SPEC ACCEL: A Standard Application Suite for Measuring Hardware Accelerator Performance

Guido Juckeland[1,2(✉)], William Brantley[1,3], Sunita Chandrasekaran[1,4], Barbara Chapman[1,4], Shuai Che[1,3], Mathew Colgrove[1,5], Huiyu Feng[1,6], Alexander Grund[1,2], Robert Henschel[1,7], Wen-Mei W. Hwu[1,8], Huian Li[1,7], Matthias S. Müller[1,9], Wolfgang E. Nagel[1,2], Maxim Perminov[1,10], Pavel Shelepugin[1,10], Kevin Skadron[1,11], John Stratton[1,8,12], Alexey Titov[1,3], Ke Wang[1,11], Matthijs van Waveren[1,13], Brian Whitney[1,14], Sandra Wienke[1,9], Rengan Xu[1,4], and Kalyan Kumaran[1,15]

[1] SPEC High Performance Group, Gainesville, USA
info@spec.org
http://www.spec.org/hpg
[2] Center for Information Services and High Performance Computing (ZIH), Technische Universität Dresden, 01062 Dresden, Germany
guido.juckeland@tu-dresden.de
[3] Advanced Micro Devices, Inc., Sunnyvale, CA, USA
[4] University of Houston, Houston, TX, USA
[5] NVIDIA, Santa Clara, CA, USA
[6] Silicon Graphics International Corp., Milpitas, CA, USA
[7] Indiana University, Bloomington, IN, USA
[8] University of Illinois (UIUC), Champaign, IL, USA
[9] RWTH Aachen University, Aachen, Germany
[10] Intel, Nizhny Novgorod, Russia
[11] University of Virginia, Charlottesville, VA, USA
[12] Colgate University, Hamilton, NY, USA
[13] Compilaflows, Toulouse, France
[14] Oracle, Redwood Shores, CA, USA
[15] Argonne National Laboratory, Lemont, IL, USA

Abstract. Hybrid nodes with hardware accelerators are becoming very common in systems today. Users often find it difficult to characterize and understand the performance advantage of such accelerators for their applications. The SPEC High Performance Group (HPG) has developed a set of performance metrics to evaluate the performance and power consumption of accelerators for various science applications. The new benchmark comprises two suites of applications written in OpenCL and OpenACC and measures the performance of accelerators with respect to a reference platform. The first set of published results demonstrate the viability and relevance of the new metrics in comparing accelerator performance. This paper discusses the benchmark suites and selected published results in great detail.

S.A. Jarvis et al. (Eds.): PMBS 2014, LNCS 8966, pp. 46–67, 2015.
DOI: 10.1007/978-3-319-17248-4_3

Keywords: SPEC · SPEC ACCEL · OpenCL · OpenACC · Energy measurements

1 Introduction

The Standard Performance Evaluation Cooperation (SPEC) stands as a successful example of collaboration among vendors and researchers in creating benchmarks that lead to fair comparison and reproducible results. SPEC's High Performance Group (HPG) has been active for over 20 years – since its initial benchmark derived from David Kuck's Perfect Suite – in creating industry standard benchmarks that highlight and compare various aspects of high performance computing systems. The group's members are leading high performance computing (HPC) vendors, national laboratories, and universities from all around the world.

SPEC HPG has been developing and maintaining application based benchmarks and performance metrics supporting a variety of programming models and stressing various hardware features. This includes inter-node parallelism (covered by SPEC MPI2007), intra-node parallelism (covered by SPEC OMP2012), and offloading computation to a hardware accelerator (covered by SPEC ACCEL in this paper). SPEC MPI2007 offers a suite of 18 applications running on up to 2,048 message passing interface (MPI) ranks [22]. Its goal is to evaluate MPI-parallel, floating point, compute intensive performance of clusters and multi-processor systems. SPEC OMP2012 offers a suite of 14 applications based on scientific and engineering application codes using the OpenMP 3.1 standard [21]. The benchmark also includes an optional metric for measuring energy consumption.

The advent of hardware accelerators as a standard component in high performance computers led SPEC HPG to investigate performance characterization in this additional layer of parallelism. Keeping with the group's guidelines, a performance evaluation must be based on an open programming model so that multiple hardware and software environments can be evaluated. As a result, the popular but vendor specific programming model—CUDA—was not investigated. Instead OpenCL, as a low level, and OpenACC, as a high level, hardware accelerator programming models have been chosen to provide two independent subsuites within SPEC ACCEL[1]. In a similar manner, the group has carefully brought together applications from various computational and scientific domains that stress the accelerator with very different demands. The runtime and energy consumption of the application is monitored during multiple runs and results are presented in the typical SPEC manner. The peer review process for every published result ensures the validity and reproducibility of a SPEC ACCEL run.

This paper first presents previous work in Sect. 2, then introduces the measurement methodology in Sect. 3. The selected applications for each of the two subsuites/programming model are discussed in Sect. 4. Section 5 shows how energy

[1] Since OpenMP 4.0 offloading is still limited to one hardware platform and one compiler it has at the moment vendor specific characteristics. OpenACC on the other hand offers three different compilers and also four (via the CAPS compilers, two via the PGI compilers) hardware platforms.

measurements can enrich the performance data. Section 6 demonstrates how the first published results underline the usefulness of the benchmark in comparing both hardware and software environment for accelerators.

2 Related Work

There has been work done in creating benchmarks for measuring hardware accelerator performance, but all of them are academic in nature and none of them share SPEC's philosophy when it comes to design for standard benchmarks. SPEC strongly believes in same source code for all, a detailed set of run and reporting rules for compliant results, and a peer review process for all results before publication on the SPEC website. This enables fair comparison of results.

With respect to OpenCL benchmarking, both the Parboil [27] and Rodinia [4,5] have been very popular academic benchmarks with more than 1,000 citations between them in research papers. The Parboil and Rodinia developers approached SPEC to standardize the benchmark, to develop a set of run and reporting rules that enable fair performance metrics, and to build a result repository, since the groups could not provide that themselves. SPEC HPG worked with both groups of developers to ensure that the benchmarks taken into the OpenCL suite are running on all available platforms. A number of improvements suggested by SPEC HPG has made it into recent releases of Parboil and Rodinia.

SHOC [7] is a benchmark suite that evolved in the academic circles and includes both the OpenCL and CUDA implementations. As another approach for OpenCL, the SHOC benchmark measures low level hardware performance features rather than general application run time performance. It is, therefore, not suitable for the SPEC approach, but has its relevance on comparing very specific small scale algorithms on various platforms.

On the OpenACC side, the Edinburgh Parallel Computing Centre (EPCC) has developed a benchmark suite [14] comprising a set of low-level operations designed to test raw performance of compilers and hardware and a set of kernels found in scientific codes. The SPEC ACCEL OpenACC suite, on the other hand, is comprised of full scientific applications rather than kernels.

3 Design and Principles of SPEC ACCEL

3.1 Benchmark Philosophy and General Design

The goal of SPEC ACCEL is to measure the performance of compute intensive applications using hardware acceleration. It is designed to compare different accelerator platforms, but also different devices within a platform. A platform consists of all the hardware and software components necessary to execute SPEC ACCEL: the accelerator, the host system including its CPU, the interconnect or bus used for the data transfers between host and accelerator, the support libraries and drivers, and the compiler.

