

# Adaptive Resource Management for HPC Systems

Edited by

Michael Gerndt<sup>1</sup>, Masaaki Kondo<sup>2</sup>, Barton P. Miller<sup>3</sup>, and Tapasya Patki<sup>4</sup>

1 TU München, DE, [gerndt@in.tum.de](mailto:gerndt@in.tum.de)

2 Keio University – Yokohama, JP, [kondo@acs1.ics.keio.ac.jp](mailto:kondo@acs1.ics.keio.ac.jp)

3 University of Wisconsin-Madison, US, [bart@cs.wisc.edu](mailto:bart@cs.wisc.edu)

4 LLNL – Livermore, US, [patki1@llnl.gov](mailto:patki1@llnl.gov)

---

## Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 21441 “Adaptive Resource Management for HPC Systems”. The seminar investigated the impact of adaptive resource management of malleable applications on the management of the HPC system, the programming of the applications, the tools for performance analysis and tuning, as well as the efficient usage of the HPC systems. The discussions led to a joint summary presenting the state-of-the-art, required techniques on the various levels of HPC systems, as well as the foreseen advantages of adaptive resource management.

**Seminar** November 1–5, 2021 – <http://www.dagstuhl.de/21441>

**2012 ACM Subject Classification** Computer systems organization → Distributed architectures; Software and its engineering → Parallel programming languages; Software and its engineering → Process management

**Keywords and phrases** High Performance Computing, dynamic resource management, malleable HPC applications, power management, computing continuum

**Digital Object Identifier** 10.4230/DagRep.11.10.1

## 1 Executive Summary

*Michael Gerndt*

*Masaaki Kondo*

*Barton P. Miller*

*Tapasya Patki*

**License** © Creative Commons BY 4.0 International license  
© Michael Gerndt, Masaaki Kondo, Barton P. Miller, and Tapasya Patki

Today’s supercomputers have very static resource management. Jobs are submitted via batch scripts to the resource manager, then scheduled on the machine with a fixed set of nodes. Other resources, such as power, network bandwidth and storage are not actively managed and are provided only on a best-effort basis. This inflexible, node-focused and static resource management will have to change in the future due to many reasons, some of them listed below.

First, applications are becoming increasingly more dynamic. Techniques such as adaptive mesh refinement, e.g., as used in Tsunami simulations, lead to scalability changes over the application’s execution. Furthermore, only some application phases might profit from specialized accelerators, and I/O phases might even run best with a limited number of compute resources.



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 4.0 International license

Adaptive Resource Management for HPC Systems, *Dagstuhl Reports*, Vol. 11, Issue 10, pp. 1–19

Editors: Michael Gerndt, Masaaki Kondo, Barton P. Miller, and Tapasya Patki



Dagstuhl Reports

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Additionally, the execution environment of applications is also becoming dynamic. Modern processors change the clock frequency according to the instruction mix as well as power and thermal envelopes. Heavy use of the vector units can lead to a lower clock frequency to stay in the thermal power budget, for example.

As an independent concern, due to the sheer number of components, failure rates are expected to increase thus slowing down computation or even leading to an increased number of node failures.

Finally, the upcoming machines will be power constrained, which means that the power will have to be carefully distributed among all running applications. The resulting power capping will impact the application's performance due to adaptation of the clock frequency and due to manufacturing variability. These challenges in HPC will only be solvable by using a more adaptive resource management approach. For example, compute nodes need to be redistributed among running applications to adapt to changes in the application's resource requirements either due to a varying number of grid points or interspersed algorithmic phases that profit from certain accelerators; network and I/O bandwidth will have to be assigned to applications to avoid interference caused by contention of concurrent communication and I/O phases; power needs to be dynamically redistributed both within an application and across applications to enable increased efficiency. Dynamic redistribution of resources will also give more flexibility to the resource manager to schedule jobs on the available resources and thus reduce idle times and efficiency lowering contention scenarios, e.g., in the situation of big jobs waiting for execution.

This Dagstuhl Seminar investigated a holistic, layered approach for adaptive resource management. It started with the resource management layer being responsible for scheduling applications on the machine and dynamically allocating resources to the running applications. At the programming level, applications need to be programmed in a resource-aware style such that they can adapt to resource changes and can make most efficient usage of the resources. On top of the programming interfaces, programming tools have to be available that allow the application developers to analyze and tune the applications for the varying amount of available resources. At the application level, applications have to be redesigned to enable significant gains in efficiency and throughput, e.g., adaptive mesh refinement, approximate computing, and power-aware algorithms are a few aspects to mention here.

The discussions led to a joint summary presenting the state-of-the-art, required techniques on these layers of HPC systems, as well as the foreseen advantages of adaptive resource management.

