Report from Dagstuhl Seminar 14402

Resilience in Exascale Computing

Edited by

Hermann Härtig[1], Satoshi Matsuoka[2], Frank Mueller[3], and Alexander Reinefeld[4]

1 **TU Dresden, Germany,** haertig@os.inf.tu-dresden.de
2 **Tokyo Institute of Technology, Japan,** matsutitech@gmail.com
3 **North Carolina State University, USA,** mueller@cs.ncsu.edu
4 **Zuse Institute Berlin, Germany,** reinefeld@zib.de

Abstract

From September 28 to October 1, 2014, the Dagstuhl Seminar 14402 "Resilience in Exascale Computing" was held in Schloss Dagstuhl – Leibniz Center for Informatics. During the seminar, several participants presented their current research, and ongoing work and open problems were discussed. Abstracts of the presentations given during the seminar as well as abstracts of seminar results and ideas are put together in this paper. The first section describes the seminar topics and goals in general. Links to extended abstracts or full papers are provided, if available. Slides of the talks and abstracts are available online.

Seminar September 28 to October 1, 2014 – http://www.dagstuhl.de/14402
1998 ACM Subject Classification B.3.3 Performance Analysis and Design Aids, B.5.1 Design, C.1.2 Multiple Data Stream Architectures (Multiprocessors), D.1.3 Concurrent Programming
Keywords and phrases Exascale computing, resilience, fault tolerance, manycore computers, operating systems, micro kernels, work-load balancing, checkpointing
Digital Object Identifier 10.4230/DagRep.4.9.124

1 Executive Summary

Hermann Härtig
Satoshi Matsuoka
Frank Mueller
Alexander Reinefeld

Motivation

The upcoming transition[1] from petascale to exascale computers requires the development of radically new methods of computing. Massive parallelism, delivered by manycore processors and their assembly to systems beyond 10^7 processing units will open the way to extreme computing with more than 10^{18} floating point operations per second. The large number of functional components (computing cores, memory chips, network interfaces) will greatly increase the probability of partial failures. Already today, each of the four fastest supercomputers in the TOP500 list[2] comprises more than half a million CPU cores, and this tendency

[1] IDC top ten market prediction no. 2: *The Global Petascale/Exascale Race Will Keep Shifting the Market Toward Larger Systems*, IDC, March 2013.
[2] http://www.top500.org

towards massive parallelism is expected to accelerate in the future. In such large and complex systems, component failures are the norm rather than an exception. Applications must be able to handle dynamic reconfigurations during runtime and system software is needed to provide fault tolerance (FT) at a system level. For example, Jaguar reportedly experience 20 faults per hour in production mode[3], some of which could be mitigated while others could not.

To prevent valuable computation to be lost due to failures, checkpoint/restart (C/R) has become a requirement for long running jobs. However, current C/R mechanisms are insufficient, because the communication channels between the main memory and the parallel file system are far too slow to allow to save (and restore) a complete memory dump to disk farms. As an alternative, the memory of neighboring compute nodes may be used to keep partial checkpoints, but then erasure coding must be used to prevent against the loss of data in case of single node failures. To make things worse, precious communication bandwidth is needed for writing/reading checkpoints, which slows down the application. Techniques for data compression or application-specific checkpointing (with a reduced memory footprint) were proposed as a solution, but they only alleviate the problem by a certain extent.

We assume exascale hardware architectures to consist of a heterogeneous set of computational units (ranging from general-purpose CPUs to specialized units such as today's GPUs), memory chips (RAM, flash, phase-change memory), and various kinds of interconnects. The operating system and its load balancing mechanisms need to adapt to the hardware's properties as well as to workload characteristics. With the co-existence of legacy applications and new applications, it can be assumed that exascale systems must be capable of executing a broad range of parallel programming paradigms like MPI, OpenMP, PGAS, or MapReduce. These will not always and in every case require the functionality of a fully fledged operating system. We furthermore expect applications to become more complex and dynamic. Hence, developers cannot be expected to continuously handle load balancing and reliability. It is the operating system's task to find a sweet spot that on the one hand provides generic means for load management and checkpointing, while on the other hand allowing application developers full control over the performance-relevant functionality if required.