SPEC ACCEL uses vendor independent programming standards to target the accelerators. In its current implementation, OpenCL and OpenACC are supported. Both standards apply the offload model for the accelerated computation. The offload model consists of a CPU (host) which runs the main program, copies the data needed by the accelerated computation to and from discrete memory on the accelerator, and launches the accelerated routines.

The SPEC ACCEL benchmark is provided within the SPEC harness that is also used for other SPEC benchmarks like SPEC CPU2006, SPEC OMP2012, and SPEC MPI2007. With the help of a user supplied config file, the benchmark codes are automatically compiled, the total execution times measured, the results verified for correctness, and a report generated. Optionally, a power measurement is also included to allow the comparison of both time-to-solution and energy-to-solution [16].

The generated performance reports may be submitted to SPEC for publication. The SPEC HPG committee reviews SPEC ACCEL results for consistency, adherence to the run rules, and whether enough details have been supplied for others to reproduce the results. If the committee accepts the results, they are published together with the config file on the SPEC website.

3.2 Run Rules

The run rules cover the procedure for the building and running the benchmark and disclosing the benchmark results. They closely follow the established SPEC run rules but need to take into account the peculiarities of systems with hardware accelerators. This section explains where the run rules from SPEC ACCEL extend or deviate from the common rule set of SPEC. The goal is that users of accelerator systems can compare objectively the accelerators of different vendors on the SPEC web site.

The SPEC ACCEL benchmark suite supports base, peak, and power metrics. The performance metrics are the geometric mean of the run time ratios of the system under test with the run time of the reference machine. The reference system is a SGI C3108-TY11 (a dual socket Intel Xeon E5620 system with 24 GB of main memory) using an NVIDIA Tesla C2070 with error checking and correcting (ECC) enabled as the accelerator. The system runs SLES11 SP2 as the operating system and uses the built-in GNU compilers for the OpenCL suite and the PGI compilers version 13.9 for the OpenACC suite. The reference measurements also include energy metrics recorded from a ZES Zimmer LMG450 power analyzer. All benchmarks in the two suites were targeted to run for at least 100 s on the reference machine in order to provide a useful time for measurements, even for future hardware with significantly higher performance.

A set of tools is supplied to build and run the benchmarks. These SPEC tools must be used to generate publishable results. This helps ensure reproducibility of results by requiring that all individual benchmarks in the suite are run in the same way and that a configuration is available that defines the optimizations used.

The optimizations used are expected to be applicable beyond the SPEC benchmarks, and it is expected that system or compiler vendors would endorse the general use of these optimizations by customers who seek to achieve good application performance. The system components, including software, must be generally available within 90 days of publication; there needs to be a certain level of maturity and general applicability in the methods.

For the base metric, the same compiler must be used for all applications of a given language within a benchmark suite. Except for portability flags, all flags or options that affect the transformation process from SPEC-supplied source to the completed executable must be the same for all modules of a given language. For the peak metric, each module may be compiled with a different compiler and a different set of flags or options. For the OpenCL suite, it is also allowed to change the work distribution on the accelerator by using a different work group size per benchmark.

As used in these run rules, the term run-time dynamic optimization (RDO) refers broadly to any method by which a system adapts an executing program for improved performance based upon observation of its behavior as it runs. Run time dynamic optimization is allowed, subject to the provisions that the techniques must be generally available, documented, and supported.

Results are published on the SPEC web site. A published result must contain enough information to enable others to replicate the result. The information to document an accelerator includes the model name, name of hardware vendor, name and type of the accelerator, description of the connection to the host system, whether ECC is enabled or not, and the device driver names and versions.

4 Description of the Applications

The applications comprising the SPEC ACCEL benchmark fall into two categories depending on the programming model: OpenCL or OpenACC. They cover a wide range of scientific domains and also have very different performance characteristics as shown in Tables 1 and 2. The SPEC ACCEL suite is written to comply with OpenCL 1.1 [15] and OpenACC 1.0 [1]. This section introduces both suites of the SPEC ACCEL benchmark.

4.1 SPEC ACCEL OCL Suite

In order to fit into the design principles for SPEC ACCEL (see Sect. 3), the original benchmarks taken into the OpenCL suite were in part heavily modified. Some benchmarks were dropped when they could not be modified to meet the SPEC HPG requirements. The Parboil Benchmark Suite [27] is the origin of the first nine OpenCL applications of SPEC ACCEL, the other ten applications are taken from the Rodinia Benchmark Suite [4,5]. A number of benchmarks received larger data sets than they originally had, so that their runtime increased to the required 100 s on the reference machine. All benchmarks were tested on all hard- and software platforms available to the HPG members. This resulted in

numerous bug fixes both in the benchmarks but also in OpenCL runtime environments, thus, showcasing how this benchmark suite can be used as a validation suite for OpenCL hardware and software as well.

The selected applications span a wide area of science ranging from astronomy, bioinformatics, computer science, electrical engineering, mathematics, mechanical engineering, medicine and physics. They are also selected to cover different usage modes for hardware accelerators. Benchmarks like *101.tpacf* and *121.lavamd* use one or two long running kernels. Other benchmarks like *123.nw* use almost 350,000 very short kernel launches in order to see how well the accelerator ecosystem can handle such extreme cases. The same is true for the number of data transfers between the host and device and the amount of data being transferred. While most benchmarks follow the usual offloading scheme of limiting the amount of transfers, *116.histo*, *117.bfs*, and *127.srad* use well over 10,000 data transfers, in case of *127.srad* also of very small size. The accelerator utilization (which is the amount of time the accelerator is occupied), as well as the time for data transfers, also offers a broad spectrum of load situations. However, most applications try to utilize the accelerator fully, while only a few like *116.histo*, *120.kmeans*, or *127.srad* primarily stress the host-device transfers. *114.mriq* is a special case since it shows both a high device utilization, but also high data transfer time. In this case, the data transfers are launched asynchronously, but the NVIDIA OpenCL runtime forces them to synchronize, thus, completing the transfer only after the previously launched kernel has completed.

The computational algorithms employed by the applications of the benchmark suite also vary widely:

101.tpacf computes the two-point angular correlation function of a collection of observed and randomly generated astronomical bodies. It compares pairs of angular coordinates, computes their angular distance, and computes a histogram of those distances. The histogram is privatized, with multiple copies in each work group, reducing bandwidth and atomic operation demand on the global memory system.

103.stencil implements an iterative Jacobi solver of the heat equation on a 3-D structured grid. The implementation uses double buffering to eliminate timing effects on numerical output values for a fixed number of iterations. On the reference machine, each iteration completes quickly enough so that platform overheads for kernel launches and other operations has an impact on the total performance.

104.lbm is related to the SPEC CPU2006 benchmark of the same name, and implements the Lattice-Boltzmann Method for fluid dynamics simulation [23]. This particular implementation supports immobile solid obstacles to fluid flow in a lid-driven closed cavity. Individual iterations have a long enough runtime that kernel execution performance is the most relevant factor for the total application performance.

110.fft implements a 1-D, Radix-2 Fast Fourier Transform. The kernel source included could be configured to support other radices, but for consistency, the benchmark only supports Radix-2.

