Chapter 18
HPC-Smart Infrastructures: A Review and Outlook on Performance Analysis Methods and Tools

Thaha Muhammed, Rashid Mehmood, Aiiad Albeshri, and Fawaz Alsolami

18.1 Introduction

High-performance computing (HPC) plays a vital role in driving transformations across various smart-city infrastructures such as healthcare, agriculture, environment, and other infrastructures [94]. It is a vital cog in autonomous adaption of urban infrastructure to various events and stimuli (e.g., severe hurricane, high traffic due to accidents). HPC is a major component in developing phenomenal computationally intensive models for various smart-city infrastructures.

Driving high efficiency from shared-memory and distributed-memory HPC systems have always been challenging. Big data and HPC convergence, system heterogeneity, cloud computing, and many other developments have increased the complexities of HPC systems [36, 51, 77, 108, 124]. There are increasing pressures on energy efficiency for developing exascale computers and therefore development of highly efficient HPC applications and systems has become essential.

Various performance analysis tools exist that help in improving the performance and efficiency of HPC scientific applications and increase their potential. Performance analysis is a crucial part in the development of the HPC applications. Performance optimization is not just identifying the bottlenecks in the code but also identifying the causes of bottlenecks and the required changes that need to be made to the parallel applications [99]. This requires more advanced tools such as hardware performance counters. Diagnosing the problems manually requires

T. Muhammed (✉) · A. Albeshri · F. Alsolami
Department of Computer Science, FCIT, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: m.thaha.h@ieee.org; aaalbeshri@kau.edu.sa; falsolami1@kau.edu.sa

R. Mehmood
High Performance Computing Center, King Abdulaziz University, Jeddah, Saudi Arabia
e-mail: rmehmood@kau.edu.sa

© Springer Nature Switzerland AG 2020
R. Mehmood et al. (eds.), *Smart Infrastructure and Applications*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-13705-2_18

deep knowledge about the architecture, hardware of the system, and the compiler. Performance analysis is important to determine the different optimization strategies for the same application on different HPC platforms such as GPU, MIC, cloud, and MPI-based grids.

Corresponding to debugging and testing, performance analysis and optimization of HPC applications are vital stages in the development cycle. It is a crucial condition for assuring efficient use of costly and limited resources. The performance analysis phase evaluates the actual performance (speed of computation, throughput, and resource consumption) on a given platform with regard to memory, storage, network, and runtime. Moreover, it has to identify improvements and reduction in the usage of resources.

This paper reviews the performance analysis tools and techniques for HPC applications and systems. The contributions of this article can be summarized below.

1. A review of the tools for the performance analysis of HPC applications. The works on the HPC performance analysis are numerous and we do not claim to be exhaustive in this paper.
2. A discussion on the performance of various HPC applications on a number of HPC platforms.
3. A discussion on the common HPC applications used by the researchers and HPC benchmarking suites for analysis.
4. A qualitative comparison of various tools used for the performance analysis of HPC applications is provided.
5. A discussion on the future research directions and issues.

The rest of the paper is organized as follows. Section 18.2 describes various Benchmark toolkits and various HPC applications that are used for the performance analysis. Section 18.3 presents existing work by various researchers in analyzing the performance of various HPC applications. Section 18.4 provides a qualitative comparison of various tools used for HPC application performance analysis. Future research issues and directions are provided in Sect. 18.5. Finally, Sect. 18.6 concludes the paper.

18.2 HPC Applications and Benchmarking Suites

In this section, we discuss various HPC-based applications from various domains which are prone to performance problems. Table 18.1 summarizes various HPC applications that are used in various application domains. We shall discuss some major domains in which HPC is a necessity and is used abundantly.

– **Automobile and Aeronautics:** This field has a lot of simulation and modeling, model prediction and verification including probabilistic modeling, computer aided drawing, graphic designing, design automation, the design of structures,

Table 18.1 A summary and comparison of commonly used HPC applications

Application	Domain	Language	Developers	OS	Open
BigDFT	Chemistry	F90	Genovese et al. [42]	Li/Un	Yes
Bifrost	Atmosphere	F90	Gudiksen et al. [46]	Li/Un	No
ChaNGa	Cosmology	Charm++	Jetley et al. [55]	Li/Un	Yes
COSMO	Weather	C++	CINECA	Li/Un	Yes
CORSIKA	Astrophysics	F77/F90	Heck et al. [50]	Li/Un	No
ECHAM/MESSy	Environment	F77/F90	Jöckel et al. [57]	Li/Un	No
EUTERPE	Fusion	C++/C	Saez et al. [110]	Li/Un	No
Gamess	Material Science	F77	Schmidt et al. [45, 112]	Li/Un	No
IBM WATSON	Graph Analysis	C++	IBM	Li/Un	No
IMPACT-T	Math. modeling	F90	Qinag et al. [107]	Li/Un	Yes
Jacobi2D	Math. modeling	Charm++	–	Li/Un	Yes
LIBMESH	Math. modeling	C/C++	Kirk et al. [64]	Li/Un	Yes
MAESTRO	Astrophysics	F90	Nonaka et al.[100]	Li/Un	Yes
MILC	Quantum Theory	C/C++	Bailey et al. [20]	Li/Un	Yes
MP2C	Particle collision	F90	Freche et al. [39]	Li/Un	No
NAMD	Chemistry	Charm++	Bhatele et al. [23]	Li/Un	Yes
NQueens	Backtracking	C/C++	–	Li/Un	Yes
OpenFOAM	Fluid dynamics	C++	Jacobsen et al. [54]	Li/Un	Yes
Paratec	Quantum theory	C/C++	Pfrommer et al. [104]	Li/Un	No
PEPC	Gravitation	F2003	Gibbon [44]	Li/Un	Yes
ProFASI	Protein structure	C++	Irbäck et al. [52]	Li/Un	No
PRISM	Probab. modeling	Java/C	Kwiatkowska et al. [72]	Li/Win	Yes
Quantum expresso	Molecular structure	F90	Giannozzi et al. [43]	Li/Win	Yes
SIMONA	Nano science	C++	Strunk et al. [116]	Li/Un	No
SMMP	Protein structure	F90	Meinke et al. [90]	Li/Un	Yes
SPECFEM3D	Wave propagation	F90/C	Dimitri et al. [67]	Li/Un	Yes
Sweep3D	Material science	F77	Wylie et al.. [128]	Li/Un	Yes
YALES2	Combustion	C++/F90	Moureau et al. [91]	Li/Un	No
WIEN2K	Chemistry	F90	Schwarz et al. [113]	Li/Un	No

Li Linux, *Un* Unix, *Win* Windows

automated plan building, analysis of design, and concrete modeling [66, 73, 98, 130].

– **Astrophysics and quantum physics:** A lot of applications based on physics, especially on quantum physics and astrophysics, has very large computations as they receive a large input data. Load balancing of spin-image algorithm on CPU and MIC has been studied in [22, 34].
– **Biosciences:** Biosciences including bioinformatics have a large number of programs that require computation including the mathematical modeling of diseases. It also has issues with memory management. See, for example, [7].
– **Earth sciences:** A large number of earth related activities such as earthquake prediction, monitoring, weather prediction, and prediction of climate change due

to global warming needs high computation [61, 109]. These applications are highly data intensive and take days to run on serial machines.

- **Electronics:** The design and analysis of electronic components have a high computation due to the simulation and modeling before the actual production [117, 129]. Other things include the static timing analysis and lithography.
- **Material sciences:** Material science includes modeling of nanoscale particle, the modeling behavior of nanoscale particle, and modeling of molecules and chemical reactions [2, 74]. These require a lot of computation and memory.
- **Computational fluid dynamics (CFD):** CFD is used to model the flow of fluid around and within an object by solving mathematical equations governing the flow with the help of numerical methods. It is an inter-disciplinary domain and has applications in multiple domains. Complex flow phenomenon can be simulated with CFD. However, it requires massively parallel supercomputers to run the simulations efficiently and effectively [41, 105].
- **Graph computations:** Graphs are extensively used combinatorial tools in computing. They are used for representing sparse matrices, assist load balancing in computations [16], model molecular structures, traffic networks [16], and social media networks [8, 119], and distribution networks. It is also used in bioinformatics, business-analytics, and city planning. As graphs grow larger in size, we require powerful computational techniques for effective processing.
- **Computational and artificial intelligence (AI):** AI has become a fundamental technique for developing smarter algorithms and solutions in all scientific computations domains. Training deep learning and machine learning-based models require large computing power. For example, AI in healthcare networked systems [95] and educational systems [88] provides a better quality of service (QoS) and experience to the users. Deep learning has also been used to forecast traffic conditions for smart cities [14], and there are numerous other applications of AI, machine and deep learning. Probabilistic methods have also been used for computational intelligence, see, e.g., [71, 80] and the references therein. Solution of sparse linear equation system is an important part of such computational intelligence techniques. Solving sparse linear equation system mainly consists of sparse matrix vector multiplication which requires efficient utilization of parallel devices and this is discussed next.
- **Linear algebra and matrix computations:** A large number of scientific domains require linear algebra and matrix computations, such as dense or sparse matrix matrix multiplications (MMM), dense matrix vector products (MVPs), and sparse matrix vector products (SpMV) [3]. Application of HPC for dense linear algebra (MVP/MMM) can be seen in [17, 122, 123]. Work on efficiently utilizing parallel devices for SpMV can be seen in [11, 12, 69, 70, 79–82, 85, 86].
- **Big data:** Big data refers to "the emerging technologies that are designed to extract value from data having four vs characteristics; volume, variety, velocity and veracity" [87]. Big data technologies are being used in many application areas that require HPC to address big data challenges, see, e.g., [5, 83, 88, 95, 118, 119]. There are many ongoing efforts on the convergence of HPC and big data [36, 108, 124].