Objectives and Expected Results

The objective of this seminar is to bring together researchers and developers with a background on HPC system software (OS, network, storage, management tools) to discuss medium to long-term approaches towards resilience in exascale computers. Two concrete outcomes are (a) outlines for alternatives for resilience at extreme scale with trade-offs and dependencies on hardware/technology advances and (b) initiation of a standardization process for a resilience API. The latter is driven by current trends of resilience libraries to let users specify important data regions required for tolerating faults and for potential recovery. Berkeley Lab's BLCR, Livermore's SCR and Capello's FTI feature such region specification in their APIs, and so do may in-house application-specific solutions. A standardized resilience API would allow application programmers to be agnostic of future underlying resilience mechanisms and policies so that resilience libraries can be exchanged at will (and might even become inter-operable). The focus of solutions is on the practical system side and should reach

[3] A. Geist, "What is the Monster in the Closet?", August 2011, Invited Talk at Workshop on Architectures I: Exascale and Beyond: Gaps in Research, Gaps in our Thinking

beyond currently established solutions. Examples of areas of interest are:

- What is the "smallest denominator" that defines a resilience API? How can the standardization of a resilience API be realized?
- How can reactive FT schemes that respond to failures be enhanced to reduce system overhead, ensure progress in computation and sustain ever shorter MTBFs?
- How should low-energy and/or persistent memory be included on nodes for checkpointing (for example PCM) and used by applications and the OS?
- Can a significant number of faults be predicted with exact locations ahead of time so that proactive FT may provide complementary capabilities to move computation away from nodes that are about to fail?
- Can message logging, incremental checkpointing and similar techniques contribute to lower checkpointing overhead?
- Is redundant execution a viable alternative at exascale? How can partly redundant execution contribute to increased resilience in exascale algorithms?
- Can algorithm-based fault tolerance be generalized to entire classes of algorithms? Can results continuously be checked?
- What is the impact of silent data corruption (SDC) on HPC computing today? Which solvers can tolerate SDCs, which ones need to be enhanced (and how)?
- How do current/novel network architectures interact with the OS (e. g., how does migration interact with RDMA)?
- How can execution jitter be reduced or tolerated on exascale systems, particularly in the presence of failures?
- Can an interface be designed that allows the application to give "hints" to the OS in terms of execution steering for resilience handling? How does this approach interact with scalability mechanisms and policies, e. g., load balancing, and with programming models, e. g., to define fault handlers?
- Do distributed communication protocols offer better resilience? How do they support coordination between node-local and inter-node scheduling?
- Does "dark silicon" offer new opportunities for resilience?
- How can I/O on exascale be efficient and resilient (e. g., in situ analysis of simulation results)?

As a result of the seminar, we expect that this list of objectives will be refined, extended, and approaches to address each of these problems will be formulated, We anticipate that participants engage in increased coordination and collaboration within the currently (mostly) separate communities of HPC system software and application development.

Furthermore, the standardization process will be kicked off. One challenge is to find the most promising context for standardization. Current HPC-related standards (MPI, OpenMP, OpenACC) do not seem suitable since resilience cuts across concrete runtime environments and may also extend beyond HPC to Clouds and data centers involving industry participants from these area (in future standardization meetings beyond the scope of this meeting).

Overall, the objective of the workshop is to spark research and standardization activities in a coordinated manner that can pave the way for tomorrow's exascale computers to the benefit of the application developers. Thus we expect not only HPC system developers to benefit by the seminar but also the community of scientific computing at large, well beyond computer science. Due to the wide range of participants (researchers and industry practitioners from the U.S., Europe, and Asia), forthcoming research work may significantly help enhance FT properties of exascale systems, and technology transfer is likely to also reach general-purpose computing with many-core parallelism and server-style computing. Specifically, the work should set the seeds for increased collaborations between institutes in Europe and the U.S./Asia.