## 2 Table of Contents

### Executive Summary

<i>Michael Gerndt, Masaaki Kondo, Barton P. Miller, and Tapasya Patki . . . . .</i>	1
---	---

### Overview of Talks

Flux: Next-Generation Resource Management and Scheduling for Scientific Workflow Enablement <i>Dong Ahn . . . . .</i>	5
Ongoing Efforts on Co-scheduling and Holistic Power Management <i>Eishi Arima . . . . .</i>	5
A Scalable RISC-V Power Controller Platform for HPC Processors <i>Andrea Bartolini . . . . .</i>	5
The Malleability Problem Statement: Differences between Supercomputing and the Cloud <i>Isaías A. Comprés Ureña . . . . .</i>	6
Towards Machine Learning Generation of Parallel Applications Performance Models <i>Eduardo César and Anna Sikora . . . . .</i>	6
Invasive Computing in HPC <i>Michael Gerndt . . . . .</i>	7
Towards Dynamic Node Resource Management in Next-Generation HPC Environments <i>Balazs Gerofti . . . . .</i>	8
Challenges of Resource Management on the “Data Platform”: mdx <i>Toshihiro Hanawa . . . . .</i>	8
InvasiC HPC Programming: iMPI and EPOP <i>Jophin John and Santiago Narvaez Rivas . . . . .</i>	8
From GEOPM to the OIEP Reference Model: Embedding Energy and Power Runtime Systems into the Big Picture of HPC <i>Matthias Maiterth . . . . .</i>	9
Converged Computing: Combining the Best of HPC and the Cloud <i>Daniel John Milroy . . . . .</i>	9
Overview of State-of-the-Art Parallel Performance Measurement and Analysis Tools for heterogeneous systems <i>Bernd Mohr . . . . .</i>	10
Dynamism in HPC Resource Control <i>Frank Mueller . . . . .</i>	10
Dynamic Tuning of HPC Applications – ESPRESO FEM Library <i>Lubomír Ríha . . . . .</i>	11
Invasive Computing – A Systems Programming Perspective <i>Wolfgang Schröder-Preikschat . . . . .</i>	11

From COMM\_WORLD to Sessions and Bubbles: How new MPI Features Need to Interact with Resource Managers  
*Martin Schulz* . . . . . 11

Dynamic Tuning of HPC Applications – MERIC  
*Ondrej Vysocky* . . . . . 12

The Price Performance of Performance Models  
*Felix Wolf* . . . . . 12

**Outlook: Techniques for Malleability-Enabled Machines**

Resource Management . . . . . 12

Programming Models: MPI and Beyond . . . . . 13

Unified Monitoring . . . . . 14

Turning Applications into Malleable Applications . . . . . 15

Incentives to Help in Dynamic Resource Management . . . . . 16

What is the Potential of Malleable Applications? . . . . . 16

Converged Computing . . . . . 17

**Participants** . . . . . 19

**Remote Participants** . . . . . 19

## 3 Overview of Talks

### 3.1 Flux: Next-Generation Resource Management and Scheduling for Scientific Workflow Enablement

*Dong Ahn (LLNL – Livermore, US)*

License  Creative Commons BY 4.0 International license  
© Dong Ahn

Many emerging scientific workflows that target high-end HPC systems require complex interplay with the resource and job management software (RJMS). However, portable, efficient and easy-to-use scheduling and execution of these workflows is still an unsolved problem. In this talk, I will present Flux, a next-generation RJMS designed specifically to address the key scheduling challenges of modern workflows in a scalable, easy-to-use, and portable manner. At the heart of Flux lies its ability to be seamlessly nested within batch allocations created by itself as well as other system schedulers (e.g., SLURM, MOAB, LSF, etc), serving the target workflows as their “personalized RJMS instances”. In particular, Flux’s consistent and rich set of well-defined APIs portably and efficiently support those workflows that can often feature non-traditional execution patterns such as requirements for complex co-scheduling, massive ensembles of small jobs and coordination among jobs in an ensemble. As part of this talk, I will also discuss Flux’s graph-based resource data model, Flux’s response to needing to schedule increasingly diverse resources, and how this model is becoming the center of our industry co-design efforts: for example, multi-tiered storage scheduling co-design with HPE and Cloud resource co-design with IBM T.J. Watson and RedHat OpenShift.

### 3.2 Ongoing Efforts on Co-scheduling and Holistic Power Management


*Eishi Arima (TU München, DE)*

License  Creative Commons BY 4.0 International license  
© Eishi Arima

This presentation covers our ongoing efforts related to co-scheduling, i.e., colocating multiple jobs at the same time on a node, and also holistic power management for HPC systems. More specifically, the talk will include: (1) open software architecture for sophisticated resource management, in particular power stack, and integrating our software stack based on the architecture; (2) a variety of co-scheduling studies for sophisticating them; and (3) opportunities to co-ordinate with the malleability.

### 3.3 A Scalable RISC-V Power Controller Platform for HPC Processors

*Andrea Bartolini (University of Bologna, IT)*

License  Creative Commons BY 4.0 International license  
© Andrea Bartolini

Today’s high-performance computing (HPC) workloads crave data bandwidth, capacity and floating-point performance. High-performance chips feature with many performance-capable cores, vector units, DDRs and HBMs memory controllers’, high-bandwidth and low-latency

coherent I/Os, as well as domain-specific accelerators with staggering (several hundreds of Watts) peak power requirements. The peak power exceeds the TDP, and the package cost constrains the maximum TDP and sustainable peak power. Motherboards' form-factor, layout, and cost constraint the power distribution design and demand effective and reliable on-chip thermal management. Power, temperature, and energy are critical aspects that must be controlled and optimized online with a low-latency feedback loop with the on-chip power management IPs and sensors, Operating System, Security Subsystem, off-chip Board Management Controller (BMC) and power converters. We propose ControlPULP, a fully-digital and highly capable RISC-V based parallel microcontroller IP optimized for power management of complex HPC processors. Its design supports a single-core manager core and peripherals paired with a cluster of 8 processors to accelerate the Power Control Firmware workload, Direct Memory Access (DMA) engine for accessing on-chip sensors, a uDMA engine for off-chip AVSBus/PMBUS peripheral support and BMC-based communication through the Management Component Transport Protocol (MCTP). The controller implements basic System Control and Management Interface (SCMI) doorbell-based protocol hosting up to 144 external interrupt lines. On the SW side, it relies on an open-source Real-time Operating System (FreeRTOS) for agile scheduling of the underlying Control Policy.