Table 18.2 A summary and comparison of benchmarking suites for HPC applications

Name	Developers	Benchmark type	Supports	Language
DEISA	EUS	Real apps	Cloud+MPI	C
HPC challenge benchmark	DARPA	Micro	MPI+OpenMP	C
Iometer	Intel	I/O	All network environments	C++
LINPACK	Dongarra et. al.	Kernel	MPI+OpenMP	Fortran
NAS parallel benchmark (NPB)	NASA	Kernel	MPI	C/C++
NPB multi-zone(NPB-MZ)	NASA	Kernel	MPI+OpenMP	C/C++
PARSEC	Princeton Univ.	Kernel	Multi-threaded SMA	C/C++
PerfKitBenchmarker	Google	Kernel	Cloud environment	Python
PMaC HPC	SciDac PERC	Micro	MPI+OpenMP	C++
Rodinia	Kevin Skadron	Real apps	CUDA/OpenMP	C/C++
STREAM	UOV	Kernel	MPI	C++/F90
VMmark	VMware	Virtual machine	Virtual machines & cloud	C/C++

– **Smart Cities, societies, and infrastructure:** Smart cities are driven by the rapid advancements in ICT technologies. These ICT developments have given rise to the integration and convergence of digital and physical systems such as computing, communications, big data, transport, healthcare, and city operations [6, 8, 10, 15, 16, 63, 83, 84, 89, 94, 95, 111, 118, 119, 124]. See, e.g., [88] for background on smart cities and societies.

Researchers have used applications from the discussed domains to study the performance of these applications in high-performance environments. Several works analyze the performance of applications from discussed domains using various known benchmarks. Table 18.1 lists some of the important applications that use HPC and Table 18.2 provides some of the major benchmarks used. Performance analysis tools have been used to analyze the applications to detect the bottlenecks in the code. We shall discuss these tools in later sections along with the review of earlier research.

18.3 Performance Analysis of HPC Applications: Literature Review

18.3.1 Performance Analysis Metrics (Theoretical)

Carrington et al. [27] analyze the metrics used for evaluating the performance of HPC applications. They mainly evaluate a simple synthetic metric, a linear combination of various single metric with weights. They also test a metrics derived by convolving an application transfer function with the system performance data obtained using any one single simple metric which is also known as predictive metrics. The authors evaluate the performance of ten Department of Defense high-performance computing modernization applications (HPCMP) [31]. Of the ten selected application, five of the applications are workload dependent and the other five are workload independent. Each of these ten applications was run on ten target systems with a distinct architecture. The simple benchmarks such as LINPACK [31], STREAM [78], and HPC challenge [76] have a weak correlation to performance and hence the authors additionally use synthetic benchmarks in combination with prediction and performance modeling framework [48]. A transfer function is applied to the test result by the prediction model that enables the representation of multiple categories using one single metric. In the simple metric scheme, the metric from a single benchmark is used, whereas in predictive benchmark they use a set of single benchmark metrics along with a real-time trace of the application. Simple metrics is modeled as follows:

$$T'(A, B) = \frac{L(A)}{L(A_o)} \cdot T(A_o, B) \tag{18.1}$$

where $T'(A, B)$ is the predicted clock time for application B on system A, $L(A)$ is the result for a specific single benchmark for system L, A_o denotes the base system benchmark, and $T(A, B)$ is the measured wall clock time for A on B. The errors reported are calculated as

$$\%\,Error = \frac{T'(A, B) - T(A_o, B)}{T(A, B)} \tag{18.2}$$

For predictive metrics, the authors use a tracer such as MetaSim tracer [28] for dynamic tracing of the base station and synthetic probes are used for measuring the rates for each operation on a target system. The MetaSim convolver [114] divides the execution operation count by operation rate to achieve execution type for current basic block per operation. After experimentation, the authors conclude that the correlation between the metric and real-time performance is higher than simple metrics.

18.3.2 HPC on the Clouds

Gupta et al. [47] provide an evaluation and comparison between the performance of HPC applications on the cloud and on traditional HPC systems such as super-computers and clusters. They also answer questions such as which HPC application is suitable for cloud, when is it suitable to choose cloud to run HPC application, and what application can be run on the cloud. The authors grade the performance of a number of selected applications on a number of the platform including supercomputer, a different type of clouds and clusters. The authors recognize various bottlenecks and the correlation between the characteristics and performance of the HPC application. The authors use three different benchmarks to analyze the performance of the HPC application. These are NAS parallel benchmark [97], a benchmark based on MPI [75], and a benchmark based on Charm++ [60]. The following systems were used to test the applications:

– Ranger supercomputer
– Taub (an HPC optimized cluster)
– Open Cirrus (physical nodes with commodity Interconnect)
– Private cloud
– Public cloud
– Amazon EC2-CC cloud

They ran the following HPC applications on the above machines:

– Jacobi2D
– NAMD [23]
– ChaNGa [55]
– Sweep3D [128]
– NQueens

The authors made three observations based on running the above HPC application on all the machines discussed above:

1. Some application scaled really well on all platform
   (a) Applications such as Jacobi-2D and NQueens scaled well on all the cores.
2. Scaling only till 32 cores on private cloud
   (a) NAMD and ChaNGa show this behavior. This is the effect of virtualization of the network.
3. Variable runtime for HPC applications on different execution in clouds
   (a) The variability was seen to increase when increasing the scaling.

The authors used various MPE, Jumpshot [131], and Projection tools to trace the HPC applications communication characteristics. On analyzing the communication, the authors have come to a conclusion that communication performance is a major bottleneck. They also conclude that virtualization decreases the performance of an HPC application. This is due to the presence of high latency and reduction in bandwidth. It was also observed that there are random idle times which the authors attribute to interference by other systems. The authors also reach a conclusion on the variability of the running time of HPC application. They say that it is due to the coupling of heterogeneous components in hardware and due to sharing of the virtual machines by external users.

Jackson et al. [53] discuss the performance analysis of HPC applications on the cloud. They compare conventional high-performance computing platforms such as supercomputers to Amazon EC2 cloud using HPC applications. The authors use the NERSC benchmarking framework [29, 32, 49, 75, 96, 101–103, 125] to evaluate the performance of HPC applications on Amazon EC2. In addition to NERSC, they also use integrated performance monitoring (IPM) framework. This framework will provide us with details on the time spent by the application on computation and on networking. They use four machines for this evaluation. The first of such machine is called Carver, which is a four hundred node cluster, which belongs to Lawrence Berkeley National Laboratory. It uses a quad-core Intel Nehalem processor @ 2.67 GHz. Each node has twenty-four GB of RAM. The second machine to be tested is Franklin which is a CrayXT4 supercomputer consisting of 9660 nodes. It has a single AMD Budapest processor @ 2.3GHz. The third system to be tested is Lawrencium which is a Linux cluster that has 198 nodes and the third system is the Amazon EC2 cloud. For testing purposes, they used four compute units of Amazon where one compute unit is equal to 1.2 GHz of Xeon 2007 processors. Normally all of the traditional HPC has a shared parallel file system that between the master nodes and the slave nodes. This is recreated in the cloud environment using virtual clusters [38, 40, 62]. A number of python scripts were used to configure the master node and the slave nodes. The master node would submit the jobs to the slave nodes using MPI and the file system will be shared between the nodes. The shared file system is implemented with the help of elastic block store [13] device which is attached to the virtual machine. The ext3 file system was used in this disk. Eight different HPC applications were evaluated by the authors from the benchmarking suite. These applications are (1) the community atmosphere model, (2) the general atomic and molecular electronic structure system, (3) GTC, (4) IMPACT-T, (5) MAESTRO, (6) MILC, (7) PARATEC, and (8) HPCC. Sustained system performance [68] for each of these applications was computed based on the NERSC benchmark as follows:

$$SSP = N \left(\prod_{i=1}^{M} P_i \right)^{(1/M)} \tag{18.3}$$

where P_i is the performance figure in gigaflops per second per core, N is the number of computational cores. Basically, SSP is the geometric mean of P_i over$M$ applications multiplied by N. It was observed that Lawrencium and Amazon EC2 were the worst performers in terms of computation. Moreover, the network latency is very poor in both of these systems. It was observed that EC2 was 20 times worst performer than the penultimate worst performer. The memory access in the EC2 platform is 10 times slower than the next worst performer Lawrencium. Totally the results indicate that the network has a very high impact on the performance of HPC applications on the cloud.