Relation to Previous Dagstuhl Seminars

Two of the proposers, Frank Mueller and Alexander Reinefeld, previously co-organized a Dagstuhl Seminar on *Fault Tolerance in High-Performance Computing and Grids* in 2009. It provided a forum for exchanging research ideas on FT in high-performance computing and grid computing community. In the meantime, the state-of-the-art greatly advanced and it became clear, that exascale computing will not be possible without adequate means for resilience. Hence, the new seminar will be more concrete in that the pressing problems of FT for exascale computing and standardization must be tackled and solved with the joint forces of system researchers and developers.

The proposed seminar also builds on the Dagstuhl Perspective Workshop 12212 *Co-Design of Systems and Applications for Exascale*, which also relates to the DFG-funded project FFMK (http://ffmk.tudos.org/, “A Fast and Fault-tolerant Microkernel-based System for Exascale Computing”, DFG priority program 1648). Compared to the perspective workshop, our proposed seminar is much more focused on single, pressing topic of exascale computing, namely resilience.

2 Table of Contents

3 Overview of Talks

APIs, Checkpoint/Restart Systems and Resilience Benchmarks

3.1 Energy-Performance Tradeoffs in Multilevel Checkpoint Strategies

Leonardo Bautista-Gomez (Argonne National Laboratory, US)

Resilience in high-performance computing is all about protecting information. How to protect information while minimizing time, space and energy; is an open question. In this talk I will be presenting our recent work in multilevel checkpointing, lossy floating point compression, power monitoring and silent data corruption.

3.2 APIs, Architecture and Modeling for Extreme Scale Resilience

Kento Sato (Tokyo Institute of Technology, JP)

The computational power of high performance computing systems is growing exponentially, enabling finer grained scientific simulations. However, as the capability and component count of the systems increase, the overall failure rate increases accordingly. To make progress in spite of system failures, applications periodically write checkpoints to a reliable parallel file system so that the applications can restart from the last checkpoint. While simple, this conventional approach can impose huge overhead on application runtime for both checkpoint and restart operations at extreme-scale. In this presentation, to address the problem, first we introduce multi-level asynchronous checkpointing for fast checkpointing. Second, we present a fault tolerant messaging interface for fast and transparent recovery. Finally we explore new storage designs for scalable checkpoint/restart.

3.3 Portable Programming and Runtime Support for Application-Controlled Resilience

Andrew A. Chien (University of Chicago, US)

Application-aware techniques for resilience are promising approaches to efficient resilience in exascale systems. The Global View Resilience (GVR) system supports flexible, scalable, application-controlled resilience with a simple, portable abstraction – versioned, distributed arrays. Using a GVR prototype, we have evaluated the system's utility on both a number of mini-apps (miniMD, miniFE), and larger applications (a preconditioned conjugate gradient solver, OpenMC, ddcMD, and Chombo). Our results show that the programmer effort required (code change) to adopt version-based resilience is small (<1% code modified) and localized, requiring no software architecture changes. The application changes are also portable (machine-independent) and create a gentle-slope path to tolerating growing error rates in future systems. With the same applications, we evaluate the overhead of version-based resilience based on an early prototype GVR system, and find that low overheads can be achieved for all of them.

3.4 Open Discussion: Of Apples, Oranges and (Non-)reproducability

Frank Mueller (North Carolina State University, US)

Current resilience work in HPC lacks a common API and common benchmarks. The objective of this discussion was to present and adapt an API that embraces most if not all resilience methods to date in a minimal set of routines, and to identify potential candidates for resilience benchmarks, execution scenarios with and without fault injection and metrics to report.

HPC Resilience methods and Beyond

3.5 MPI Fault Tolerance: The Good, The Bad, The Ugly

Martin Schulz (LLNL – Livermore, US)

The MPI forum is currently investigating the inclusion of fault tolerance as a feature in the MPI specification. This issue has raised and continuous to raise some controversy about what MPI implementations can reasonably be expected to provide and what is useful for application developers. In this talk I will present the main proposals that are currently on the table, their advantages and the main concerns against them. The goal of this talk is to expand the discussion and to gather feedback that will help the MPI forum to come to a solution that is helpful for the larger HPC community.