### 3.4 The Malleability Problem Statement: Differences between Supercomputing and the Cloud

*Isaías A. Comprés Ureña (TU München, DE)*

**License** © Creative Commons BY 4.0 International license  
© Isaías A. Comprés Ureña

Malleability can bring benefits to our distributed memory supercomputers that are not possible with current static allocations. Malleability has already been achieved in cloud computing systems. The reasons are related to the differences between the workloads that are typical across these systems. The workloads of supercomputing pose additional challenges that have caused delays in the deployment of malleability in this domain.

### 3.5 Towards Machine Learning Generation of Parallel Applications Performance Models

*Eduardo César (Autonomous University of Barcelona, ES), Anna Sikora (Autonomous University of Barcelona, ES)*

**License** © Creative Commons BY 4.0 International license  
© Eduardo César and Anna Sikora

**Joint work of** Eduardo César, Jordi Alcaraz, Anna Sikora

**Main reference** Jordi Alcaraz, Steven Sleder, Ali Tehrani Jamsaz, Anna Sikora, Ali Jannesari, Joan Sorribes, Eduardo César: "Building representative and balanced datasets of OpenMP parallel regions", in Proc. of the 29th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP 2021, Valladolid, Spain, March 10-12, 2021, pp. 67-74, IEEE, 2021.

**URL** <http://dx.doi.org/10.1109/PDP52278.2021.00019>

Malleable HPC computing will require, among other things, efficient methods for estimating the amount of resources that an application needs to be executed efficiently. However, due to the heterogeneity found in HPC systems, adequate analytical models for performance improvement can be very difficult to generate. An alternative to traditional analytical models

could be the use of machine learning algorithms, which may help to automatically create performance models to predict the appropriate configuration for one or multiple application's parameters.

Incorporating machine learning for automatic performance analysis and tuning is a promising path, but it introduces the need for generating balanced and representative datasets of parallel applications' executions.

First, to be able to build performance models, measurements are needed to calculate or select the proper values for one or multiple parameters which can impact performance. The selection of the right measurements is important as information can be redundant or, in the worst case, insufficient to correctly describe the relationship between them and performance parameters.

Second, these measurements should be used for building datasets of representative parallel code regions patterns. Thus, a methodology is needed for determining whether a given region covers a unique part of the input space not yet covered by the patterns already included in the dataset.

Finally, when such a dataset is used for performance tuning, an imbalance problem appears as the targets are now performance parameters instead of representative code regions. This imbalance appears because some performance parameters' values generally provide better performance than others for most cases. Consequently, machine learning algorithms may under-perform due to underrepresented cases, making the use of techniques to counter this natural imbalance necessary.

### 3.6 Invasive Computing in HPC

*Michael Gerndt (TU München, DE)*

License  Creative Commons BY 4.0 International license  
© Michael Gerndt


TUM started the research on malleable HPC application as part of the Transregional Research Center TRR89 "Invasive Computing". This is a collaboration between the FAU Erlangen, KIT Karlsruhe, and Technische Universität München. The major focus is to develop concepts for resource-aware programming of embedded application for highly parallel chip multiprocessors.

TUM is investigating the extension of this concept for HPC applications. The resource management of HPC systems is static. The system is partitioned for system services, interactive access, and batch jobs. Nodes are assigned to applications for the applications whole lifetime. The concept of dynamic resource management for HPC systems allows to distribute resources dynamically to system services and running applications, and thus allows for a more efficient sharing of the resources.

TUM developed extensions of OMP and MPI, known as iMPI, for writing malleable MPI applications. Furthermore, a scalable in-memory application-level checkpointing system iCheck is under development. The benefits demonstrated in the TRR are better system utilization, more efficient resource usage, and a dynamic power corridor management. The work of invasive computing is continued in the European HPC project Deep-Sea.

### 3.7 Towards Dynamic Node Resource Management in Next-Generation HPC Environments


*Balazs Gerofi (RIKEN – Kobe, JP)*

License  Creative Commons BY 4.0 International license  
© Balazs Gerofi

Workload diversity in high-performance computing (HPC) environments has experienced an explosion in recent years. The increasing prevalence of Big Data processing, in-situ analytics, artificial intelligence (AI) and machine learning (ML) workloads, as well as multi-component workflows is pushing the limits of supercomputing systems that have been primarily designed to serve parallel simulations. In addition, with the growing complexity of the hardware there is also a growing interest for multi-tenancy and for a more dynamic, cloud-like execution environment. All these trends bring together a large variety of runtime components that do not cooperate well with each other, which in turn can lead to suboptimal performance. This talk will enumerate a number of representative workloads that stress the limitations of the traditional HPC center. We then highlight some of the underlying forces which shape requirements of next generation systems and propose a cross-stack coordination layer that aims to resolve these conflicts. Finally, through some of our previous efforts in this space we demonstrate the benefits of the overall approach.