Roberto et al. [35] analyze the performance of HPC applications on the cloud. They analyze major performance bottlenecks in cloud platform using Amazon EC2 cluster [33] computing environment. Amazon provides two cluster computing instances named CC1 and CC2. CC1 consists of two quad-core processors whereas CC2 consists of two octa-core processors. They are made keeping in mind the requirement of HPC applications and high network using applications [25]. The authors evaluate 64 instances of CC1 and 32 instances of CC2, which sums up to a total of 512 cores. NAS parallel benchmark suite [19] and NPB multi-zone suite [56] are used for evaluation. Both of these cluster instances use Xen hypervisor for virtualization. The input–output is managed by paravirtual drivers that improve the performance as compared to normal clouds. OpenMPI [92] and MPICH2 [93] are used as the messaging middleware for codes using C/C++. Whereas, for Java-based code they use FastMPJ [120]. Initially, they performed a micro-benchmarking of data transfer from one point to another. They analyzed data transfer between both the inter-cluster and intra-cluster. Micro-benchmarking was conducted using Intel MPI benchmarks suite [126]. Then they analyzed the performance of the HPC kernels especially the effects on scalability due to paravirtualization. This was performed using NAS parallel benchmarks. Then they analyze the suitability of the amazon-based cloud networks for HPC application execution. The metrics they consider are millions of operation per second (MOPS). From the inter-VM communication analysis, it was found that OpenMPI and FastMPJ had lower start-up latency than MPICH2 but still as compared to an application running on barebones hardware these values are not enough. The results in the octa-core cloud were better than the performance of quad-core cloud. Still the major bottleneck is the delay in networking due to the para-virtualized access of the virtual machines of the cloud. In the Intra-VM data transfer, we only use the shared memory and does not use the network infrastructure and hence it was observed that it was better than Inter-VM performance. In this case, the latency was as low as 0.3 and 0.45 μs on both CC1 and CC2. In CC2, it is slightly higher due to the higher clock frequency of CC2. It was observed that cache hierarchy influences the performance of shared memory in both the cluster instances. HPC kernel analysis was performed using Fourier transform, integer sort, and conjugate gradient applications from the NPB kernels. It was observed that the scalability is higher if shared memory was used for data transfer but the moment the application was run on Inter-VM the performance dropped. The analysis reveals that the para-virtualized access of the network interface card by the cloud results in higher start-up latency which limits the scalability of the HPC applications that use intensive communication. The authors conclude that the major bottleneck in running HPC applications in cloud is the communication bottleneck of the cloud due to para-virtualized access of the network interface card by the cloud platform. It has also concluded that CC1 has better scalability than CC2 even though CC2 has higher computational power. If the performance to cost ratio is compared, then CC1 is better than using CC2. The scalability can be increased by running a single process per VM so that shared memory is used instead of network infrastructure. Multiple levels of parallelism have been shown to increase the scalability and performance such as multi-threading with message passing.

Waseem et al. [4] propose a framework for porting scientific applications to between heterogeneous clouds.

18.3.3 Performance Analysis Tools

Burtscher et al. [26] propose a tool for analyzing the performance of HPC applications. It consists of an advanced engine behind a highly usable GUI for bottleneck analysis. In an application, for each procedure, class, and loop it can analyze core, socket, and other bottlenecks. It then provides a brief evaluation and the steps required to remove that specific bottleneck including strategies to optimize the performance of the code. PerfExpert removes the need to have in-depth knowledge about computer architecture to optimize the HPC code. They also present a new metric for measuring the performance called as LCPI, which stands for local cycles per instruction. It is a combination of measurements from performance counters and architectural parameters. Since the local values for each loop are computed, procedure, and class they call it local CPI. For each procedure, loop, and class it calculates the CPI. It also returns the contribution of the following operations to the CPI: (1) memory access data, (2) memory access by instruction, (3) data TLB access, (4) instruction TLB access, (5) FP operations, and (6) branches in loops. It has fifteen performance counters to measure the overall LCPI and the LCPI associated with the six operations discussed above. It also calculates the upper bound of the latency caused due to the six operations discussed above. Some of the major performance counters provided by PerfExpert are L1 and L2 cache access by both data and instruction, total cycles and instruction in the HPC code, L2 cache miss by both data and instruction, instruction and data TLB miss, branch prediction and operations such as floating point addition, subtraction, and multiplication. These LCPI parameters are combined with various system parameters to find important bottlenecks. This can be used to restrict the conceivable causes of bottlenecks. For example, the contribution of a branch statement to the LCPI is given by

$$(BR\_NS * BR\_latency + BR\_MSP * BR\_misslat)/TOT\_INS \tag{18.4}$$

where $BRINS$ is the total branch instruction, $BRMSP$ indicates the branches missed, and $TOTINS$ is the total instructions in the HPC code. $BRmisslat$ is the miss prediction latency by the CPU and $BRlatency$ is the latency of branch. PerfExpert runs HPCToolkit [30] under its hood. The HPC code is run by the PerfExpert multiple times over the HPCToolkit. It saves the data from various performance counters to a file. The data is then accessed by PerfExpert to find out the bottlenecks in the code. On the basis of the analysis, it will then provide optimization suggestions for the code.

Knupfer et al. [65] discuss a toolset for evaluating the performance of HPC applications called the Vampir toolset. The Vampir toolset mainly consists of three components. They are (1) VampireTrace, (2) a set of visualization tools named Vampir, and (3) Vampir server. VampireTrace is an application tracer for HPC applications. Tracing an HPC application requires Instrumentation which is the modification of the application being traced to detect various events occurring. VampireTrace provides four different kinds of instrumentation, namely compiler instrumentation, source to source instrumentation, library instrumentation, and manual instrumentation. These modifications are performed at the build time of application. Special flags are provided for the compiler to generate calls for instrumentation. It supports a number of compilers such as GCC, Intel compiler suite, IBM compiler suite, Sun compilers, and NEC compilers. The major disadvantage associated with this technique is the large size of the trace file that is generated. Source to source instrumentation is used by Vampir to instrument the MPI application. Library instrumentation replaces the existing libraries of the HPC application with the libraries required for instrumentation. The major advantage of this technique is that without compiling and link frequently. The disadvantage of this technique is that it requires that all the APIs be replaced by the new libraries. Manual instrumentation is used to get more powerful control on what events need to be traced and which events not. The Vampir toolkit can record the following events.

1. Hardware performance counters: Can find out various performance parameters such as statistics on cache performance, statistics on branch predictions, and statistics on floating point operations.
2. Memory usage of the application: It can trace the memory usage of the HPC application dynamically [59]. It replaces the normal memory functions such as malloc, realloc, and free with special wrappers from the GCC compiler library.
3. Input and output activity tracking: Each and every input and output activity performed by the HPC application can be traced by the VampireTrace by intercepting the I/O calls made by the application.
4. Performance counters defined by the users: The users can define a number of performance counters such as loop counts, results, or other scalar quantities.

Tracing can cause an overhead in the system which results in the performance degradation of the HPC application and might alter the original characteristics of the application. The overhead is introduced mainly at four places in the system, namely the initialization phase, during event handling, during storage of tracing information to disk, and during finalization. Vampir server is a client–server framework that uses distributed systems for the evaluation of HPC application evaluation. We use a parallel production environment as the server and the clients can be desktop computers connected remotely for envisaging the performance graphically. The graphical client enhances the understandability of the system by showing graphical results of the evaluation. It provides various graphical timelines of the application execution. The timelines consist of a global timeline that shows the timeline of all process and threads, a summary timeline that provides the process that is involved with various activity over a period of time. The other two timelines provided are the counter timeline and the process timeline. The counter timeline shows the state of counters with respect to time. Other than timeline display it also provides various statistical displays such as the summary chart, activity chart, message statistics, and input–output statistics. The authors conclude that it is a robust tool that provides a good analysis of HPC applications.

Wolf et al. [127] discuss SCALSCA toolset that has been specially designed to analyze the performance of high-performance parallel applications. It provides a complete runtime summaries and provides insight into the application behavior through various techniques such as tracing and measuring various parameters. It has been designed to be used with large end HPC systems such as IBM blue gene. SCALASCA can identify uneven workloads and evaluate it. It can also detect wait states that occur due to the above-mentioned phenomenon. SCALASCA can be used for applications using OpenMP, MPI, and other hybrid applications written in C/C++ and FORTRAN. It can be run on a wide range of platforms. SALASCA is accessed using the command scales with various flags. Initially, before we start the analysis the code to be analyzed has to be instrumented. By instrumentation, it means that the code has to be modified to record and evaluate various events related to the performance of the system. This process is automated in some platforms whereas in some other platforms it has to be done manually. After this, the instrumented code is run. After the run, there is the option to get the trace of individual runtime events from which a GUI-based timeline representation can be created. Alternatively, a profile that aggregates the performance of various events and a summary report from it can be created. The second option provides an overall summary whereas the trace provides a detailed information. The trace produces a trace file that is used loaded into the memory for evaluation. During the evaluation, it detects various events of significance. It performs a pattern analysis and provides a report about this analysis. Both the pattern report and summary produced contain various performance metrics that is useful for the user. The result of tracing can also be analyzed by other third party tools. Some of the disadvantages of this are that the trace analysis for OpenMP is done serially and the summary produced has only metrics for OpenMP. The authors are improving it by incorporating more functionalities. They are trying to come up with a workaround for the restrictions imposed by the CUBE file format. The bottlenecks or the performance degradation might occur at a later time than the time at which the actual event took place. They are also trying to find a workaround for this issue.