3.6 Supporting the Development of Resilient Message Passing Applications

Christian Engelmann (Oak Ridge National Lab., US)

An emerging aspect of high-performance computing (HPC) hardware/software co-design is investigating performance under failure. The presented work extends the Extreme-scale Simulator (xSim), which was designed for evaluating the performance of message passing interface (MPI) applications on future HPC architectures, with fault-tolerant MPI extensions proposed by the MPI Fault Tolerance Working Group. xSim permits running MPI applications with millions of concurrent MPI ranks, while observing application performance in a simulated extreme-scale system using a lightweight parallel discrete event simulation. The newly added features offer user-level failure mitigation (ULFM) extensions at the simulated MPI layer to support algorithm-based fault tolerance (ABFT). The presented solution permits investigating performance under failure and failure handling of ABFT solutions. The newly enhanced xSim is the very first performance tool that supports ULFM and ABFT.

3.7 Fault Tolerance for Remote Memory Access Programming Models

Torsten Hoefler (ETH Zürich, CH)

Remote Memory Access (RMA) is an emerging mechanism for programming high-performance computers and datacenters. However, little work exists on resilience schemes for RMA-based applications and systems. In this paper we analyze fault tolerance for RMA and show that it is fundamentally different from resilience mechanisms targeting the message passing (MP) model. We design a model for reasoning about fault tolerance for RMA, addressing both flat and hierarchical hardware. We use this model to construct several highly-scalable mechanisms that provide efficient low- overhead in-memory checkpointing, transparent logging of remote memory accesses, and a scheme for transparent recovery of failed processes. Our protocols take into account diminishing amounts of memory per core, one of major features of future exascale machines. The implementation of our fault-tolerance scheme entails negligible additional overheads. Our reliability model shows that in-memory checkpointing and logging provide high resilience. This study enables highly-scalable resilience mechanisms for RMA and fills a research gap between fault tolerance and emerging RMA programming models.

3.8 Application level asynchronous checkpointing/restart: first experiences with GPI

Gerhard Wellein (Universität Erlangen-Nürnberg, DE)

Automatic check-point/restart is one potential way to address the resilience challenge of exascale computing. This talk presents first experiences of a prototypical application that is able to recover itself after detecting a failed node. It has been accomplished using GPI communication library in combination with checkpoint/restart technique. We investigate the potential of asynchronous checkpointing both to neighboring nodes and parallel file systems to reduce or even hide the costs of checkpointing.

Resilience Models

3.9 Operating System Support for Redundant Multithreading

Björn Döbel (TU Dresden, DE)

Implementing fault tolerance methods in software allows to protect commercial-off-the-shelf computer systems against the effects of transient and permanent hardware errors. In this talk I am going to present ROMAIN, an operating system service that provides transparent replication to binary-only applications on top of the L4 microkernel. I will describe how ROMAIN supports replication of multithreaded applications and replication of accesses to shared-memory channels between applications.

In the second part of the talk I am going to discuss how the experiences we gained while developing ROMAIN might benefit the HPC community and which of ROMAIN's assumptions need to be adjusted in this context.

3.10 Resilient gossip algorithms for online management of exascale clusters

Amnon Barak (The Hebrew University of Jerusalem, IL)

Management of forthcoming exascale clusters requires frequent collection and sharing of runtime information about the health of the nodes, their resources and the running applications. We present a new paradigm for online management of scalable clusters, consisting of a large number of computing nodes (nodes) and a small number of servers (masters) that manage these nodes. We describe the details of gossip algorithms for sharing local information within subsets (colonies) of nodes and for sending selected global information to the masters, which hold recent information on all the nodes.

The presented algorithms are decentralized and resilient – they can work well when some nodes are down and without needing any recovery protocol. We give formal expressions for approximating the average age of the local information at each node and the average age of the collected information at a master. We then show that the results of these approximations closely match the results of simulations and measurements on a real cluster for different-size colonies.