### 3.8 Challenges of Resource Management on the “Data Platform”: mdx

*Toshihiro Hanawa (University of Tokyo, JP)*

License  Creative Commons BY 4.0 International license  
© Toshihiro Hanawa

mdx is the infrastructure for leveraging data all over Japan that enables (1) a rapid PoC environment for R&D data leveraging activities including industry-academia-government collaboration projects (2) wide-area virtual private infrastructure with high performance computing and storage resources (3) realtime data processing with security.

mdx introduces a virtualization technology like multi-tenant cloud. On the other hand due to the resource limitation, we have to efficiently manage the resource and consider to offload heavy-load, large-capacity processing to Supercomputer.

### 3.9 InvasiC HPC Programming: iMPI and EPOP

*Jophin John (TU München, DE) and Santiago Narvaez Rivas (TU München, DE)*

License  Creative Commons BY 4.0 International license  
© Jophin John and Santiago Narvaez Rivas

As more and more emphasis is given to the malleable resources and adaptive resource management, it is necessary to facilitate programming models that enable the application programmers to exploit this dynamism. Our talk focussed on programming models for malleable application development, specifically the malleable MPI API (iMPI) provided by the invasive infrastructure developed as part of the InvasIC project (TCRC 89 “Invasive Computing”). A tsunami simulation code (eSamoa) was adapted using such extension,



showing that, albeit the runtime of malleable applications might increase with respect to static ones, the resources are used more efficiently. During the development of the eSamoa, several challenges were discovered. In particular, both the redistribution of data after an adaptation occur, as well as keeping the logical flow of the application consistent, proved to be relevant. To tackle the latter, we proposed a new phase-oriented programming extension, namely the Elastic Phase Oriented Programming (EPOP) model on top of iMPI, that facilitates easier malleable application development. We also discussed a scenario of system-level power management using the EPOP based malleable model.

Along with application malleability, our discussion also extended to dynamic fault-tolerant systems for malleable and non-malleable distributed-memory applications. We talked about iCheck, an invasive checkpointing system that could dynamically increase and decrease the resources for checkpointing. Additionally, using such a system for data redistribution among malleable applications will benefit malleable application development. We showed results that emphasize dynamism and provides better and faster checkpointing abilities.

### 3.10 From GEOPM to the OIEP Reference Model: Embedding Energy and Power Runtime Systems into the Big Picture of HPC

*Matthias Maiterth (TU München, DE)*

License  Creative Commons BY 4.0 International license  
© Matthias Maiterth

The talk briefly introduces GEOPM, PowerStack and the OIEP Reference Model, and shows their connection.

The presentations initially gives an overview of the GEOPM runtime and shows the software infrastructure provided by the GEOPM framework.

Since tools do not exist in isolation, a sound setup of tools (such as GEOPM) have to exist in a software echo system. The definition of a software stack for energy and power was addressed by the PowerStack effort with still an open outcome.

The later part of the presentation shows the presenters Dissertation work, where the OIEP reference model is presented, giving vocabulary and a method for representing and arranging energy and power management setups in HPC. This is a missing foundation for efforts such as the PowerStack and other tools, which so far often only consider specialized setups or even lack integration in a holistic energy and power management setup.

### 3.11 Converged Computing: Combining the Best of HPC and the Cloud

*Daniel John Milroy (LLNL – Livermore, US)*


License  Creative Commons BY 4.0 International license  
© Daniel John Milroy

Since the early 2000s, computing has relied on increasing levels of parallelism together with Moore's law to drive performance improvement. As Moore's law now begins to taper, demand for increasing performance and new capabilities is spurring development of heterogeneous and dynamic systems and new software environments. Large-scale scientific applications are adapting to use the new tools and technologies and pushing computing boundaries through multi-component workflows.

This talk describes work on facilitating cutting-edge current and next-generation scientific workflows through integration of cloud computing with Flux, a novel graph-based Resource and Job Management Software (RJMS) developed at LLNL. The integration is aimed to advance converged computing, an environment that offers the best features of HPC (performance, efficiency, sophisticated scheduling) and the cloud (resiliency, elasticity, portability, and automation) to next-generation high-performance workflows. The talk will also detail work to build industry collaborations to make lasting, sustainable contributions to the broader computing community.

### 3.12 Overview of State-of-the-Art Parallel Performance Measurement and Analysis Tools for heterogeneous systems

*Bernd Mohr (Jülich Supercomputing Centre, DE)*

License  Creative Commons BY 4.0 International license  
© Bernd Mohr

Current HPC systems consist of complex configurations of potentially heterogeneous components. In addition, the hard- and software configuration can change dynamically due to fault recovering processes or power saving efforts. Deep hierarchies of large, complex software components are needed to operate them. Developing efficient and high performance application software for these systems is challenging. Therefore, sophisticated performance measurement and analysis capabilities are required. The talk will present an overview of state-of-the-art parallel performance measurement and analysis tools, high-lightening the scalability of the tools and their support for current heterogeneous node architectures.