18.3.4 Performance Analysis of Exascale Systems

Abraham et al. [1] discuss the possible performance evaluation of HPC applications on exascale systems. Of all the HPC applications present, only a few HPC applications are capable of exploiting even the petaflop systems [18]. The traditional measurement-based evaluation on exascale system cannot be done because an exascale system does not yet exist. The authors identify the challenges and provide solutions for various problems that are faced when running an HPC application on the exascale system including optimization of the code, formal modeling, and static and runtime analysis. They also propose a conceptual framework for HPC application performance analysis to run it on an exascale system. They try to adapt the HPC application code to achieve good utilization of resources abstractly.

For this, they get the resource usage footprint of various modules at various granularity. An abstract behavioral specification language to describe both the task and deployment model can be used [58]. Also, the language should have big parallel operators over big resource footprint [106]. A resource footprint can be specified using the standard model in model driven development when the code is developed from scratch. The authors also apply formal methods to the code, unlike others who monitor the code and report various statistics regarding the performance. They also provide the challenges faced at runtime analysis such as the absence of a good tool to measure energy metrics with accuracy and given time [9, 24]. Current runtime analysis includes runtime measurement like profiling and tracing data [25]. This has many disadvantages such as manual changing of code and ineffective improvement in performance. They also provide a conceptual framework for designing HPC applications for exascale. The major components in this framework can be summarized as follows:

– Use of domain specific exascale language (DSEL): This is useful for expressing the non-functional aspects of execution.
– Scalable model-based analyzer (SMA): They are responsible for monitoring and evaluating resource consumption.
– Exascale runtime data collector (ERDC): It performs functionalities such as data mining, filtering, and data analysis.
– Autonomous feedback loop (AFL): AFL gets the feedback from the runtime that is fed back into the model to tune the model.

Energy-Efficient Systems (Embedded Systems Nodes)

One of the major roadblocks for the adoption of exascale systems for HPC computations is the energy consumption of exascale systems. A supercomputer is not supposed to exceed 20 MW. We would require an efficiency of almost fifty gigaflops (GFLOPS) per watt to build an exascale system below 20 MW. To break through this barrier, we need to increase the efficiency of the current systems by a factor of twenty-five, whereas in the embedded systems the energy is of great importance. Therefore, one way to do this is to use the components used in the embedded systems as analyzed by Stanisic et al. [115]. Stanisic et al. [115] developed HPC clusters with the help of low-power embedded components and they evaluate various existing HPC applications on this embedded platform. They study the scalability of the existing HPC applications to the low-power embedded system-based HPC clusters. They used eleven HPC applications to run on the low-power HPC clusters. SPECFEM3D and BigDFT are the two major HPC applications used by them. The authors have developed a board called snowboard which is a fully embedded computer by itself. It uses a powerful ARM dual cortex A9 running at 1GHz. It has 8 GB of e-MMC memory. It has a full HD supported HDMI port along with Mali 400 GPU. The power consumption of this board is just 2.5 W. This board is compared with an Intel Xeon running at quad-core CPU at 2.6 GHz and has a power consumption of 95 W. Three benchmarks are used to compare these two systems: (1)LINPACK, (2) CoreMark, and (3) StockFish. It was found that it takes the same amount of energy on both platforms to run LINPACK but benchmarking SPECFEM3D using CoreMark requires only less than 5 times of energy required by Xeon. It was found out that the applications scale well for more than 90% on the embedded low-power platform. The scalability is almost ideal when using SPECFEM3D whereas BigDFT showed a lesser scalability. They further investigate the performance of HPC application by analyzing the memory. For this, they run a heavy memory rigorous kernel as shown in [121]. In this technique they measure the time it takes to access the elements of a fixed size array inside a loop using fixed steps. The memory bandwidth is calculated after running the above kernel. It is the ratio of the total number of access that was required to access to the time it took to execute the kernel. The memory structure of ARM is dissimilar to the Intel architecture. Hence, different behaviors were observed. A large number of cache miss was observed because for an array of size 32 KB the page allocation was not consecutive. Moreover, the memory had to be frequently cleaned. The use of real-time schedulers also resulted in the degradation of the HPC applications performance which is caused by wrong decisions taken by the OS scheduler. A variety of code optimization was done to HPC applications including changing element size, unrolling of loop, etc. Increasing the size of variables as well as loop rolling had optimistic results. There was quite an improvement in the performance due to the doubling of the bandwidth. The authors conclude that more HPC code optimization has to be performed to run HPC applications on low-power HPC platforms.

18.4 Discussion and Analysis

In this section we shall provide a qualitative analysis and discuss the performance analysis tools. Table 18.3 summarizes some of the major performance analysis tools that have been reviewed in this paper.

Most of the researchers have run the HPC applications on the cloud to study the feasibility of HPC applications on the cloud. References [21, 35, 47, 53, 120] ran various HPC applications on the cloud. One of the major findings of all these studies was that cloud as a platform does not scale well for HPC applications. One of the major reasons for this non-scalability after a certain extent was due to the network communication between the virtual machines. It scaled really well if the application was run using a single VM. The reason for this is due to the virtualization of the network interface card leads to degradation in the speed of the network. Shared memory can be used to reduce the network communication to increase the performance of the cloud-based application. The performance to cost ratio of the clouds is also that enticing even though some packages offered do provide the good cost to performance ratio as compared to other HPC platforms such as grids, clusters, and supercomputers. From the discussion of various researchers, we

Table 18.3 A summary and comparison of various performance analysis tools for HPC applications

Features	Vampir Suite	PerfExpert	Scalasca	Periscope	Kojak	HPCToolkit
Auto-instrumentation	Yes	Yes	Yes	Yes	Yes	Yes
Bottlenecks	Yes	Yes	Yes	Yes	Yes	Yes
Data visualization	Yes	No	Yes	No	No	Yes
GUI	Yes	No	Yes	Yes	No	Yes
Language	C++	C++/C	C++/C	C++/C	C/F90	C++
Open source	No	Yes	No	No	No	Yes
Profiling	Yes	Yes	Yes	No	No	Yes
Restructure	No	Yes	Yes	No	No	No
Tracing	Yes	Yes	Yes	Yes	Yes	Yes
Timeline	Yes	No	Yes	No	No	No

conclude that the scalability of HPC applications on the cloud depends on a number of factors such as network communication especially between the virtual and real components. Combining message passing with multi-threading also increases the scalability of the HPC applications on the cloud platform. HPC applications with less communication and less interference sensitivity are suitable for the cloud. The HPC applications have to be modified to make it cloud-aware.

Moreover, we need to consider the future wherein we have to efficiently use the exascale systems. A major roadblock for this is the power consumption and hence we found that HPC applications can also be executed on low cost embedded hardware [115]. The MONT-BLANC project deals with analyzing the performance of HPC applications on low-power embedded systems. It was observed that it performs really well on embedded platform but had issues with real-time scheduling and physical page application. A framework for exascale systems has been developed by Ábrahám et al. [1]. This will be useful for modifying the HPC application accordingly and in increasing their performance. Graphical processing units (GPUs) are also among the highest power-efficient devices. A large number of supercomputers in the Top500 list[1] use GPUs. The Green500[2] lists the top 500 supercomputers in the world by energy efficiency. Supercomputers with the highest Flops/Watts ratio are ranked first in this list. GPUs are the accelerating unit for seven supercomputers in the top ten in Green500.[3]

A number of performance analysis tools exist that help in analyzing the performance of the HPC applications. A brief summary and comparison are provided in Table 18.3. All of the tools provide profiling and tracing features. Profiling is the feature where it would provide various global performance metrics. Tracing consists of following the code in runtime and tracing the code. These actually require the

[1] https://www.top500.org/.

[2] https://www.top500.org/green500/.

[3] https://www.top500.org/green500/lists/2018/06/.

modification of the code. This process of modification is called instrumentation. All of them also provide instrumentation support at various levels. They help in detecting the bottlenecks, but all of them do not give suggestion and where to change the code and what code needs to be changed. An example of this is PerfExpert that provides the complete details on code creating the bottleneck and the restructuring required. But PerfExpert does not provide GUI support whereas others provide a complete statistical information using charts, graphs, and timelines in which you can zoom through various scales of timescale. The Vampir toolkit provides separate timelines for the summary, counter, and various process. It also provides statistical charts for various activities, message statistics, and I/O activities. SCALSCA is also very similar to PerfExpert, but PerfExpert is open source whereas SCALASCA is proprietary software.

18.5 Future Research and Issues

There are a lot of issues associated with analyzing the performance of HPC application. We list some of the major issues and the direction of the required future work.

– Most of the tools available are for the Linux platform. Hence, we need to port various performance analysis tools to Windows platform
– There are a lot of redundancies when the application performs a trace. Eliminating redundancy in trace has to be done.
– Pattern detection in large datasets using complete call graphs [37].
– Applications that use single instruction multiple data (SIMD) consists of redundancies in execution. It is required to replace these redundant behaviors with a single instance which will result in higher performance and lesser memory.
– More case studies with more applications on more platforms are required.
– Expand current capabilities of the performance analysis tools by using counters that are not based on performance.
– The performance analysis tools should automatically not only suggest solutions for the bottlenecks but also automatically implement these into the code for all kinds of bottlenecks.
– The performance analysis tools should also provide a higher level of input–output optimization.
– The variability and the reduction of speed in clouds were attributed to the network communication in the cloud. This issue of network delay has to be mitigated.
– Further improvement in the scalability of the performance analysis tools is required.
– The OpenMP analysis in SCALASCA performance analysis tool has to be parallelized.
– There are no existing tools to analyze the input–output in MPI, currently this is performed by serial tracer in SCALASCA.