The main outcome of this study is that a division of a large cluster to colonies can reduce the overall number of messages and the average load at the masters.

3.11 FFMK: Towards a fast and fault-tolerant micro-kernel-based Operating System

Hermann Härtig (TU Dresden, DE)

FFMK's architecture is based on a combination of well-proven technologies: the L4 micro-kernel and L4-based virtualization, on-line management algorithms as used in the MosiX system, coding as used in RAID, and the XTreem-FS distributed file-system. I plan to explain the overall architecture and then start discussing the points in the architecture that require special attention for fault tolerance. My hope is, that – in interaction with the audience – we will arrive at a clearer idea where which types of fault-tolerance measures are needed or possible in view of requirements and fault models for exa-scale systems.

3.12 Open Discussion: A Holistic Model for Resilience

Hermann Härtig (TU Dresden, DE)

The following discussion suggested that a failure-model-based systematic analysis is needed rather than an ad-hoc discussion. It was suggested that the notion of *containment domains* makes sense as a starting point. Overall, a holistic view on fault tolerance techniques for exascale systems does yet exist.

Soft Errors

3.13 Memory Errors in Modern Systems

Vilas Sridharan (Advanced Micro Devices, Inc. – Boxborough, US)

This talk presents fault and error data collected from several production systems in the field, and uses this data to project to node hardware reliability in an exascale timeframe.

3.14 Fault Tolerance for Iterative Linear Solvers

James J. Elliott (North Carolina State University, US)

Computer hardware trends may expose incorrect computation or storage to application codes. Silent data corruption (SDC) will likely be infrequent, yet one SDC suffices to make numerical algorithms like iterative linear solvers cease progress towards the correct answer. Initially, we focus our efforts on the resilience of the iterative linear solver GMRES to a single transient SDC. Our experiments show that when GMRES is used as the inner solver of an inner-outer iteration, it can "run through" SDC of almost any magnitude in the computationally intensive orthogonalization phase. That is, it gets the right answer using faulty data without any required roll back. Those SDCs, which it cannot run through, are caught by our detection scheme. We analyze our solvers in the presence of multiple faults, and discuss how fault rates and fault detection influences iterative solver selection.

3.15 Scalable Fault Tolerance at the Extreme Scale

Zizhong Chen (University of California – Riverside, US)

URL http://www.dagstuhl.de/mat/Files/14/14402/14402.ChenZizhong.Slides.pdf

Extreme scale supercomputers available before the end of this decade are expected to have 100 million to 1 billion computing cores. Due to the large number of components involved, extreme scale scientific applications must be protected from errors. When an error occurs, the affected application either continues or stops. If the application continues, we call it a fail-continue error. Otherwise, we call it a fail-stop error. In this talk, In this talk, I will discuss our recent work on scalable fault tolerance at the extreme scale. We have developed some highly efficient techniques for selected widely used scientific algorithms to tolerate both fail-continue and fail-stop errors according to their specific algorithmic characteristics. The algorithms we consider include direct methods for solving dense linear systems and eigenvalue problems, iterative methods for solving sparse linear systems and eigenvalue problems, and Newton's method for solving systems of non-linear equations and optimization problems. By leveraging the algorithmic characteristics of these algorithms, the proposed techniques can achieve much higher efficiency than the traditional general techniques (i. e., Triple Modular Redundancy for fail-continue errors and checkpoint for fail-stop errors) and therefore have potential to scale to exascale and beyond. A highly scalable checkpointing scheme is also developed for general applications.

3.16 Algorithms for coping with silent errors

Yves Robert (ENS – Lyon, FR & University of Tennessee, US)

Joint work of Guillaume Aupy, Anne Benoit, Aurélien Cavelan, Thomas Hérault, Saurabh K. Raina, Hongyang Sun, Frédéric Vivien, and Dounia Zaidouni
URL http://www.dagstuhl.de/mat/Files/14/14402/14402.RobertYves.Slides.pdf

Silent errors have become a major problem for large-scale distributed systems. Detection is hard, and correction is even harder. This talks presents generic algorithms to achieve both detection and correction of silent errors, by coupling verification mechanisms and checkpointing protocols. Application-specific techniques will also be investigated for sparse numerical linear algebra.