Table 1. OpenCL application key facts. Profiling data taken from VampirTrace OpenCL tracing when running on NVIDA Tesla K20 using NVIDIA OpenCL.

Application	Lines of code	Language	Area
101.tpacf	520	C++	Astrophysics
103.stencil	308	C++	Thermodynamics
104.lbm	853	C++	Fluid dynamics
110.fft	642	C	Signal processing
112.spmv	1,108	C++	Sparse linear algebra
114.mriq	569	C	Medicine
116.histo	1,174	C	Silicon wafer verification
117.bfs	452	C	Electronic design automation, graph traversals
118.cutcp	1,192	C	Molecular dynamics
120.kmeans	2,243	C++	Dense linear algebra, data mining
121.lavamd	773	C	N-body, molecular dynamics
122.cfd	1,677	C++	Unstructured grid, fluid dynamics
123.nw	468	C++	Dynamic programming, bioinformatics
124.hotspot	407	C	Structured grid, physics simulation
125.lud	656	C++	Dense linear algebra, linear algebra
126.ge	1,497	C++	Dense linear algebra, linear algebra
127.srad	1,499	C	Structured grid, image processing
128.heartwall	4,671	C	Structured grid, medical imaging
140.bplustree	2,870	C	Graph traversal, search

Application	Kernel invocations	Accelerator utilization	MiBytes transferred	Number of transfers	Transfer time
101.tpacf	1	98.7%	56.2	3	0.02%
103.stencil	20,000	97.0%	294	3	0.10%
104.lbm	5,000	95.3%	331	3	0.15%
110.fft	12,800	97.9%	200	1	0.07%
112.spmv	50,000	88.3%	161	8	0.04%
114.mriq	197	94.2%	61.3	206	96.0%
116.histo	48,000	0.29%	186,755	37,041	95.8%
117.bfs	40,977	73.2%	17,636	84,349	20.8%
118.cutcp	3,250	94.3%	72.6	5	0.07%
120.kmeans	1,617	4.74%	6,296	3,233	60.0%
121.lavamd	2	94.8%	2,650	5	0.94%
122.cfd	72,217	81.9%	12.5	8	0.49%
123.nw	347,088	70.4%	512	2	0.51%
124.hotspot	30,000	85.8%	16.0	1	0.01%
125.lud	17,988	97.2%	6,836	7	1.43%
126.ge	11,262	97.9%	484	6	0.19%
127.srad	39,002	23.2%	112	13,006	97.1%
128.heartwall	100	99.0%	186	109	0.05%
140.bplustree	3,200	98.6%	68.7	18	0.14%

112.spmv implements a sparse-matrix, dense-vector multiplication. The input sparse matrix file format is given in coordinate (COO) format, which is internally translated into a transposed jagged diagonal storage (JDS) format before multiplication. The benchmark reflects classes of applications where the sparse matrix remains constant, but is iteratively multiplied into a variety of vectors, allowing the cost of the data format conversion to be amortized over a large number of operations.

114.mriq computes the Q matrix used in non-Cartesian magnetic resonance image reconstruction algorithms [26]. It is used to compensate for artifacts caused by the sampling trajectory on the actual samples recorded. The first kernel preprocesses one of the input sets, and is negligible in the total runtime. The second kernel accumulates contributions from each sample point to each cell in a 3-D regular grid, using a large number of trigonometric operations. The combination of the multiplicative algorithm complexity and the more complex mathematical operations cause this second kernel to dominate the runtime.

116.histo implements a saturating histogram, which is a very large, two-dimensional matrix of char-type bins with a maximum value of 255. The benchmark is customized to a certain class of input, exemplary of a silicon wafer verification application, which follows a nearly Gaussian distribution, roughly centered in the output histogram. The benchmark executes kernels in four phases. It first runs a small kernel on a subset of the input to estimate the centroid of the output distribution. Second, it decomposes the histogram indexes of the input into separate row and column indexes. Work-groups in the third kernel privatize a portion of the histogram locally, and scan the input for items that fall within that region. Finally, the results from all the privatized histograms are combined into the complete results. Each kernel runs very quickly, and the benchmark executes iteratively, representing the streaming analysis application in which it would be deployed. The relatively small runtime for each individual kernel increase the relative impact of kernel launch and device communication overheads in the platform.

117.bfs implements a single-source shortest-path search through a graph using a breadth-first search [20]. The application performs multiple simultaneous searches on the same graph to estimate the average distance between each node in the graph and all other nodes, based on a sampled subset of sources.

118.cutcp computes a cutoff-limited Coulomb potential field for a set of charges distributed in a volume [10]. The application is set to use a cutoff distance of 12 Å, and builds a spatial data structure of the input charges to reduce the number of distance tests that must be performed for each output cell. The field calculation is performed iteratively, reflecting the computational pattern of a typical analysis of the time-averaged field values.

120.kmeans [3] implements the well-known clustering algorithm of data-mining - K-means. In *120.kmeans*, a data object is comprised of several features. By dividing a set of data objects into K clusters, k-means represents all the data objects by the mean values or centroids of their respective clusters. In each iteration, the algorithm associates each data object with its nearest center, based on some chosen distance metric. The new centroids are calculated by taking the

mean of all the data objects within each cluster respectively. As a data intensive application, *120.kmeans* transposes the data matrix before doing clustering for better coalesced memory access. However, this benchmark still stresses the memory bandwidth when many single instruction multiple data (SIMD) compute units access global memory simultaneously.

121.lavamd [29] implements an algorithm of molecular dynamic simulation in 3D space. The code calculates particle potential and relocation due to mutual forces between particles within a large 3D space. This space is divided into cubes, or large boxes, that are allocated to individual cluster nodes. The large box at each node is further divided into cubes, called boxes. 26 neighbor boxes surround each box (the home box). Home boxes at the boundaries of the particle space have fewer neighbors. Cutoff-radius strategy is applied enforcing short-range interaction between particles, which stress communication between neighboring work-item groups. *121.lavamd* requires the communication of boundary elements of each box with it neighbor boxes. On a typical GPU, the inter-work-group communication can only be done via synchronized global-memory-access. This benchmark stresses both memory latency and synchronization.

122.cfd [6] is an unstructured-grid, finite-volume solver for the 3D Euler equations for compressible flow. The Runge-Kutta method is used to solve a differential equation. Effective GPU memory bandwidth is improved by reducing total global memory access and overlapping computation, as well as using an appropriate numbering scheme and data layout. Each time step depends on the results of the previous time step and each time step needs a kernel finalization (an implicit synchronization) and re-launch. This benchmark stresses memory bandwidth and has many kernel launches.

123.nw [3] is a nonlinear global optimization method for DNA sequence alignments - Needleman-Wunsch. The potential pairs of sequences are organized in a 2D matrix. In the first step, the algorithm fills the matrix from top left to bottom right, step-by-step. The optimum alignment is the pathway through the array with maximum score, where the score is the value of the maximum weighted path ending at that cell. Thus, the value of each data element depends on the values of its northwest-, north-, and west-adjacent elements. The first step is parallelized on the GPU. Data blocks in each diagonal strip can be processed in parallel with serial dependency across strips. Blocks are mapped to local memory for data locality. In the second step, the maximum path is traced backward to deduce the optimal alignment. When computation is going on, the workload of each step increases at first and then decreases. At some steps, the computation workload is not enough to fill up all the computation units. In certain phases, the throughput is constrained by *123.nw*'s limited parallelism.