– The current file formats that are used for the performance analysis provide a lot of restrictions such as storing and processing of metrics that cannot be aggregated. An efficient file system has to be developed.
– The bottlenecks that appear in the application might appear at a later time than when the event that caused it occurs. The performance analysis tools should be able to figure out the events that caused the issue in such occurrence.
– HPC applications have to be modified so that it performs efficiently in exascale and petascale.

18.6 Conclusions

In this chapter, we reviewed the works on analyzing HPC applications and discussed several performance analysis tools for HPC applications. The researchers observed the performance of the HPC applications on various platforms and identified various bottlenecks and issues related to the performance of the HPC application on parallel platforms. Some researchers have discussed the usage of performance tools for binding and replacing the bottleneck in the HPC applications. We also studied and qualitatively compared various tools for performance analysis of HPC applications. We discussed the performance of various HPC applications on a number of HPC platforms. Moreover, we discussed various HPC benchmarking suites and various HPC applications being used for the performance analysis and thereafter, we also discussed the future research directions and issues.

Acknowledgements The authors acknowledge with thanks the technical and financial support from the Deanship of Scientific Research (DSR) at the King Abdulaziz University (KAU), Jeddah, Saudi Arabia, under the grant number G-651-611-38. The work carried out in this paper is supported by the High Performance Computing Center at the King Abdulaziz University, Jeddah.

References

1. Ábrahám, E., Bekas, C., Brandic, I., Genaim, S., Johnsen, E.B., Kondov, I., Pllana, S., Streit, A.: Preparing HPC applications for exascale: Challenges and recommendations (2015). CoRR abs/1503.06974. http://arxiv.org/abs/1503.06974
2. Abraham, M.J., Murtola, T., Schulz, R., Páll, S., Smith, J.C., Hess, B., Lindahl, E.: Gromacs: high performance molecular simulations through multi-level parallelism from laptops to supercomputers. SoftwareX **1-2**, 19–25 (2015). http://www.sciencedirect.com/science/article/pii/S2352711015000059
3. Agullo, E., Demmel, J., Dongarra, J., Hadri, B., Kurzak, J., Langou, J., Ltaief, H., Luszczek, P., Tomov, S.: Numerical linear algebra on emerging architectures: the plasma and magma projects. J. Phys. Conf. Ser. **180**, 012037 (2009)
4. Ahmed, W., Khan, M., Khan, A.A., Mehmood, R., Algarni, A., Albeshri, A., Katib, I.: A framework for faster porting of scientific applications between heterogeneous clouds. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. pp. 27–43. Springer International Publishing, Cham (2018)

5. Alam, F., Mehmood, R., Katib, I., Albogami, N.N., Albeshri, A.: Data fusion and IoT for smart ubiquitous environments: a survey. IEEE Access **5**, 9533–9554 (2017)
6. Alam, F., Mehmood, R., Katib, I.: D2TFRS: an object recognition method for autonomous vehicles based on RGB and spatial values of pixels. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. pp. 155–168. Springer International Publishing, Cham (2018)
7. Alamoudi, E., Mehmood, R., Albeshri, A., Gojobori, T.: Dna profiling methods and tools: a review. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications, pp. 216–231. Springer International Publishing, Cham (2018)
8. Alomari, E., Mehmood, R.: Analysis of tweets in Arabic language for detection of road traffic conditions. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. pp. 98–110. Springer International Publishing, Cham (2018)
9. Alonso, P., Badia, R.M., Labarta, J., Barreda, M., Dolz, M.F., Mayo, R., Quintana-Orti, E.S., Reyes, R.: Tools for power-energy modelling and analysis of parallel scientific applications. In: 2012 41st International Conference on Parallel Processing (ICPP), pp. 420–429. IEEE, New York (2012)
10. Alotaibi, S., Mehmood, R.: Big data enabled healthcare supply chain management: opportunities and challenges. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications, pp. 207–215. Springer International Publishing, Cham (2018)
11. Alyahya, H., Mehmood, R., Katib, I.: Parallel sparse matrix vector multiplication on Intel MIC: performance analysis. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications, pp. 306–322. Springer International Publishing, Cham (2018)
12. Alzahrani, S., Ikbal, M.R., Mehmood, R., Fayez, M., Katib, I.: Performance evaluation of Jacobi iterative solution for sparse linear equation system on multicore and manycore architectures. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications, pp. 296–305. Springer International Publishing, Cham (2018)
13. Amazon: AWS | Amazon Elastic Block Store (EBS) - Incremental Backup & Persistent Storage. http://aws.amazon.com/ebs/
14. Aqib, M., Mehmood, R., Albeshri, A., Alzahrani, A.: Disaster management in smart cities by forecasting traffic plan using deep learning and GPUs. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications, pp. 139–154. Springer International Publishing, Cham (2018)
15. Arfat, Y., Aqib, M., Mehmood, R., Albeshri, A., Katib, I., Albogami, N., Alzahrani, A.: Enabling smarter societies through mobile big data fogs and clouds. Proc. Comput. Sci. **109**, 1128–1133 (2017). http://www.sciencedirect.com/science/article/pii/S1877050917311213. 8th International Conference on Ambient Systems, Networks and Technologies, ANT-2017 and the 7th International Conference on Sustainable Energy Information Technology, SEIT 2017, 16-19 May 2017, Madeira
16. Arfat, Y., Mehmood, R., Albeshri, A.: Parallel shortest path graph computations of United States road network data on apache spark. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications, pp. 323–336. Springer International Publishing, Cham (2018)
17. Azad, A., Ballard, G., Buluç, A., Demmel, J., Grigori, L., Schwartz, O., Toledo, S., Williams, S.: Exploiting multiple levels of parallelism in sparse matrix-matrix multiplication. SIAM J. Sci. Comput. **38**(6), C624–C651 (2016). https://doi.org/10.1137/15M104253X
18. Bader, D.A.: Petascale Computing: Algorithms and Applications. CRC Press, Boca Raton (2007)
19. Bailey, D.H., Barszcz, E., Barton, J.T., Browning, D.S., Carter, R.L., Dagum, L., Fatoohi, R.A., Frederickson, P.O., Lasinski, T.A., Schreiber, R.S., et al.: The NAS parallel benchmarks. Int. J. High Perform. Comput. Appl. **5**(3), 63–73 (1991)

20. Bailey, J.A., Bazavov, A., Bernard, C., Bouchard, C.M., DeTar, C., Du, D., El-Khadra, A.X., Foley, J., Freeland, E.D., Gámiz, E., Gottlieb, S., Heller, U.M., Kim, J., Kronfeld, A.S., Laiho, J., Levkova, L., Mackenzie, P.B., Meurice, Y., Neil, E.T., Oktay, M.B., Qiu, S.W., Simone, J.N., Sugar, R., Toussaint, D., Van de Water, R.S., Zhou, R.: Refining new-physics searches in $b \to d\tau\nu$ with lattice QCD. Phys. Rev. Lett. **109**, 071802 (2012). https://link.aps.org/doi/10.1103/PhysRevLett.109.071802
21. Benedict, S.: Performance issues and performance analysis tools for HPC cloud applications: a survey. Computing **95**(2), 89–108 (2013)
22. Berriman, G.B., Juve, G., Deelman, E., Regelson, M., Plavchan, P.: The application of cloud computing to astronomy: A study of cost and performance. In: 2010 Sixth IEEE International Conference on e-Science Workshops, December, pp. 1–7 (2010)
23. Bhatele, A., Kumar, S., Mei, C., Phillips, J.C., Zheng, G., Kale, L.V.: Overcoming scaling challenges in biomolecular simulations across multiple platforms. In: IEEE International Symposium on Parallel and Distributed Processing, 2008 (IPDPS 2008), pp. 1–12. IEEE, New York (2008)
24. Bohra, A.E.H., Chaudhary, V.: Vmeter: power modelling for virtualized clouds. In: 2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and PhD Forum (IPDPSW), pp. 1–8. IEEE, New York (2010)
25. BPG: Best Practice Guides. http://www.prace-ri.eu/best-practice-guides/
26. Burtscher, M., Kim, B.D., Diamond, J., McCalpin, J., Koesterke, L., Browne, J.: PerfExpert: an easy-to-use performance diagnosis tool for HPC applications. In: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–11. IEEE Computer Society, Washington (2010)
27. Carrington, L.C., Laurenzano, M., Snavely, A., Campbell Jr., R.L., Davis, L.P.: How well can simple metrics represent the performance of HPC applications? In: Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference, pp. 48–48. IEEE, New York (2005)
28. Carrington, L., Snavely, A., Wolter, N.: A performance prediction framework for scientific applications. Fut. Gener. Comput. Syst. **22**(3), 336–346 (2006)
29. Carter, J., Oliker, L., Shalf, J.: Performance evaluation of scientific applications on modern parallel vector systems. In: High Performance Computing for Computational Science-VECPAR 2006, pp. 490–503. Springer, New York (2007)
30. Djoudi, L., Barthou, D., Carribault, P., Lemuet, C., Acquaviva, J.T., Jalby, W.: Exploring application performance: a new tool for a static/dynamic approach. In: Proceedings of the 6th LACSI Symposium (2005)
31. Dongarra, J.L.A.P.: The LINPACK benchmark: past, present and future. Concurr. Comput. Pract. and Exp. **15**, 1–18 (2003)
32. Dunigan Jr, T.H., Vetter, J.S., White III, J.B., Worley, P.H.: Performance evaluation of the Cray x1 distributed shared-memory architecture. Micro, IEEE **25**(1), 30–40 (2005)
33. ECC2. Elastic Compute Cloud (EC2) Cloud Server & Hosting – AWS. //aws.amazon.com/ec2/
34. Eleliemy, A., Fayez, M., Mehmood, R., Katib, I., Aljohani, N.: Loadbalancing on parallel heterogeneous architectures: Spin-image algorithm on CPU and MIC. In: 9th EUROSIM Congress on Modelling and Simulation. EUROSIM (2016). http://edoc.unibas.ch/53117/
35. ExpóSito, R.R., Taboada, G.L., Ramos, S., Touriño, J., Doallo, R.: Performance analysis of HPC applications in the cloud. Fut. Gener. Comput. Syst. **29**(1), 218–229 (2013)
36. Farber, R.: The convergence of big data and extreme-scale HPC (2018). https://www.hpcwire.com/2018/08/31/the-convergence-of-big-data-and-extreme-scale-hpc/
37. Ferreira, G., Kästner, C., Pfeffer, J., Apel, S.: Characterizing complexity of highly-configurable systems with variational call graphs: analyzing configuration options interactions complexity in function calls. In: Proceedings of the 2015 Symposium and Bootcamp on the Science of Security. p. 17. ACM, New York (2015)
38. Foster, I., Freeman, T., Keahy, K., Scheftner, D., Sotomayer, B., Zhang, X.: Virtual clusters for grid communities. In: Sixth IEEE International Symposium on Cluster Computing and the Grid, 2006 (CCGRID 06), vol. 1, pp. 513–520. IEEE, New York (2006)