### 3.13 Dynamism in HPC Resource Control

*Frank Mueller (North Carolina State University – Raleigh, US)*

License  Creative Commons BY 4.0 International license  
© Frank Mueller

This talk provides an overview of our recent work on dynamic resource control in HPC environments. First, power controls are discussed for phase-changing applications. Second, performance and power for application execution over hybrid DRAM/non-volatile memory is characterized and used to provide guided allocation within both memory spaces to reduce energy while maintaining performance. Third, a method for co-scheduling of jobs on multiple heterogeneous accelerators is developed. Fourth, HPC resilience is extended to workflows and jobs are scheduled on heterogeneous nodes according to their resource needs to trade off response time, utilization, and cost for HPC and cloud allocations. Overall, HPC resource control is becoming increasingly dynamic. Future work needs to coordinate different control mechanisms to achieve higher-level objectives in terms of workload and center objectives.

### 3.14 Dynamic Tuning of HPC Applications – ESPRESO FEM Library

*Lubomír Riha (VSB-Technical University of Ostrava, CZ)*

License  Creative Commons BY 4.0 International license  
© Lubomír Riha

This talk introduces the ESPRESO FEM library developed at IT4Innovations. The library was described from computer science point of view, and we highlighted its potential for dynamic resource management. The key component of ESPRESO that enables its elasticity is the I/O module which is capable of checkpoint / restart simulation on various number of MPI ranks. Finally, we have proposed changes needed in this module to fully support iMPI.

### 3.15 Invasive Computing – A Systems Programming Perspective

*Wolfgang Schröder-Preikschat (Universität Erlangen-Nürnberg, DE)*

License  Creative Commons BY 4.0 International license  
© Wolfgang Schröder-Preikschat

Invasive Computing is a research program that aims at developing a new paradigm to address the hardware- and software challenges of managing and using massively-parallel MPSoCs of the years 2020 and beyond. The program follows the idea of a corresponding Transregional Collaborative Research Center funded by the DFG in its third four-year period (2018-2022). It currently comprises seventeen projects from the areas of computer architecture, system software, programming systems, algorithm engineering and applications. Basic concept is to let applications manage the available computing resources on a local scope and to provide means for a dynamic and fine-grained expansion and contraction of parallelism. This talk provides a brief overview of the program and presents thoughts and intermediate results on system software support for it.

### 3.16 From COMM\_WORLD to Sessions and Bubbles: How new MPI Features Need to Interact with Resource Managers

*Martin Schulz (TU München, DE)*

License  Creative Commons BY 4.0 International license  
© Martin Schulz

MPI 4.0 introduced the new concept of MPI Sessions, which enables a new way of MPI initialization and resource management. While currently still defined as an interface for static resources, it provides the needed base mechanisms to develop a more dynamic view. In this talk I will introduce the MPI Sessions API and its implications, and will then discuss options for extensions, which could provide a truly dynamic and malleable MPI.

### 3.17 Dynamic Tuning of HPC Applications – MERIC


*Ondrej Vysocky (VSB-Technical University of Ostrava, CZ)*

License  Creative Commons BY 4.0 International license  
© Ondrej Vysocky

This talk presents the MERIC tool for dynamic tuning of HPC hardware or runtime systems while running parallel application. The goal is to minimize the energy to solution with user-defined impact on application performance. Additionally, we also discuss the potential of tuning the hardware under power-cap which not only opens opportunity for energy savings but also for performance improvements. As the tool is continuously used to evaluate potential of energy savings for different applications under H2020 and EuroHPC projects it is being extended with new features. This includes support for new hardware, such as GPUs or new CPU architectures as presented. As the approach we use can perform dynamic tuning at relatively high rate, it is suitable for dynamic resource management.

### 3.18 The Price Performance of Performance Models

*Felix Wolf (TU Darmstadt, DE)*

License  Creative Commons BY 4.0 International license  
© Felix Wolf

To understand the scaling behavior of HPC applications, developers often use performance models. A performance model is a formula that expresses a key performance metric, such as runtime, as a function of one or more execution parameters, such as core count and input size. Performance models offer quick insights on a very high level of abstraction, including predictions of future behavior. In view of the complexity of today's applications, which often combine several sophisticated algorithms, creating performance models manually is extremely laborious. Empirical performance modeling, the process of learning such models from performance data, offers a convenient alternative, but comes with its own set of challenges. The two most prominent ones are noise and the cost of the experiments needed to generate the underlying data. In this talk, we will review the state of the art in empirical performance modeling and investigate how we can employ machine learning and other strategies to improve the quality and lower the cost of the resulting models.

## 4 Outlook: Techniques for Malleability-Enabled Machines

This section is the outcome of a joint effort of the seminar participants to provide a summary of the results of the fruitful discussions during the seminar. The section covers all the aspects of introducing dynamic resource management for malleable applications on HPC systems.

### 4.1 Resource Management

The dynamic behavior of malleable applications introduces new requirements to the Resource Management (RM) infrastructure. Current RM implementations are static, as a consequence of our traditional workloads being static themselves. This aspect of our workloads has been changing over the years. The amount of dynamism varies, and malleability support has become important for a subset of current highly dynamic workloads.

