2002, pp. 370–379.

[41] M. S. Müller *et al.*, "SPEC OMP2012—an application benchmark suite for parallel systems using OpenMP," in *IWOMP*. Springer, 2012, pp. 223–236.

[42] M. S. Müller *et al.*, "SPEC MPI2007—an application benchmark suite for parallel systems using MPI," *Concur. and Comput.: Practice and Experience*, vol. 22, no. 2, pp. 191–205, 2010.

[43] G. Juckeland *et al.*, "SPEC ACCEL: A standard application suite for measuring hardware accelerator performance," in *PMBS*. Springer, 2014, pp. 46–67.