39. Freche, J., Frings, W., Sutmann, G.: High-throughput parallel-I/O using SIONlib for mesoscopic particle dynamics simulations on massively parallel computers. In: Parallel Computing: From Multicores and GPU's to Petascale Advances in Parallel Computing, vol. 19, pp. 371–378. IOS Press, Amsterdam (2010)
40. Freeman, T., Keahey, K., Sotomayor, B., Zhang, X., Foster, I., Scheftner, D.: Virtual clusters for grid communities. Citeseer (2006)
41. Gel, A., Hu, J., Ould-Ahmed-Vall, E., Kalinkin, A.A.: Modernization and optimization of a legacy open-source CFD code for high-performance computing architectures. Int. J. Comput. Fluid Dynam. **31**(2), 122–133 (2017). https://doi.org/10.1080/10618562.2017.1285398
42. Genovese, L., Videau, B., Ospici, M., Deutsch, T., Goedecker, S., Méhaut, J.F.: Daubechies wavelets for high performance electronic structure calculations: The BigDFT project. Comptes Rendus Mécanique **339**(2), 149–164 (2011). http://www.sciencedirect.com/science/article/pii/S1631072110002135. High Performance Computing
43. Giannozzi, P., Baroni, S., Bonini, N., Calandra, M., Car, R., Cavazzoni, C., Ceresoli, D., Chiarotti, G.L., Cococcioni, M., Dabo, I., et al.: Quantum espresso: a modular and open-source software project for quantum simulations of materials. J. Phys. Condens. matter **21**(39), 395502 (2009)
44. Gibbon, P.: Pepc: pretty efficient parallel coulomb-solver. Sonstiger Interner Bericht ZAM-IB-2003-05, ZAM, Jülich, Forschungszentrum (2003)
45. Gordon, M.S., Schmidt, M.W.: Advances in electronic structure theory: GAMESS a decade later. In: Dykstra, C.E., Frenking, G., Kim, K.S., Scuseria, G.E. (eds.) Theory and Applications of Computational Chemistry, chapter 41, pp. 1167–1189. Elsevier, Amsterdam (2005). http://www.sciencedirect.com/science/article/pii/B9780444517197500846
46. Gudiksen, B.V., Carlsson, M., Hansteen, V.H., Hayek, W., Leenaarts, J., Martínez-Sykora, J.: The stellar atmosphere simulation code Bifrost - code description and validation. Astron. Astrophys. **531**, A154 (2011). https://doi.org/10.1051/0004-6361/201116520
47. Gupta, A., Faraboschi, P., Gioachin, F., Kale, L., Kaufmann, R., Lee, B.S., March, V., Milojicic, D., Suen, C.: Evaluating and improving the performance and scheduling of HPC applications in cloud. IEEE Trans. Cloud Comput. **4**(99), 1–1 (2014)
48. Gustafson, J.L., Todi, R.: Conventional benchmarks as a sample of the performance spectrum. In: Proceedings of the Thirty-First Hawaii International Conference on System Sciences, 1998, vol. 7, pp. 514–523. IEEE, New York (1998)
49. Gygi, F., Yates, R.K., Lorenz, J., Draeger, E.W., Franchetti, F., Ueberhuber, C.W., Supinski, B.R.D., Kral, S., Gunnels, J.A., Sexton, J.C.: Large-scale first-principles molecular dynamics simulations on the Bluegene/l platform using the Qbox code. In: Proceedings of the 2005 ACM/IEEE conference on Supercomputing, p. 24. IEEE Computer Society, Washington (2005)
50. Heck, D., Pierog, T., Knapp, J.: CORSIKA: An Air Shower Simulation Program. Astrophysics Source Code Library (2012)
51. Hwu, W.M., Chang, L.W., Kim, H.S., Dakkak, A., El Hajj, I.: Transitioning HPC software to exascale heterogeneous computing. In: Computational Electromagnetics International Workshop (CEM), July 2015, pp. 1–2 (2015)
52. Irbäck, A., Mohanty, S.: Profasi: A Monte Carlo simulation package for protein folding and aggregation. J. Comput. Chem. **27**(13), 1548–1555. https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.20452
53. Jackson, K.R., Ramakrishnan, L., Muriki, K., Canon, S., Cholia, S., Shalf, J., Wasserman, H.J., Wright, N.J.: Performance analysis of high performance computing applications on the amazon web services cloud. In: 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), pp. 159–168. IEEE, New York (2010)
54. Jacobsen, N.G., Fuhrman, D.R., Fredsøe, J.: A wave generation toolbox for the open-source CFD library: Openfoam®. Int. J. Numer. Methods Fluids **70**(9), 1073–1088. https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.2726
55. Jetley, P., Gioachin, F., Mendes, C., Kale, L.V., Quinn, T.: Massively parallel cosmological simulations with ChaNGa. In: International Symposium on Parallel and Distributed Processing, 2008 (IPDPS 2008), pp. 1–12. IEEE, New York (2008)