Many of these workloads are developed with MPI, with an increasing number of them using emerging programming models. Because of this, RM malleability support should be programming model agnostic. This can be achieved by the use of PMIx for RM, runtime system, tools and application interactions.

New features are more likely to be accepted by emerging RM infrastructures. Well established workload managers, such as Slurm, are static in design, and large scope changes to it may be undesirable; therefore, we expect malleability features to be more easily embraced by emerging workload managers, such as Flux.

Job and application level malleability specifications are necessary. For example, instead of specific resource specifications, valid ranges should be specified. These can be more than just compute resources, and can include, for example, memory and energy requirements.

The dynamic behavior of these workloads, together with the increasing parallelism of nodes (e.g., provided by GPUs), lowers the likelihood that hardware resources will be properly utilized by a single application. Co-scheduling is a way to improve node-local resource utilization, by allowing applications from multiple jobs to share nodes.

These changes will likely require updates to accounting mechanisms. The resources used by users need to be tracked in time, instead of being trivially determined on job starts. Instead of node-hours, if co-scheduling is enabled, slices of nodes will need to be tracked. Furthermore, since there may be detrimental effects of co-scheduling and malleability, accounting incentives need to be provided. Finally, node-slice counts will vary in time, if malleability is enabled.

Our systems will need to have configuration options that enable the definition of new system policies. For example, a system should be able to prioritize new job starts over resource allocation expansions via malleability, and vice versa. It should also be possible to prioritize based on a new job taxonomy that includes types of jobs that are malleable.

## 4.2 Programming Models: MPI and Beyond

To support malleability, applications themselves need to become malleable and for that need matching mechanisms in the parallel programming systems and APIs. This discussion can be split into four main categories: cross-node support based on MPI, on-node support based on shared memory models, coarse-grain support for workflows and models for heterogeneous systems. Additionally, combinations of these aspects will have to be considered and will add additional complexity and dependencies in the design of complete system-wide programming abstractions.

### 4.2.1 Support for on-node malleability and the benefits of tasking

Intra-node programming models require less effort to support malleability as there is no need for data redistribution in the event of a compute resources change. For example, the OpenMP language constructs and runtime system already allows the use of different numbers of threads for different (executions of) parallel regions. The same is true for shared-memory applications using a task-based approach (like OmpSS, StarPU, OpenMP tasks) where the computation is described as a set of tasks and the dependencies between them. The actual execution is then left to the runtime system. Enabling malleability can be supported for all applications (without the need to change them) by extending the runtime system to work together with the resource manager. However, this is only useful for HPC systems which would allow more than one application to use the same node (co-scheduling). This is rare on

today's HPC production systems, which are optimized for high performance, due to the fear of too much interference between the different applications on the same node resulting in overall bad performance.

#### 4.2.2 Support for coarse-grained malleability in Workflows

Aside from malleability of single applications, the malleability in workflows is a major topic. In this case, it may be sufficient for applications (in the sense of workflow components) to be simply moldable, but it must be possible to shift resources dynamically between different elements of the workflow. For this to be successful, though, it is critical for the resource manager to understand the workflow and its control flow and to be able to take this (in its entirety) into account when scheduling individual elements of the workflow. For this, the needed interfaces need to be designed and possibly standardized, across different workflow and management systems. These interfaces could then be made available via system resource managers like SLURM and Flux.

#### 4.2.3 Malleability in heterogeneous systems

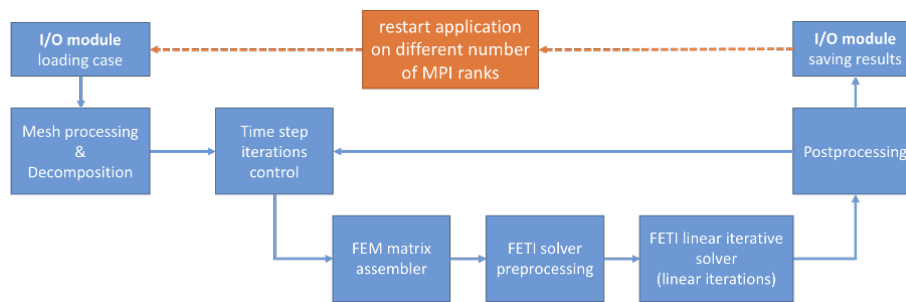
Heterogeneous systems (e.g., integrated systems with homogeneous nodes, but a range of accelerators in each, or modular systems with heterogeneous nodes, each hosting a different accelerator) pose additional challenges, but also offer additional opportunities for applying the concept of malleability. On one side, they add additional resource components and turn scheduling into a multi-objective problem, but on the other side this provides new flexibilities, especially when combined with more fine-grained workload and task descriptions that can be mapped to resources in a flexible manner. As also already discussed wrt. on-node malleability, it will likely lead to the need to support co-scheduling on nodes in order to allow the resource manager to tap the individual resource types on a node independently. Further, aspects like granularity, resource availability, contention and location have to be taken into account. A special aspect is added to the problem when also considering power and energy limitations, especially if power is limited to the point that not all accelerators (or more general, all resources) on a node can be powered at the same time and tradeoffs have to be made to shift computational power between heterogeneous resources.

### 4.3 Unified Monitoring

Malleable and adaptive systems need to collect a variety of performance data to allow allocation and scheduling decisions to be made during runtime. Our premise is that this information could also be used by application performance and visualization tools to provide useful information to programmers as to what changes in their program might result in more efficient uses of these adaptive systems.