124.hotspot [3,13] is a widely used tool to estimate processor temperature based on an architectural floor plan and simulated power measurements. This benchmark solves a differential equation boundary-value problem on a 2D structured grid by using a finite difference method. Each output cell in the computational grid represents the average temperature value of the corresponding area of the chip.

125.lud [5] implements the well-known LU decomposition for a non-singular matrix. The block-wise operation provides enough parallelism for a GPU-like SIMD device. The degree of block-level parallelism reduces as execution proceeds. *125.lud* utilizes the local memory improve data reuse and coalesced memory access. This benchmark stresses floating point computation units and the compute units' local memory.

126.ge solves linear equations using a row-by-row Gaussian elimination. The algorithm requires synchronization between row-wise iterations, but the values calculated in each iteration can be computed in parallel. This benchmark stresses fine-grained global communication and synchronization with many kernel launches.

127.srad [3,28] implements the speckle reducing anisotropic diffusion (SRAD) method, which is a diffusion method for ultrasonic and radar imaging applications based on partial differential equations (PDEs). It is used to remove locally correlated noise, known as speckles, without destroying important image features. SRAD consists of several pieces of work: image extraction, continuous iterations over the image (preparation, reduction, statistics, computation, and image compression). Each stage requires global synchronization across all the workgroups (kernel calls) before proceeding to the next stage. This benchmark also presents a lot of global memory accesses. This benchmark stresses floating point units, global memory access, and global synchronization.

128.heartwall [28] tracks the movement of a mouse heart over a sequence of ultrasound images to record response to the stimulus. In order to reconstruct approximated full shapes of heart walls, the program generates ellipses that are superimposed over the image and sampled to mark points on the heart walls (Hough search). In its final stage (heart wall tracking presented in Ref. [5]), the program tracks movement of surfaces by detecting the movement of image areas under sample points as the shapes of the heart walls change throughout the sequence of images. The tracking kernel continues dealing with consecutive image frames. This benchmark stress floating point units and memory bandwidth.

140.bplustree [9] traverses B+ trees in parallel, avoiding the overhead of selecting the entire table to transform into row-column format and leveraging the logarithmic nature of tree searches. This benchmark utilizes braided parallelism, running independent queries in each work group concurrently, to avoid the need of global synchronization. It involves irregular memory access and therefore stresses memory bandwidth and latency.

4.2 SPEC ACCEL ACC Suite

The OpenACC suite consists of 15 applications. Some applications are direct ports from the OpenCL suite, others have been ported from the SPEC OMP2012 suite. A number of numerical aerodynamic simulation (NAS) parallel benchmarks as well as a few novel applications are included as well.

Similar to the OpenCL suite, the OpenACC suite of the SPEC ACCEL benchmarks also tries to stress the hardware accelerator ecosystem in various ways. The applications vary between few (*314.omriq*) or lots of accelerator

Table 2. OpenACC application key facts. Profiling data taken from PGI OpenACC runtime using an NVIDIA Tesla K40 and the CUDA 5.5 backend

Application	Lines of code	Language	Area
303.ostencil	796	C	Thermodynamics
304.olbm	923	C	Computational fluid Dynamics, Lattice Boltzmann method
314.omriq	693	C	Medicine
350.md	2,479	Fortran	Molecular dynamics
351.palm	48,583	Fortran	Large-eddy simulation, atmospheric turbulence
352.ep	480	C	Random number generation
353.clvrleaf	6,477	C, Fortran	Explicit hydrodynamics
354.cg	638	C	Conjugate gradient
355.seismic	750	Fortran	Seismic wave modeling
356.sp	2,693	Fortran	Scalar Penta-diagonal solver
357.csp	2,364	C	Scalar Penta-diagonal solver
359.miniGhost	6,334	C, Fortran	Finite difference
360.ilbdc	1,065	Fortran	Fluid Mechanics
363.swim	249	Fortran	Weather: shallow water modeling
370.bt	5,524	C	Block tridiagonal solver for 3D PDE

Application	Kernel invocations	# of regions	Accel. usage	MiBytes transferred	# of transfers	Transfer time
303.ostencil	20,000	20,000	92.2%	392	28	0.07%
304.olbm	5,000	5,000	96.2%	16,562	1,053	0.72%
314.omriq	2	2	99.1%	112	12	0.00%
350.md	607	607	97.3%	1,019	2,428	0.01%
351.palm	50,185	46,584	35.4%	102,438	26,377	5.02%
352.ep	1,760	1,760	97.0%	0.027	2,345	0.00%
353.clvrleaf	256,021	203,821	94.4%	3,445	9,678	0.21%
354.cg	13,234	13,234	80.4%	1,173	10,781	0.08%
355.seismic	10,402	3,201	92.8%	6,142	4,431	0.66%
356.sp	27,691	27,691	95.1%	657	46	0.04%
357.csp	26,087	26,087	95.4%	657	46	0.04%
359.miniGhost	72,040	64,040	73.1%	1,622	112,124	1.14%
360.ilbdc	5,000	5,000	88.7%	1,925	121	0.42%
363.swim	90,999	26,000	44.2%	198,563	26,000	40.0%
370.bt	8,051	8,051	95.3%	206	15	0.02%

regions (*353.clvrleaf*), as well as one (most of the applications) or multiple kernel invocation per accelerator region (*363.swim*). In the same manner, the amount and number of data transfers between the host and device differ. At the moment a large amount of data transfers also results in a poorer accelerator utilization. This can, however, change for future OpenACC implementations that better overlap computation and transfer.

The included applications cover a wide area of scientific domains and computational schemes:

303.ostencil is an iterative Jacobi solver of the heat equation on a 3-D structured grid, which can also be used as a building block for more advanced multigrid PDE solvers. This code it ported from the serial version of 103.stencil from Parboil. While the accelerated loop is a fairly simple stencil operation, the code shares the same workload as 103.stencil and offers a way to directly compare an OpenCL and OpenACC implementation of the same code.

304.olbm, like 104.lbm, is ported from the SPEC CPU2006 benchmark and uses the Lattice Boltzmann Method (LBM) to simulate incompressible fluids in 3D. The accelerated portion of the code is a more complex 19-point stencil which stresses the accelerator's global memory and potential cache infrastructure.

314.omriq simulates magnetic resonance imaging (MRI) image reconstruction by converting sampled radio responses into magnetic field gradients. This is a port of the serial version of 114.mriq also from Parboil and uses the same workload. The accelerated loop is fairly small but includes an inner loop reduction, use of `cos` and `sin` functions, and due to the use of an array of structs, some memory accesses are not coalesced. Non-coalesced memory accesses are generally not well suited for accelerators, but are often found in complex applications.

350.md was written at Indiana University to perform molecular dynamics simulations of dense nuclear matter such as those occurring in Type II supernovas, the outer layers of neutron stars, and white dwarf stars [12]. While an earlier version of this code appears in the SPEC OMP2012 benchmark suite, this version has been updated to better utilize the massive parallelization available with accelerators.