56. Jin, H., Van der Wijngaart, R.F.: Performance characteristics of the multi-zone NAS parallel benchmarks. In: Proceedings of the 18th International Parallel and Distributed Processing Symposium, 2004, p. 6. IEEE, New York (2004)
57. Jöckel, P., Sander, R., Kerkweg, A., Tost, H., Lelieveld, J.: Technical note: the modular earth submodel system (MESSy) - a new approach towards earth system modeling. Atmos. Chem. Phys. **5**(2), 433–444 (2005). https://www.atmos-chem-phys.net/5/433/2005/
58. Johnsen, E.B., Hähnle, R., Schäfer, J., Schlatte, R., Steffen, M.: ABS: a core language for abstract behavioral specification. In: Formal Methods for Components and Objects, pp. 142–164. Springer, New York (2012)
59. Jurenz, M., Brendel, R., Knüpfer, A., Müller, M., Nagel, W.E.: Memory allocation tracing with VampireTrace. In: Computational Science–ICCS 2007, pp. 839–846. Springer, New York (2007)
60. Kale, L.V., Krishnan, S.: CHARM++: A Portable Concurrent Object Oriented System Based on C++, vol. 28. ACM, New York (1993)
61. Kay, J.E., Deser, C., Phillips, A., Mai, A., Hannay, C., Strand, G., Arblaster, J.M., Bates, S.C., Danabasoglu, G., Edwards, J., Holland, M., Kushner, P., Lamarque, J.F., Lawrence, D., Lindsay, K., Middleton, A., Munoz, E., Neale, R., Oleson, K., Polvani, L., Vertenstein, M.: The community earth system model (CESM) large ensemble project: a community resource for studying climate change in the presence of internal climate variability. Bull. Am. Meteorol. Soc. **96**(8), 1333–1349 (2015). https://doi.org/10.1175/BAMS-D-13-00255.1
62. Keahey, K., Figueiredo, R., Fortes, J., Freeman, T., Tsugawa, M.: Science clouds: early experiences in cloud computing for scientific applications. Cloud Comput. Appl. **2008**, 825–830 (2008)
63. Khanum, A., Alvi, A., Mehmood, R.: Towards a semantically enriched computational intelligence (SECI) framework for smart farming. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. pp. 247–257. Springer International Publishing, Cham (2018)
64. Kirk, B.S., Peterson, J.W., Stogner, R.H., Carey, G.F.: libMesh: a C++ library for parallel adaptive mesh refinement/coarsening simulations. Eng. Comput. **22**(3), 237–254 (2006). https://doi.org/10.1007/s00366-006-0049-3
65. Kпpfer, A., Brunst, H., Doleschal, J., Jurenz, M., Lieber, M., Mickler, H., Müller, M.S., Nagel, W.E.: The Vampir performance analysis tool-set. In: Tools for High Performance Computing, pp. 139–155. Springer, New York (2008)
66. Kodiyalam, S., Yang, R., Gu, L., Tho, C.H.: Multidisciplinary design optimization of a vehicle system in a scalable, high performance computing environment. Struct. Multidiscip. Optim. **26**(3), 256–263 (2004). https://doi.org/10.1007/s00158-003-0343-2
67. Komatitsch, D., Tromp, J.: Introduction to the spectral element method for three-dimensional seismic wave propagation. Geophys. J. Int. **139**(3), 806–822 (1999). https://onlinelibrary.wiley.com/doi/abs/10.1046/j.1365-246x.1999.00967.x
68. Kramer, W., Shalf, J., Strohmaier, E.: The NERSC Sustained System Performance (SSP) Metric. Lawrence Berkeley National Laboratory (2005)
69. Kwiatkowska, M., Mehmood, R.: Out-of-core solution of large linear systems of equations arising from stochastic modelling. In: Hermanns, H., Segala, R. (eds.) Process Algebra and Probabilistic Methods: Performance Modeling and Verification, pp. 135–151. Springer, Berlin/Heidelberg (2002)
70. Kwiatkowska, M., Mehmood, R., Norman, G., Parker, D.: A symbolic out-of-core solution method for Markov models. Electron. Notes Theor. Comput. Sci. **68**(4), 589–604 (2002). http://www.sciencedirect.com/science/article/pii/S1571066105803949
71. Kwiatkowska, M., Parker, D., Zhang, Y., Mehmood, R.: Dual-processor parallelisation of symbolic probabilistic model checking. In: Proceedings of the IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, MASCOTS '04, pp. 123–130. IEEE Computer Society, Washington (2004). http://dl.acm.org/citation.cfm?id=1032659.1034195

72. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) Proceedings of the 23rd International Conference on Computer Aided Verification (CAV'11). Lecture Notes in Computer Science, vol. 6806, pp. 585–591. Springer, New York (2011)
73. Letherwood, M.D., Gunter, D.D.: Ground vehicle modeling and simulation of military vehicles using high performance computing. Parallel Comput. **27**(1), 109–140 (2001). http://www.sciencedirect.com/science/article/pii/S0167819100000910. New Trends in High Performance Computing
74. Lingerfelt, E., Endeve, E., Hui, Y., Smith, C., Somnath, S., Grodowitz, N., Borreguero, J., Bao, F., Niedziela, J., Bansal, D., Delaire, O., Archibald, R., Belianinov, A., Shankar, M., Jesse, S.: BEAM: an HPC pipeline for nanoscale materials analysis and neutron data modeling. In: APS March Meeting Abstracts, p. A7.002 (2017)
75. Lusk, E., Huss, S., Saphir, B., Snir, M.: MPI: a message-passing interface standard (2009)
76. Luszczek, P.R., Bailey, D.H., Dongarra, J.J., Kepner, J., Lucas, R.F., Rabenseifner, R., Takahashi, D.: The HPC challenge (HPCC) benchmark suite. In: Proceedings of the 2006 ACM/IEEE conference on Supercomputing, p. 213. Citeseer (2006)
77. Mantripragada, K., Binotto, A., Tizzei, L., Netto, M.: A feasibility study of using HPC cloud environment for seismic exploration. In: 77th EAGE Conference and Exhibition 2015 (2015)
78. McCalpin, J.D.: Memory bandwidth and machine balance in current high performance computers (1995)
79. Mehmood, R.: A survey of out-of-core analysis techniques in stochastic modelling. Report CSR-03-7, University of Birmingham (2003). https://www.researchgate.net/publication/326827715_A_Survey_of_Out-of-Core_Analysis_Techniques_in_Stochastic_Modelling
80. Mehmood, R.: Disk-based Techniques for Efficient Solution of Large Markov Chains. Thesis (2004)
81. Mehmood, R.: Serial Disk-Based Analysis of Large Stochastic Models, pp. 230–255. Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24611-4_7
82. Mehmood, R., Crowcroft, J.: Parallel iterative solution method for large sparse linear equation systems. UCAM-CL-TR-650. Report UCAM-CL-TR-650, University of Cambridge, Computer Laboratory (2005). http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-650.pdf
83. Mehmood, R., Graham, G.: Big data logistics: a health-care transport capacity sharing model. Proc. Comput. Sci. **64**, 1107–1114 (2015). http://www.sciencedirect.com/science/article/pii/S1877050915027015. Conference on ENTERprise Information Systems/International Conference on Project MANagement/Conference on Health and Social Care Information Systems and Technologies, CENTERIS/ProjMAN/HCist 2015 October 7-9, 2015
84. Mehmood, R., Lu, J.A.: Computational Markovian analysis of large systems. J. Manuf. Technol. Manage. **22**(6), 804–817 (2011). https://doi.org/10.1108/17410381111149657
85. Mehmood, R., Parker, D., Kwiatkowska, M.: An efficient BDD-based implementation of Gauss-Seidel for CTMC analysis. Report CSR-03-13, University of Birmingham (2003). http://www.prismmodelchecker.org/bibitem.php?key=MPK03b
86. Mehmood, R., Crowcroft, J., Elmirghani, J.M.H.: A parallel implicit method for the steady-state solution of CTMCs. In: 14th IEEE International Symposium on Modeling, Analysis, and Simulation, pp. 293–302 (2006)
87. Mehmood, R., Faisal, M.A., Altowaijri, S.: Future networked healthcare systems: a review and case study. In: Boucadair, M., Jacquenet, C. (eds.) Handbook of Research on Redesigning the Future of Internet Architectures, pp. 531–558. IGI Global, Hershey, PA (2015). http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-4666-8371-6.ch022
88. Mehmood, R., Alam, F., Albogami, N.N., Katib, I., Albeshri, A., Altowaijri, S.M.: UTiLearn: a personalised ubiquitous teaching and learning system for smart societies. IEEE Access **5**, 2615–2635 (2017)
89. Mehmood, R., Meriton, R., Graham, G., Hennelly, P., Kumar, M.: Exploring the influence of big data on city transport operations: a Markovian approach. Int. J. Oper. Prod. Manage. **37**(1), 75–104 (2017). https://doi.org/10.1108/IJOPM-03-2015-0179

90. Meinke, J.H., Mohanty, S., Eisenmenger, F., Hansmann, U.H.E.: SMMP v. 3.0-simulating proteins and protein interactions in Python and Fortran. Comput. Phys. Commun. **178**, 459–470 (2008)
91. Moureau, V., Domingo, P., Vervisch, L.: Design of a massively parallel CFD code for complex geometries. Comptes Rendus Mécanique **339**(2), 141–148 (2011). http://www.sciencedirect.com/science/article/pii/S1631072110002111. High Performance Computing
92. MPI: Open MPI: Open Source High Performance Computing. http://www.open-mpi.org/
93. MPICH: MPICH | High-Performance Portable MPI. http://www.mpich.org/
94. Muhammed, T., Mehmood, R., Albeshri, A.: Enabling reliable and resilient IoT based smart city applications. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications, pp. 169–184. Springer International Publishing, Cham (2018)
95. Muhammed, T., Mehmood, R., Albeshri, A., Katib, I.: Ubehealth: a personalized ubiquitous cloud and edge-enabled networked healthcare system for smart cities. IEEE Access **6**, 32258–32285 (2018)
96. Nakajima, K.: Three-level hybrid vs. flat MPI on the earth simulator: parallel iterative solvers for finite-element method. Appl. Numer. Math. **54**(2), 237–255 (2005)
97. NAS: NAS Parallel Benchmarks. http://www.nas.nasa.gov/publications/npb.html
98. Nielsen, E.J., Diskin, B.: High-performance aerodynamic computations for aerospace applications. Parall. Comput. **64**, 20–32 (2017). http://www.sciencedirect.com/science/article/pii/S0167819117300182. High-End Computing for Next-Generation Scientific Discovery
99. Niethammer, C., Gracia, J., Knüpfer, A., Resch, M.M., Nagel, W.E.: Tools for High Performance Computing 2014: Proceedings of the 8th International Workshop on Parallel Tools for High Performance Computing, October 2014, HLRS, Stuttgart. Springer, New York (2015)
100. Nonaka, A., Almgren, A.S., Bell, J.B., Lijewski, M.J., Malone, C.M., Zingale, M.: Maestro: an adaptive low Mach number hydrodynamics algorithm for Stellar flows **188**(2), 358–383 (2010). http://dx.doi.org/10.1088/0067-0049/188/2/358
101. Oliker, L., Canning, A., Carter, J., Shalf, J., Ethier, S.: Scientific computations on modern parallel vector systems. In: Proceedings of the 2004 ACM/IEEE Conference on Supercomputing, p. 10. IEEE Computer Society, Washington (2004)
102. Oliker, L., Carter, J., Wehner, M., Canning, A., Ethier, S., Mirin, A., Parks, D., Worley, P., Kitawaki, S., Tsuda, Y.: Leading computational methods on scalar and vector HEC platforms. In: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing, p. 62. IEEE Computer Society, Washington (2005)
103. Oliker, L., Canning, A., Carter, J., Iancu, C., Lijewski, M., Kamil, S., Shalf, J., Shan, H., Strohmaier, E., Ethier, S., et al.: Scientific application performance on candidate petascale platforms. In: IEEE International Parallel and Distributed Processing Symposium, 2007 (IPDPS 2007), pp. 1–12. IEEE, New York (2007)
104. Pfrommer, B., Raczkowski, D., Canning, A., Louie, S.: Paratec (parallel total energy code), Lawrence Berkeley national laboratory (with contributions from F. Mauri, M. Cote, Y. Yoon, C. Pickard and P. Haynes). www.nersc.gov/projects/paratec
105. Pérez, F.E.H., Mukhadiyev, N., Xu, X., Sow, A., Lee, B.J., Sankaran, R., Im, H.G.: Direct numerical simulations of reacting flows with detailed chemistry using many-core/GPU acceleration. Comput. Fluids **173**, 73–79 (2018). http://www.sciencedirect.com/science/article/pii/S0045793018301786
106. Pllana, S., Brandic, I., Benkner, S.: A survey of the state of the art in performance modeling and prediction of parallel and distributed computing systems. Int. J. Comput. Intel. Res.(IJCIR) **4**, 17–26 (2008)
107. Qiang, J., Lidia, S., Ryne, R.D., Limborg-Deprey, C.: Three-dimensional quasistatic model for high brightness beam dynamics simulation. Phys. Rev. ST Accel. Beams **9**, 044204 (2006). https://link.aps.org/doi/10.1103/PhysRevSTAB.9.044204
108. Reed, D.A., Dongarra, J.: Exascale computing and big data. Commun. ACM **58**(7), 56–68 (2015). http://doi.acm.org/10.1145/2699414