We see a research agenda that could proceed in the following steps:

1. Identify what application and system performance information is currently being collected by adaptive systems and runtimes to enable malleability.
2. Design an interface so that tools can access and monitor this information.
3. Develop abstractions and mechanisms to deliver this information in forms used by current performance tools, including tracing, sampling, statistical summarization.
4. Study how to combine this adaptive system performance data with data provided by traditional performance tools.



5. Study how the techniques used by performance tools for more static applications need to be extended or changed to provide an understanding of programs running on an adaptive system.
6. Study the security and privacy issues associated with this new source of performance information and develop strategies for avoiding information leakage.

#### 4.4 Turning Applications into Malleable Applications

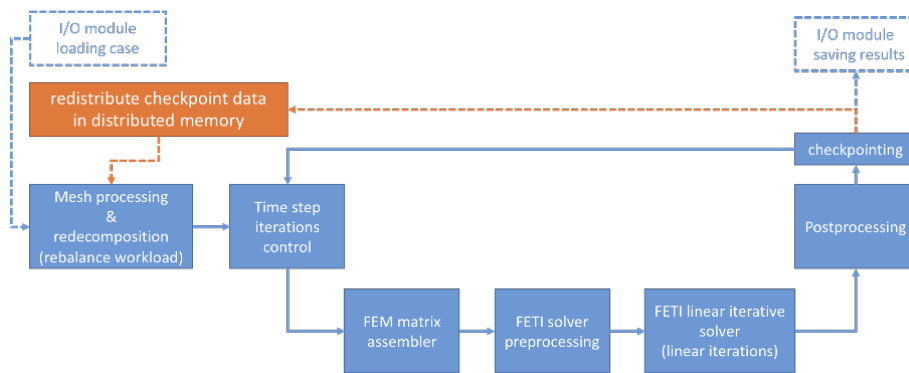
There might be more or less suitable applications for converting them into malleable. If we want to identify applications which can be in “reasonable” time and effort converted into malleable we should identify whether following capabilities are already implemented: (1) checkpoint and restart and (2) load balancing (in the best case the dynamic one). In particular, if an application is able to checkpoint at  $N$  MPI ranks and restart at  $M$  ranks it means that it already contains the functionality related to redistribution of its data structures. In the simplest scenario this can be supported through a shared filesystem that is used to store the checkpointed data.

An example of this simple scenario is shown in the figure below on an ESPRESO FEM application developed at IT4Innovations NSC in Czech Republic (<http://espresso.it4i.cz>). This application is based on a domain decomposition method and uses several variations of parallel FETI linear solvers.

The simple scenario then can be converted into a more advanced one which does not rely on a shared filesystem. This step contains extra work that must be done during the conversion into a malleable application. In case of the ESPRESO the I/O module creates the representation of a checkpoint file which contains the current state of a simulation in a distributed memory before it is saved. This file then can be directly redistributed using MPI communication.

The load balancing is then performed by calling the ParMETIS which repartition the mesh into a new number of domains. After that the mesh is redistributed and transferred into new MPI ranks. Based on mesh redistribution the results data are redistributed, accordingly.

There is also an overhead associated with mesh redistribution as all the FEM related objects generated for particular decomposition have to be rebuilt, i.e. all the FEM matrices have to be assembled again. This in general has to be considered for all applications.



## 4.5 Incentives to Help in Dynamic Resource Management

Dynamic resource management can be improved using application knowledge. Applications can, non-intrusively, provide information in their jobscript specifications. Moldable applications would specify a set of operation points and not ask for a single specific configuration. Malleable applications would specify their properties like “can shrink”, “can grow”, probably including limits and costs. In return applications that provide such information can get higher priorities in the scheduling queues (because they help to utilize the machine) and a chance to finish earlier.

Annotations about processing phases are a more intrusive way to help the resource manager. The knowledge about job phases can be used to learn and predict resource usage and provide just the right amount of resources to execute the current phase. The benefit for the application appears when the accounting excludes the unused resources, regardless of whether or not the resources are used elsewhere. With co-scheduling the phase annotations can be used to improve utilization of heterogeneous resources, e.g. to hand over the otherwise unused GPUs to a different application.

Malleable Applications are doing some internal resource management and might know about phases when less or more resources are best to achieve good efficiency. Those applications want to return some of their resources to the resource manager and also want to have a guarantee to get the resources back at a later point in time. This can be achieved by leasing the resources for a certain amount of time. If the resource manager agrees to take the resources, then it would guarantee their availability after the deadline has expired and would not charge the application for the leased resources. Such a mechanism can be used to establish a spot market where the central resource manager as well as the applications negotiate resource access and its costs.

## 4.6 What is the Potential of Malleable Applications?

The malleability support will obviously improve the total system throughput, however it is still not clear how much we can ideally gain. Assessing the potential improvement or the theoretical upper limit will motivate supercomputing centers, HPC research community, as well as the industry toward the malleability support. Throughout the discussion in the meeting, as a community, we concluded that we should quantify the potential gain for an idealistic scenario, by using a real system job trace collected at a supercomputing center, such as LRZ.