351.palm is a large-eddy simulation (LES) model for atmospheric and oceanic flows from Leibniz University of Hannover [25]. It solves prognostic equations for velocity (Navier-Stokes equation), temperature (first law of thermodynamics), and humidity (transport equation for scalar). 351.palm is the largest and most complex of the codes in SPEC ACCEL and best represents how large scale applications can utilize accelerators. The source code includes a host implementation of the Temperton fast Fourier transform (FFT) routines which dominates the compute time spent on the host. However, for the peak metric, an optimized host or accelerated Fastest Fourier Transform in the West (FFTW) library may be used.

352.ep is from the University of Houston and is a port of the embarrassing parallel (EP) benchmark from the NAS Parallel Benchmark (NPB) suite [2]. The port required the use of a blocking algorithm since the entire problem size could not fit within the 2 GB memory limit set in SPEC ACCEL. The benchmark also tests the use of reductions. [17–19]

353.clvleaf is the CloverLeaf [11] mini-application which is used to solve the compressible Euler equations on a Cartesian grid, using an explicit, second-order method.

354.cg is NPB's conjugate gradient (CG) OpenMP benchmark ported to OpenACC by the University of Houston. This benchmark uses the inverse power

method to find an estimate of the largest eigenvalue of a symmetric positive definite sparse matrix with a random pattern of nonzeros. The code required few changes from the OpenMP version. [17,18].

355.seismic is ported from University of Pau's SEISMIC_CPML perfectly matched layer (PML) Collino 3D isotropic solver [8], a 3D classical split PML program for an isotropic medium using a second-order, finite-difference spatial operator, for comparison. The code was originally ported to OpenACC for use in tutorials, but due to the minimal number of OpenACC directives used, highlights a compiler's ability to schedule loops and perform reduction operations.

356.sp and *357.csp* are both derived from NPB's singal processing (SP) benchmark, using different languages. Although they do both solve the same problem using the same data set, the SPEC HPG committee thought having both would give a good comparison of using OpenACC with Fortran versus C. The SP benchmark solves a synthetic system of partial differential equations using a penta-diagonal matrix.

359.miniGhost is a finite difference mini-application from Sandia National Laboratory [24] used to test a broad range of stencil algorithms on accelerators. The code also performs inter-process boundary (halo, ghost) exchange and global summation of grid values.

360.ilbdc is an OpenACC port from SPEC OMP2012 [21] and is geared to the collision-propagation routine of an advanced 3-D lattice Boltzmann flow solver using a two-relaxation-time (TRT-type) collision operator for the D3Q19 model. The code uses a similar algorithm to 304.lbm although written in Fortran and uses a minimal number of OpenACC directives.

363.swim is also ported from SPEC OMP2012 and is a finite-difference approximation of the shallow-water equations. Because the data is printed after each time step, the benchmark highlights the cost of moving data between the accelerator and the host which also includes the data movement between the hosts application user memory space and the accelerator driver memory space.

370.bt is NPB's BT benchmark ported to OpenACC. Like SP, it solves a synthetic system of partial differential equations, but instead uses a block tridiagonal matrix.

5 Energy Awareness

Computer systems using hardware accelerators are seen as one method for more energy efficient data processing. The SPEC ACCEL benchmark suites take that into account by providing the same power measurement capabilities as the previously released SPEComp2012 suite. As a result, the energy consumption can be recorded during a measurement run as well. Recording energy consumption is not mandatory but encouraged.

Energy measurement is enabled by changing the `power` setting in the configuration file for the measurement run to `yes` and setting up power and temperature measurement daemons (PTDaemon). The SPEC runtime system then connects to these daemons and continuously samples the energy consumption of the whole

system under test every second and the air intake temperature every five seconds. The SPEC ACCEL run rules define how the energy measurement needs to be set up. The power analyzers need to be calibrated in the last 12 months to ensure the energy measurement accuracy. The temperature is measured to prevent reducing the energy consumption by running the system under test at unusually low temperatures – a valid run needs to be carried out with at least 20 °C air intake temperature. The PTDaemon can connect to a variety of power meters and temperature probes and offers range checking, uncertainty calculation, and multi-channel measurements. The SPEC runtime system ensures that at least 99 % of all power samples are reported as valid samples by the PTDaemon. Otherwise, it will abort the run or mark it as invalid.

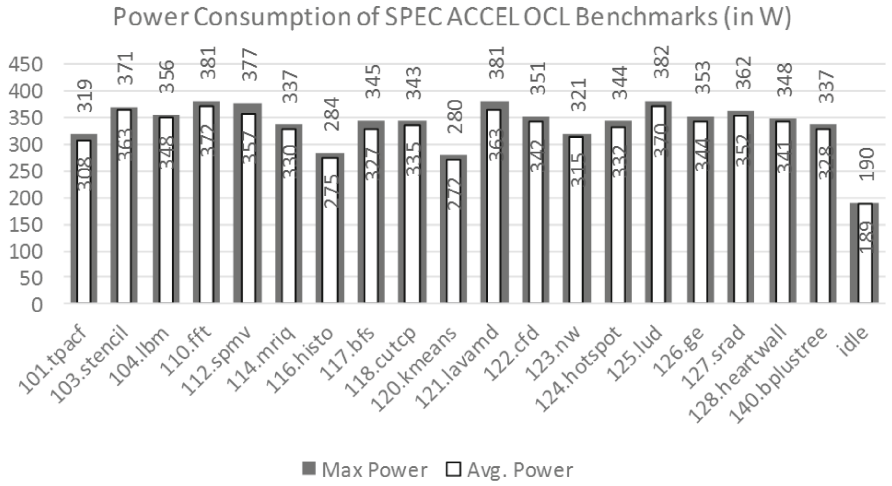
When the SPEC ACCEL benchmark is run with energy measurement enabled, it will generate two additional metrics per suite:

`SPECaccel_{acc|ocl}_energy_{base|peak}`.

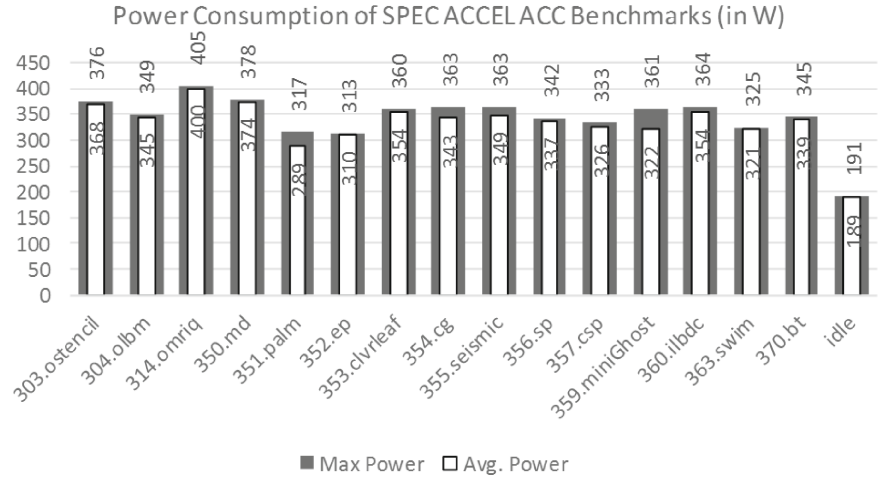
Similar to the standard metrics, the energy metrics compare the energy consumption of the system under test to the energy consumption of the reference system. A higher number indicates a lower energy consumption or better energy efficiency. Energy for this metric means power consumption integrated over time, hence an energy metric of 2 indicates that the system under test consumed half the energy (measured in Joules) than the reference system on the benchmark. As a result, the SPEC ACCEL energy metrics can be used for an energy-to-solution comparison. While the standard SPEC ACCEL metrics provide a measurement for time-to-solution, they may be used in combination to determine the reason why a system under test consumes more or less energy than the reference system. A SPEC_ocl_base rating of 2 and a SPEC_ocl_energy_base rating of 2 indicate that the system under test ran the benchmarks twice as fast as the reference system, but on average consumed the same amount of power. A SPEC_ocl_base rating of 1 and a SPEC_ocl_energy_base rating of 2 indicate that the system under test ran the benchmark in the same time as the reference system, but used on average half the power. In total, both systems consumed half the energy than the reference system, thus, running the benchmark induces only half the energy costs.