3.17 Assessing the impact of composite strategies for resilience

George Bosilca (University of Tennessee, US)

With the advances in the theoretical and practical understanding of algorithmic traits enabling Algorithm Based Fault Tolerant (ABFT) approaches, a growing number of frequently used algorithms have been proven ABFT- capable. In the context of larger applications, these algorithms provide a temporal section of the execution when the data is protected by it's own intrinsic properties, and can be algorithmically recomputed without the need of checkpoints. However, while typical scientific applications spend a significant fraction of their execution time in library calls that can be ABFT- protected, they interleave sections that are difficult or even impossible to protect with ABFT. As a consequence, the only fault- tolerance approach that is currently used for these applications is checkpoint/restart. In this talk I will present a model to investigate the efficiency of a composite protocol, that alternates between ABFT and checkpoint/restart for effective protection of an iterative application composed of ABFT-aware and ABFT- unaware sections.

3.18 Leveraging PGAS Models for Hard and Soft Errors at Scale

Abhinav Vishnu (Pacific Northwest National Lab. – Richland, US)

PGAS Models are finding increasing adoption in the community due to their productivity, asynchronous communication and high performance. In this talk, we will present research conducted at PNNL for leveraging PGAS programming models for hard faults and consider methods for soft error detection and correction using NWChem – a large scale computational chemistry application. A significant portion of the talk would be presenting the lessons learned and gaps which should be addressed in the resilience research.

3.19 Open Discussion: Soft Error

Satoshi Matsuoka (Tokyo Institute of Technology, JP)

The so-called 'soft-errors' or undetected 'silent' errors are deemed to be one of the roadblocks as systems such as supercomputers and IDCs growing exponentially large, and the overall system error rate increasing proportionally. However, we often tend to forget that, unlike embedded, autonomous systems, for scientific computing there are always humans in the loop validating the results. In such a scenario, will soft errors matters as anticipated? We conduct qualitative analysis based on TSUBAME2.0's fault statics collected over a year period, to argue that soft errors may not be serious given the improvements in HW reliability as well as humans re-evaluating false positives due to soft errors, to the extent that the entire problem space could be dealt with traditional detection and recovery techniques for hard-stop failures.

Supporting Frameworks

3.20 Abstractions and mechanisms for proportional resilience

Mattan Erez (University of Texas – Austin, US)

Systems and applications have dynamic reliability characteristics and requirements. As a result, opportunities, and perhaps even a requirement, exist for co-tuning resilience across system and application layers. Many challenges must be addressed for co-tuning to be successful, especially in bridging the gaps between layers. In this talk I will discuss a few thoughts, examples, and questions related to mechanisms and abstractions (framed by Containment Domains) for addressing some of these challenges, including: how should hardware error and detector properties and models be presented to the application? How can an application specify non- traditional recovery and error tolerance? How can dynamic applications and load-balancing be modeled w.r.t. time and energy? What metrics should be optimized?

3.21 A Non-checkpoint/restart, Non-algorithm-specific Approach to Fault-tolerance

Dorian C. Arnold (University of New Mexico – Albuquerque, US)

Hierarchical or tree-based overlay networks (TBONs) are often used to execute data aggregation operations in a scalable, piecewise fashion. We present state compensation, a scalable failure recovery model for high-bandwidth, low-latency TBON computations. By leveraging inherently redundant state information found in many TBON computations, state compensation avoids explicit state replication (for example, process checkpoints and message logging) and incurs no overhead in the absence of failures. Further, when failures do occur, state compensation uses a weak data consistency model and localized protocols that allow processes to recover from failures independently and responsively. We describe the fundamental state compensation concepts and a prototype implementation integrated into the MRNet TBON infrastructure. Our experiments with this framework suggest that for TBONs supporting up to millions of application processes, state compensation can yield millisecond recovery latencies and inconsequential application perturbation.