109. Rudi, J., Malossi, A.C.I., Isaac, T., Stadler, G., Gurnis, M., Staar, P.W.J., Ineichen, Y., Bekas, C., Curioni, A., Ghattas, O.: An extreme-scale implicit solver for complex PDEs: highly heterogeneous flow in earth's mantle. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '15, pp. 5:1–5:12. ACM, New York, (2015). http://doi.acm.org/10.1145/2807591.2807675
110. Sáez, X., Soba, A., Sánchez, E., Kleiber, R., Castejón, F., Cela, J.M.: Improvements of the particle-in-cell code EUTERPE for petascaling machines. Comput. Phys. Commun. **182**(9), 2047–2051 (2011). http://www.sciencedirect.com/science/article/pii/S001046551000531X. Computer Physics Communications Special Edition for Conference on Computational Physics Trondheim, June 23-26, 2010
111. Schlingensiepen, J., Nemtanu, F., Mehmood, R., McCluskey, L.: Autonomic Transport Management Systems—Enabler for Smart Cities, Personalized Medicine, Participation and Industry Grid/Industry 4.0, pp. 3–35. Springer International Publishing, Cham (2016)
112. Schmidt, M.W., Baldridge, K.K., Boatz, J.A., Elbert, S.T., Gordon, M.S., Jensen, J.H., Koseki, S., Matsunaga, N., Nguyen, K.A., Su, S., et al.: General atomic and molecular electronic structure system. J. Computat. Chem. **14**(11), 1347–1363 (1993)
113. Schwarz, K., Blaha, P., Madsen, G.: Electronic structure calculations of solids using the WIEN2K package for material sciences. Comput. Phys. Commun. **147**(1), 71 – 76 (2002). http://www.sciencedirect.com/science/article/pii/S0010465502002060. Proceedings of the Europhysics Conference on Computational Physics Computational Modeling and Simulation of Complex Systems
114. Snavely, A., Gao, X., Lee, C., Carrington, L., Wolter, N., Labarta, J., Gimenez, J., Jones, P.: Performance modeling of HPC applications. In: PARCO, vol. 13, pp. 777–784 (2003)
115. Stanisic, L., Videau, B., Cronsioe, J., Degomme, A., Marangozova-Martin, V., Legrand, A., Méhaut, J.F.: Performance analysis of HPC applications on low-power embedded platforms. In: Proceedings of the Conference on Design, Automation and Test in Europe, March, pp. 475–480. EDA Consortium (2013)
116. Strunk, T., Wolf, M., Brieg, M., Klenin, K., Biewer, A., Tristram, F., Ernst, M., Kleine, P.J., Heilmann, N., Kondov, I., Wenzel, W.: Simona 1.0: An efficient and versatile framework for stochastic simulations of molecular and nanoscale systems. J. Comput. Chem. **33**(32), 2602–2613. https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.23089
117. Subbiah, A., Wasynczuk, O.: Computationally efficient simulation of high-frequency transients in power electronic circuits. IEEE Trans. Power Electron. **31**(9), 6351–6361 (2016)
118. Suma, S., Mehmood, R., Albugami, N., Katib, I., Albeshri, A.: Enabling next generation logistics and planning for smarter societies. Proc. Comput. Sci. **109**, 1122–1127 (2017). http://www.sciencedirect.com/science/article/pii/S1877050917311225. 8th International Conference on Ambient Systems, Networks and Technologies, ANT-2017 and the 7th International Conference on Sustainable Energy Information Technology, SEIT 2017, 16–19 May 2017, Madeira
119. Suma, S., Mehmood, R., Albeshri, A.: Automatic event detection in smart cities using big data analytics. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications, pp. 111–122. Springer International Publishing, Cham (2018)
120. Taboada, G.L., Touriño, J., Doallo, R.: F-MPJ: scalable java message-passing communications on parallel systems. J. Supercomput. **60**(1), 117–140 (2012)
121. Tikir, M.M., Carrington, L., Strohmaier, E., Snavely, A.: A genetic algorithms approach to modeling the performance of memory-bound computations. In: Proceedings of the 2007 ACM/IEEE conference on Supercomputing, p. 47. ACM, New York (2007)
122. Tomov, S., Nath, R., Ltaief, H., Dongarra, J.: Dense linear algebra solvers for multicore with GPU accelerators. In: 2010 IEEE International Symposium on Parallel Distributed Processing, Workshops and PhD Forum (IPDPSW), April, pp. 1–8 (2010)
123. Tomov, S., Dongarra, J., Baboulin, M.: Towards dense linear algebra for hybrid GPU accelerated manycore systems. Parall. Comput. **36**(5), 232–240 (2010). http://www.sciencedirect.com/science/article/pii/S0167819109001276. Parallel Matrix Algorithms and Applications

124. Usman, S., Mehmood, R., Katib, I.: Big data and hpc convergence: the cutting edge and outlook. In: Mehmood, R., Bhaduri, B., Katib, I., Chlamtac, I. (eds.) Smart Societies, Infrastructure, Technologies and Applications. pp. 11–26. Springer International Publishing, Cham (2018)
125. Vetter, J.S., Alam, S.R., Dunigan, T.H., Fahey, M.R., Roth, P.C., Worley, P.H.: Early evaluation of the Cray XT3. In: 20th International Parallel and Distributed Processing Symposium, 2006 (IPDPS 2006), 10 pp. IEEE, New York (2006)
126. Voorsluys, W., Garg, S.K., Buyya, R.: Provisioning spot market cloud resources to create cost-effective virtual clusters. In: Algorithms and Architectures for Parallel Processing, pp. 395–408. Springer, Berlin (2011)
127. Wolf, F., Wylie, B.J., Abrahám, E., Becker, D., Frings, W., Fürlinger, K., Geimer, M., Hermanns, M.A., Mohr, B., Moore, S., et al.: Usage of the scalasca toolset for scalable performance analysis of large-scale parallel applications. In: Tools for High Performance Computing, pp. 157–167. Springer, New York (2008)
128. Wylie, B.J.N., Geimer, M., Mohr, B., Böhme, D., Szebenyi, Z., Wolf, F.: Large-scale performance analysis of Sweep3D with the scalasca toolset. Parall. Process. Lett. **20**(04), 397–414 (2010). https://doi.org/10.1142/S0129626410000314
129. Yan, S., Zhou, Z., Dinavahi, V.: Large-scale nonlinear device-level power electronic circuit simulation on massively parallel graphics processing architectures. IEEE Trans. Power Electron. **33**(6), 4660–4678 (2018)
130. Yang, R., Gu, L., Tho, C., Sobieszczanski-Sobieski, J.: Multidisciplinary design optimization of a full vehicle with high performance computing. In: Fluid Dynamics and Co-located Conferences, June. American Institute of Aeronautics and Astronautics, Reston (2001). https://doi.org/10.2514/6.2001-1273
131. Zaki, O., Lusk, E., Gropp, W., Swider, D.: Toward scalable performance visualization with Jumpshot. Int. J. High Perform. Comput. Appl. **13**(3), 277–288 (1999)