First, to conduct this estimation, we need a metric like scale-time product – here we define it as the integral of the number of nodes over time. For instance, if we can reduce the scale-time product by one  $X$ th for all the jobs by supporting malleability, we can potentially gain  $X$  times system throughput improvement (of course, this is an ideal case, i.e., only if the job scheduler can fully utilize the extra resources brought by the malleability support).

As a next step, we should know how much we can potentially reduce the scale-time product for each application in the job trace, and then we can quantify the potential system-level throughput improvement by just integrating them (again, this is an ideal case). However, as it is extremely difficult to estimate it for all the jobs, we should introduce some assumptions, conditions, and so forth.

One option for this is focusing on few large-scale and time-consuming jobs, investigating the malleability opportunities for them, and estimating the potential throughput improvement at system level by the simple calculation stated above. Another option for this is a rather statistical approach, e.g., assuming a distribution function that describes the relationship between the scale-time product and the reduction rate of it by the malleability support, and calculates the potential gain based on it.

Then, the next step should be more realistic, such as estimating the gain for some different scheduling algorithms, which is going to be simulation-based experiments. In this evaluation, we will need the number of nodes as a function of time for each job. In the end, this can be extended to cover other sophisticated resource managements such as co-scheduling or power management.

## 4.7 Converged Computing

Malleability would enable the integration of interactive workloads with typical HPC jobs on the system. The different characteristics of interactive workloads have to be taken into account, i.e., they come in bursts, they have to be executed immediately, and the length and resource requirements are not specified in the form of a batch script.

The opportunities for the HPC center are that the interactive workloads are excellent candidates for increasing the overall system utilization. Furthermore, new application domains become available for the centers, while these domains would profit from computation on powerful HPC systems.

Three use cases were introduced in the seminar. Argonne is working on integrating HPC into the continuum of resources for edge based IoT applications. Another is to bring entire scientific workflows that combine microservice based components with HPC jobs. At LLNL the combination of Kubernetes resource management and Flux is investigated. TUM is researching automatic distribution of function invocations in serverless computing to the heterogeneous computing continuum of HPC, Cloud, Edge, and IoT devices. Due to the limitations on production HPC systems, the use of webassembly for isolation and optimization for heterogeneous nodes of an HPC system is investigated at TUM.

The malleability of HPC jobs and the dynamic resource management can be used to provide resources even on an HPC system on demand for the interactive tasks, without overprovisioning in low demand periods. The approach taken by different groups is to integrate the resource management of a Spark cluster or Kubernetes with the resource management on the machine, providing additional resources to the interactive cluster if required. These resources can be idle nodes or be taken from malleable applications. In case the interactive cluster shrinks, the nodes are given back to the HPC applications.

Malleability would also be beneficial in case the HPC system would be connected to HPC in the Cloud for overflow computation. Application scaling is a major advantage of Cloud systems, e.g., auto-scaling is used to adapt cloud based services to a changing workload. If malleable HPC applications are provided already on the HPC system these could significantly benefit from the practically unlimited resources in the cloud. On-demand allocation of resources depending on the status of the application matches the inherent strength of the cloud. In the cloud context, the incentive to make your applications malleable would even be much higher than on HPC systems as resource usage is paid in the pay-per-use model.

## Participants

- Eishi Arima  
TU München, DE
- Eduardo César  
Autonomous University of  
Barcelona, ES
- Isaías Alberto Comprés Ureña  
TU München, DE
- Michael Gerndt  
TU München, DE
- Jophin John  
TU München, DE
- Matthias Maiterth  
TU München, DE
- Barton P. Miller  
University of Wisconsin-  
Madison, US
- Bernd Mohr  
Jülich Supercomputing  
Centre, DE
- Frank Mueller  
North Carolina State University –  
Raleigh, US
- Santiago Narvaez Rivas  
TU München, DE
- Mirko Rahn  
Fraunhofer ITWM –  
Kaiserslautern, DE
- Lubomir Riha  
VSB-Technical University of  
Ostrava, CZ
- Martin Schulz  
TU München, DE
- Anna Sikora  
Autonomous University of  
Barcelona, ES
- Ondrej Vysocky  
VSB-Technical University of  
Ostrava, CZ
- Felix Wolf  
TU Darmstadt, DE



## Remote Participants

- Dong Ahn  
LLNL – Livermore, US
- Andrea Bartolini  
University of Bologna, IT
- Pete Beckman  
Argonne National Laboratory –  
Lemont, US
- Mohak Chadha  
TU München, DE
- Julita Corbalan  
Barcelona Supercomputing  
Center, ES
- Balazs Gerofi  
RIKEN – Kobe, JP
- Toshihiro Hanawa  
University of Tokyo, JP
- Shantenu Jha  
Rutgers University –  
Piscataway, US
- Rashawn Knapp  
Intel – Hillsboro, US
- Masaaki Kondo  
Keio University – Yokohama, JP
- Daniel John Milroy  
LLNL – Livermore, US
- Tapasya Patki  
LLNL – Livermore, US
- Barry L. Rountree  
LLNL – Livermore, US
- Roxana Rusitoru  
Arm – Cambridge, GB
- Sakamoto Ryuichi  
Tokyo Institute of Technology, JP
- Wolfgang Schröder-Preikschat  
Universität Erlangen-  
Nürnberg, DE