The report for a benchmark run lists the consumed energy, the maximum power usage, the average power usage, and the energy ratio for each individual benchmark. The idle power consumption can be taken from the log-file of the benchmark run. Figure 1 shows that the maximum and average power consumption varies quite a lot between benchmarks. The power measurement can be used to indirectly deduce the behavior of the various benchmarks:

- A benchmark with low maximum and average power consumption is mainly data transfer bound since both the host and the device are idle during the transfers.
- A benchmark with a high maximum and average power is largely device bound. There can be both compute or memory access activity on the device.



(a) OpenCL benchmarks



(b) OpenACC benchmarks

Fig. 1. Maximum and average power consumption for all benchmarks as well as idle power consumption when running both SPEC ACCEL suites on the reference system

– A benchmark with a significantly higher maximum than average power consumption has both: phases with lots of data transfers, but also device bound phases resulting in a high variation in power consumption.

6 Discussion of First Results

A run of SPEC ACCEL produces a number of output files in the `result` subdirectory. It writes a logfile of the benchmark run – in case of any errors also a more detailed debug log – as well as text and raw output for each data set (test, train, or ref) it was run on. The runtime and energy consumption of the benchmarks, when executed on the reference machine serve as the basis for normalization. If your SPEC rate is larger than 1, this indicates that your system performs better at running the workload of the selected benchmark suite than the reference system. As a result, the single metric enables a first method of comparing hardware platforms and software environments. The text output for the ref data set also allows a benchmark-by-benchmark comparison with the published results, as shown in Table 3. In this result, it can be seen that not all benchmarks benefit equally from the more modern accelerator. 120.kmeans, for example, only shows an 11 % performance increase while 114.mriq runs over three times as fast. In a similar manner, one can see that this hardware platform is more energy efficient than the reference system and requires less than half the energy to run the suite (as indicated by the SPECaccel_ocl_energy_base value).

The SPEC tool `rawformat` can produce reports from a measurement run that are comparable to the results officially published on the SPEC website. It shows the results from all runs of the ref data set, so that run-to-run variation can be examined as well. The SPEC tools also run tests to determine the hardware and software configuration to aid the gathering of all performance relevant data about the setup of the system under test.

In order to share your results with others on the SPEC website, a reportable run must be done. This will invoke the benchmark suite with the test and train data set once, and the ref data set at least three times. The `rawformat` tool checks for missing system setup information in the result file. One very common issue is a lack of compiler flag description. SPEC requires an xml-based description of all used compiler commands and compiler flags. A result that has been submitted for publication is peer reviewed by HPG members in order to ensure compliance of the benchmark result with the run rules. The review process also ensures that the result contains all information necessary to reproduce the measurement. All published results have passed multiple stages of checking, verification, and cross-checking, thus, serving as a sustainable source for performance data.

A published result is split into multiple sections[2]:

- The header lists the hardware vendor, the used accelerator, and the system name, along with results in all four metrics of the benchmark. It also lists who ran the benchmark, when it was carried out, and when the used hardware and software components are available.
- It is followed by a diagram that shows the distribution of the individual benchmarks. The distribution provides insight into which applications perform well

[2] The reference result for the OpenCL suite is available at http://spec.org/accel/results/res2014q1/accel-20140228-00006.html and for the OpenACC suite at http://spec.org/accel/results/res2014q1/accel-20140228-00005.html.

Table 3. SPEC ACCEL OpenCL results for an ASUS P9X79 Motherboard with an Intel Core i7-3930K and an NVIDIA Tesla K40c (ECC enabled) using base optimizations.

Benchmarks	Ref. time	Run time	Ratio	Energy	Max power	Average power	Energy ratio
101.tpacf	107	67.7	1.58	15.3	241	225	2.14
103.stencil	125	61.6	2.03	17.5	296	284	2.59
104.lbm	112	43.4	2.58	12.2	289	280	3.16
110.fft	111	76.0	1.46	22.9	316	302	1.79
112.spmv	147	79.0	1.86	21.8	293	276	2.41
114.mriq	109	33.2	3.28	8.49	271	256	4.25
116.histo	114	80.8	1.41	16.0	216	198	1.95
117.bfs	117	59.2	1.98	14.7	266	248	2.59
118.cutcp	99	34.4	2.88	9.01	273	262	3.68
120.kmeans	100	90.1	1.11	18.0	211	199	1.50
121.lavamd	109	60.2	1.81	17.3	307	288	2.28
122.cfd	126	73.3	1.72	19.1	273	260	2.26
123.nw	115	69.8	1.65	16.0	237	229	2.26
124.hotspot	114	38.7	2.95	10.9	303	281	3.48
125.lud	119	80.9	1.47	22.8	295	282	1.93
126.ge	155	54.1	2.86	14.3	280	265	3.74
127.srad	114	60.7	1.88	16.9	292	278	2.36
128.heartwall	106	88.0	1.20	21.7	255	247	1.66
140.bplustree	108	70.0	1.54	17.3	257	247	2.05
SPECaccel_ocl_energy_base							2.43
SPECaccel_ocl_base			1.87				

or not so well on the system under test. The bars also have ticks for all runs of the ref data set so that run-to-run variation is also easily visible.

- The system description section lists the host and accelerator's hardware properties along with the software set up.
- With energy measurement enabled, the next section shows the properties of its setup including power supply, power analyzer used, and the temperature probe.
- The result table(s) lists the execution time and the ratio for each iteration of every benchmark, as well as the energy measurement results (if energy measurement is enabled).
- The notes section shows the output from the SPEC sysinfo tool and any custom notes by the submitter of the result.