3.22 Dynamic Resource Management and Scheduling for Fault Tolerance

Felix Wolf (GRS for Simulation Sciences – Aachen, DE)

To dynamically recover from node failure, a parallel job usually needs to replace the failed nodes. While the static allocation of spare nodes is technically simpler, pooling spare nodes across all jobs and allocating them dynamically is more efficient in terms of the number of required spare nodes but requires dynamic resource management. In this talk, we present an extension of the Torque/Maui batch systems to support dynamic node allocation for running parallel jobs and discuss how it can support fault-tolerant applications in the re-acquisition of failed nodes. As long as the parallel runtime system can continue running the application with replaced nodes, they can be obtained from the batch system through an API.

4 Conclusion

Through the lively engagement of all participants, the seminar was very successful and conducted in a professional, friendly and collegiate atmosphere supported by the kind and helpful staff at Schloss Dagstuhl. Lively discussions continued every day well beyond meeting times. The group meeting in its three-day format combined with the cozy confinement of Dagstuhl provided an umbrella for thoughtful discussion that conferences or workshops cannot provide. This helped create a community feeling that could become a building block for a concerted effort to coordinate future research activities, cooperate in outreach effort and maximize everyone's productivity and impact in fulling together each one's unique expertise for a combined effort to successfully solve the grand challenges of resilience in exascale HPC and beyond. During the meeting, follow-up action items with community-building character were identified, as detailed in the online discussion notes of the discussion sessions. They include (a) creation of a mailing list to coordinate activities and disseminate information on FT in high-performance computing and related areas, (b) development a standard set of benchmarks and an API for arbitrary resilience mechanisms, and (c) organization of follow-on meetings for the community. Within one month of the seminar, action item (a) have been realized and the benchmarks set (b) and a follow-up meeting (c) are in the planning stages.

Participants

- Dorian C. Arnold
University of New Mexico – Albuquerque, US
- Amnon Barak
The Hebrew University of Jerusalem, IL
- Leonardo Bautista-Gomez
Argonne National Laboratory, US
- George Bosilca
University of Tennessee, US
- Zizhong Chen
University of California – Riverside, US
- Andrew A. Chien
University of Chicago, US
- Nathan DeBardeleben
Los Alamos National Lab., US
- Björn Döbel
TU Dresden, DE
- James J. Elliott
North Carolina State Univ., US
- Christian Engelmann
Oak Ridge National Lab., US
- Mattan Erez
University of Texas – Austin, US
- Hermann Härtig
TU Dresden, DE
- Torsten Hoefler
ETH Zürich, CH
- Larry Kaplan
Cray Inc. – Seattle, US
- Dieter Kranzlmüller
LMU München, DE
- Matthias Lieber
TU Dresden, DE
- Naoya Maruyama
RIKEN – Kobe, JP
- Satoshi Matsuoka
Tokyo Institute of Technology, JP
- Frank Mueller
North Carolina State Univ., US
- Alexander Reinefeld
Konrad-Zuse-Zentrum – Berlin, DE
- Yves Robert
ENS – Lyon, FR
- Robert B. Ross
Argonne National Laboratory, US
- Kento Sato
Tokyo Institute of Technology, JP
- Thorsten Schütt
Konrad-Zuse-Zentrum – Berlin, DE
- Martin Schulz
LLNL – Livermore, US
- Vilas Sridharan
Advanced Micro Devices, Inc. – Boxborough, US
- Thomas Steinke
Konrad-Zuse-Zentrum – Berlin, DE
- Jeffrey Vetter
Oak Ridge National Lab., US
- Abhinav Vishnu
Pacific Northwest National Laboratory – Richland, US
- Gerhard Wellein
Univ. Erlangen-Nürnberg, DE
- Felix Wolf
GRS for Simulation Sciences – Aachen, DE