Table 4. SPEC ACCEL OpenACC results for an ASUS P9X79 Motherboard with an Intel Core i7-3930K and an NVIDIA Tesla K40c (ECC enabled) running at various GPU clock frequencies using base optimizations

	745 MHz		810 MHz			
Benchmarks	**Ratio**	**ERatio**	**Ratio**	**Speedup**	**ERatio**	**ESaving**
303.ostencil	2.60	3.09	2.96	13%	3.27	5%
304.olbm	1.99	2.61	2.20	9%	2.74	5%
314.omriq	2.37	2.96	2.57	8%	2.99	1%
350.md	2.31	2.97	2.59	11%	3.05	3%
351.palm	1.88	2.50	1.97	5%	2.57	2%
352.ep	1.36	1.80	1.49	9%	1.91	5%
353.clvrleaf	2.65	3.37	2.94	10%	3.48	3%
354.cg	2.50	3.24	2.74	9%	3.40	5%
355.seismic	2.38	3.20	2.64	10%	3.37	5%
356.sp	2.04	2.65	2.26	10%	2.78	5%
357.csp	1.65	2.16	1.82	10%	2.28	5%
359.miniGhost	2.17	2.82	2.69	19%	3.25	13%
360.ilbdc	3.11	4.10	3.64	14%	4.45	8%
363.swim	2.31	3.14	2.50	7%	3.25	4%
370.bt	2.50	3.35	2.81	11%	3.58	7%
Overall	**2.21**	**2.88**	**2.47**	**10%**	**3.03**	**4%**

	745 MHz		875 MHz			
Benchmarks	**Ratio**	**ERatio**	**Ratio**	**Speedup**	**ERatio**	**ESaving**
303.ostencil	2.60	3.09	3.17	18%	3.16	2%
304.olbm	1.99	2.61	2.35	15%	2.78	6%
314.omriq	2.37	2.96	2.70	12%	2.91	-2%
350.md	2.31	2.97	2.78	17%	2.97	0%
351.palm	1.88	2.50	2.01	7%	2.54	2%
352.ep	1.36	1.80	1.61	16%	1.96	8%
353.clvrleaf	2.65	3.37	3.07	14%	3.44	2%
354.cg	2.50	3.24	2.84	12%	3.37	4%
355.seismic	2.38	3.20	2.79	15%	3.38	5%
356.sp	2.04	2.65	2.35	13%	2.74	3%
357.csp	1.65	2.16	1.89	13%	2.25	4%
359.miniGhost	2.17	2.82	2.80	22%	3.21	12%
360.ilbdc	3.11	4.10	3.77	18%	4.37	6%
363.swim	2.31	3.14	2.60	11%	3.25	3%
370.bt	2.50	3.35	2.95	15%	3.56	6%
Overall	**2.21**	**2.88**	**2.59**	**14%**	**3.01**	**4%**

– The compiler section lists the compiler(s) and compiler flags used for every individual application in the suite. It also provides a link to the previously mentioned flags file explaining the compiler settings in more detail.

Among the initially submitted results from the SPEC ACCEL OpenACC suite is an experiment on how different GPU clock frequency affects application per-

Table 5. SPEC ACCEL OpenACC results for an ASUS P9X79 Motherboard with an Intel Core i7-3930K and an NVIDIA Tesla K40c using base optimizations with ECC enabled and disabled

Benchmarks	ECC enabled		ECC disabled			
	Ratio	ERatio	Ratio	Speedup	ERatio	ESaving
303.ostencil	2.60	3.09	2.67	2.7 %	3.19	3.2 %
304.olbm	1.99	2.61	4.37	120 %	5.62	115 %
314.omriq	2.37	2.96	2.86	20.7 %	3.45	16.6 %
350.md	2.31	2.97	2.35	1.7 %	3.00	1.0 %
351.palm	1.88	2.50	1.96	4.3 %	2.62	4.8 %
352.ep	1.36	1.80	1.37	0.7 %	1.81	0.6 %
353.clvrleaf	2.65	3.37	2.98	12.5 %	3.72	10.4 %
354.cg	2.50	3.24	2.60	4.0 %	3.43	5.9 %
355.seismic	2.38	3.20	2.55	7.1 %	3.43	7.2 %
356.sp	2.04	2.65	2.45	20.1 %	3.19	20.4 %
357.csp	1.65	2.16	1.91	15.8 %	2.51	16.2 %
359.miniGhost	2.17	2.82	2.84	30.9 %	3.62	28.4 %
360.ilbdc	3.11	4.10	4.09	31.5 %	5.21	27.1 %
363.swim	2.31	3.14	2.46	6.5 %	3.35	6.7 %
370.bt	2.50	3.35	2.80	12.0 %	3.79	13.1 %
Overall	**2.21**	**2.88**	**2.59**	**22.7 %**	**3.35**	**16.3 %**

formance. The experiment uses the GPU Boost capabilities of the NVIDIA K40c GPU where the clock speed can be increased from the default 745 MHz to 810 and 875 MHz. The results are shown in Table 4. All benchmarks benefit from the increased GPU clock rate and none consume more energy to run the applications. Since the energy savings are less than the performance gain, the system actually draws more power, but over a shorter period of time. Increasing the GPU's clock speed also helps with memory bandwidth efficiency, hence, some benchmarks see improvements greater than the clock boost. Other benchmarks see less performance since they either have a high percentage of time spent on the host (351.palm, 354.cg) or have higher memory transfer rate between the host and device (363.swim). As a result of this study, a site such as Oak Ridge or or National Center for Supercomputing Applications (NCSA) could decide to increase the GPU clock rate by default since a broad range of applications benefit from it (reduced runtime) without extra costs (same or less energy consumption).

Another widely discussed question that can be answered with the currently published results is the impact of ECC on accelerator performance. Table 5 shows the results for the SPEC ACCEL OpenACC benchmarks with ECC turned on and off. As expected, performance improvements, due to the increased memory bandwidth when ECC is disabled, actually vary by a very large amount for the

applications used. On average, disabling ECC yields a performance increase of 22.7 %, and the energy consumption also slightly improves due to the reduced computing times. Whether this nominal performance increase is worth the risk of wrong results is a different discussion. Within the SPEC harness, the result verification routine ensures that the applications generate the expected results.

7 Summary and Future Work

SPEC HPG set out to develop a performance measurement environment based on the SPEC principles for hardware accelerators. As a result, two application suites – one with OpenCL and one with OpenACC applications – have been released with SPEC ACCEL. They deliver performance and energy consumption metrics that enable comparing hardware devices and software environments. The goals set by HPG for the development of these application suites are met. The metrics reflect the impact of different hardware and hardware settings, but also show how different software environments (e.g., compilers, runtimes) affect application performance. The mix of selected applications also demonstrates that not all applications react in a similar manner to such a change. The suites can also serve as a yardstick for determining the best hardware and software for solving particular scientific problems. Furthermore, the suites have already been used by compiler and runtime vendors as a mean for verification of the developed software stacks.

SPEC ACCEL is set apart from other accelerator benchmarks for hardware accelerators since it is simple to run, yet has a performance evaluation process that uses real world applications under a strict measurement environment and a peer review process for published results. Furthermore, the energy consumption metric enables comparison between results not only by runtime of the applications, but also energy consumed.

HPG plans to extend SPEC ACCEL with a third suite covering OpenMP 4.0 target directives in the near future. The OpenACC applications will be ported to support OpenMP 4.0 target directive so that devices that are currently not supported by OpenACC may be compared to devices that are. Beyond that effort, SPEC HPG is investigating future updates to the various suites to support more current versions of OpenCL and OpenACC.

Acknowledgments. The authors thank Cloyce Spradling for his work on the SPEC harness as well as the SPEC POWER group for their work on enabling the integration of power measurements into other SPEC suites.

Contributions by the University of Houston were supported in part by NVIDIA and Department of Energy under Award Agreement No. DE-FC02-12ER26099.

References

1. The OpenACC Application Programming Interface, November 2011. http://www.openacc.org/sites/default/files/OpenACC.1.0_0.pdf, version 1.0
2. Bailey, D., Barszcz, E., Barton, J., Browning, D., Carter, R., Dagum, L., Fatoohi, R., Fineberg, S., Lasinski, T., Schreiber, R., Simon, H., Venkatakrishnan, V., Weeratunga, S.: The NAS parallel benchmarks. Technical report RNR-94-2007, NASA (1994). http://www.nas.nasa.gov/assets/pdf/techreports/1994/rnr-94-007.pdf
3. Che, S., Boyer, M., Meng, J., Tarjan, D., Sheaffer, J.W., Skadron, K.: A performance study of general-purpose applications on graphics processors using CUDA. J. Parallel Distrib. Comput. **68**(10), 1370–1380 (2008). http://dx.doi.org/10.1016/j.jpdc.2008.05.014
4. Che, S., Boyer, M., Meng, J., Tarjan, D., Sheaffer, W.J., Lee, S.H., Skadron, K.: Rodinia: a benchmark suite for heterogeneous computing. In: Proceedings of the IEEE International Symposium on Workload Characterization (IISWC), pp. 44–54, October 2009
5. Che, S., Sheaffer, W.J., Boyer, M., Szafaryn, L.G., Wang, L., Skadron, K.: A characterization of the rodinia benchmark suite with comparison to contemporary CMP workloads. In: Proceedings of the IEEE International Symposium on Workload Characterization (IISWC), December 2010
6. Corrigan, A., Camelli, F., Lohner, R., Wallin, J.: Running unstructured grid CFD solvers on modern graphics hardware. In: Proceedings of the 19th AIAA Computational Fluid Dynamics Conference, June 2009
7. Danalis, A., Marin, G., McCurdy, C., Meredith, J.S., Roth, P.C., Spafford, K., Tipparaju, V., Vetter, J.S.: The scalable heterogeneous computing (SHOC) benchmark suite. In: Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units, GPGPU 2010, pp. 63–74. ACM, New York (2010). http://doi.acm.org/10.1145/1735688.1735702
8. Komatitsch, D., Martin, R.: University of Pau: SEISMIC_CPML. http://geodynamics.org/cig/software/seismic_cpml/
9. Fix, J., Wilkes, A., Skadron, K.: Accelerating braided B+ tree searches on a GPU with CUDA. In: Proceedings of the 2nd Workshop on Applications for Multi and Many Core Processors: Analysis, Implementation, and Performance (A4MMC), in Conjunction with ISCA, June 2011
10. Hardy, D.J., Stone, J.E., Vandivort, K.L., Gohara, D., Rodrigues, C., Schulten, K.: Fast molecular electrostatics algorithms on GPUs. In: GPU Computing Gems (2010)
11. Herdman, J., Gaudin, W., McIntosh-Smith, S., Boulton, M., Beckingsale, D., Mallinson, A., Jarvis, S.: Accelerating hydrocodes with OpenACC, OpeCL and CUDA. In: 2012 SC Companion: High Performance Computing, Networking, Storage and Analysis (SCC), pp. 465–471, November 2012
12. Horowitz, C.J., Berry, D.K., Brown, E.F.: Phase separation in the crust of accreting neutron stars. Phys. Rev. E **75**, 066101 (2007). http://link.aps.org/doi/10.1103/PhysRevE.75.066101
13. Huang, W., Ghosh, S., Velusamy, S., Sankaranarayanan, K., Skadron, K., Stan, M.: HotSpot: a compact thermal modeling methodology for early-stage VLSI design. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **14**(5), 501–513 (2006)
14. Johnson, N.: EPCC OpenACC benchmark suite. https://www.epcc.ed.ac.uk/research/computing/performance-characterisation-and-benchmarking/epcc-openacc-benchmark-suite

15. Khronos Group: OpenCL 1.1 API and C Language Specification, June 2011. https://www.khronos.org/registry/cl/specs/opencl-1.1.pdf, revision 44
16. Lange, K.D.: Identifying shades of green: the SPECpower benchmarks. Computer **42**, 95–97 (2009)
17. Lee, S., Eigenmann, R.: OpenMPC: extended OpenMP programming and tuning for GPUs. In: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–11. IEEE Computer Society (2010)
18. Lee, S., Min, S.J., Eigenmann, R.: OpenMP to GPGPU: a compiler framework for automatic translation and optimization. ACM Sigplan Not. **44**(4), 101–110 (2009)
19. Lee, S., Vetter, J.S.: Early evaluation of directive-based gpu programming models for productive exascale computing. In: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, p. 23. IEEE Computer Society Press (2012)
20. Luo, L., Wong, M., Hwu, W.W.: An effective GPU implementation of breadth-first search. In: Proceedings of the 47th Design Automation Conference, pp. 52–55, June 2010
21. Müller, M.S., et al.: SPEC OMP2012 — an application benchmark suite for parallel systems using OpenMP. In: Chapman, B.M., Massaioli, F., Müller, M.S., Rorro, M. (eds.) IWOMP 2012. LNCS, vol. 7312, pp. 223–236. Springer, Heidelberg (2012). http://dx.doi.org/10.1007/978-3-642-30961-8_17
22. Müller, M.S., van Waveren, M., Lieberman, R., Whitney, B., Saito, H., Kumaran, K., Baron, J., Brantley, W.C., Parrott, C., Elken, T., Feng, H., Ponder, C.: SPEC MPI2007 - an application benchmark suite for parallel systems using MPI. Concurr. Comput. Pract. Exper. **22**(2), 191–205 (2010). http://dx.doi.org/10.1002/cpe.v22:2
23. Qian, Y.H., D'Humieres, D., Lallemand, P.: Lattice BGK models for navier-stokes equation. Europhys. Lett. **17**, 479–484 (1992)
24. Barrett, R.F., Vaughan, C.T., Heroux, M.A.: MiniGhost: A miniapp for exploring boundary exchange strategies using stencil computations in scientific parallel computing, Version 1.0. Techical report (2012)
25. Raasch, S.: Leibniz University of Hannover: PALM. http://palm.muk.uni-hannover.de/
26. Stone, S.S., Haldar, J.P., Tsao, S.C., Hwu, W.W., Liang, Z., Sutton, B.P.: Accelerating advanced MRI reconstructions on GPUs. In: International Conference on Computing Frontiers, pp. 261–272 (2008)
27. Stratton, J.A., Rodrigues, C., Sung, I.J., Obeid, N., Chang, L., Liu, G., Hwu, W.W.: Parboil: a revised benchmark suite for scientific and commercial throughput computing. Technical report IMPACT-12-01. University of Illinois at Urbana-Champaign, Urbana, March 2012
28. Szafaryn, L.G., Skadron, K., Saucerman, J.J.: Experiences accelerating MATLAB systems biology applications. In: Proceedings of the Workshop on Biomedicine in Computing: Systems, Architectures, and Circuits (BiC) 2009, in Conjunction with the 36th IEEE/ACM International Symposium on Computer Architecture (ISCA), June 2009
29. Szafaryn, L.G., Gamblin, T., de Supinski, B.R., Skadron, K.: Trellis: portability across architectures with a high-level framework. J. Parallel Distrib. Comput. **73**(10), 1400–1413 (2013)