# 25th APART Anniversary Meeting

**Monday 12 February 2024 - Thursday 15 February 2024**

**University Center Obergurgl**

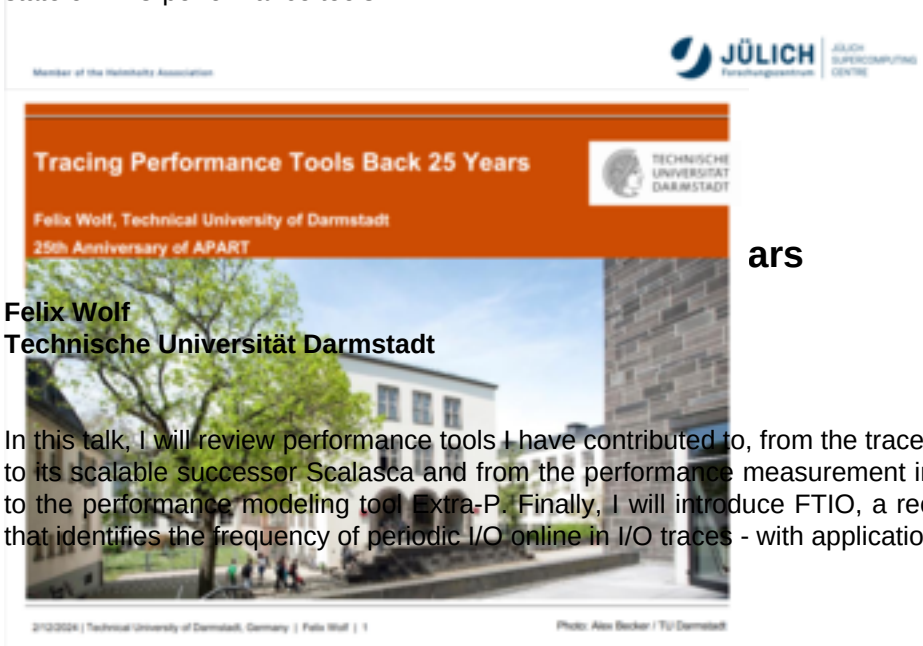
# Program

# Monday, February 12, 2024

## 37 years of Performance Tools Development: Success Stories and Failures

**Bernd Mohr**
**Jülich Supercomputing Centre**


I created my first performance tool for parallel systems as part of my master thesis at the Friedrich-Alexander University Erlangen-Nuremberg, Bavaria in 1987. Since then, I never changed my research topic and was involved in the development of many HPC performance tools: TDL/POET, ZM4, SIMPLE, TAU, Vampir, EPILOG, KOJAK, Score-P, and Scalasca. The talk will highlight successes and failures of the different tools and will provide a personal assessment of the state of HPC performance tools.




## ars

**Felix Wolf**
**Technische Universität Darmstadt**

In this talk, I will review performance tools I have contributed to, from the trace-analysis tool KOJAK to its scalable successor Scalasca and from the performance measurement infrastructure Score-P to the performance modeling tool Extra-P. Finally, I will introduce FTIO, a recently developed tool that identifies the frequency of periodic I/O online in I/O traces - with applications in I/O scheduling.

[Video recording] [Slides]

## Lessons learnt from 15+ years of VI-HPS & POP CoE training

**Brian Wylie**
**Jülich Supercomputing Centre**

Over more than fifteen years, an international consortium of tools developers have worked closely together in the Virtual Institute High Productivity Supercomputing (VI-HPS), collaborating on tools development and integration, latterly augmented by the Performance Optimisation and Productivity Centre of Excellence (POP CoE) which created a universal parallel performance assessment and improvement methodology focused on application execution efficiency and scalability. This was complemented with associated training and coaching of application developers (often in teams) for productive use of the portfolio of tools, which will be reviewed and lessons learnt discussed.


[Video recording] [Slides]

# Tuesday, February 13, 2024

## Binary Code Patching: An Ancient Art Refined for the 21st Century

**Barton Miller**
**University of Wisconsin-Madison**

Patching binary code dates back to some of the earliest computer systems. Binary code patching allows access to a program without having access to the source code, obviating the need to recompile, re-link, and, in the dynamic case, re-execute.

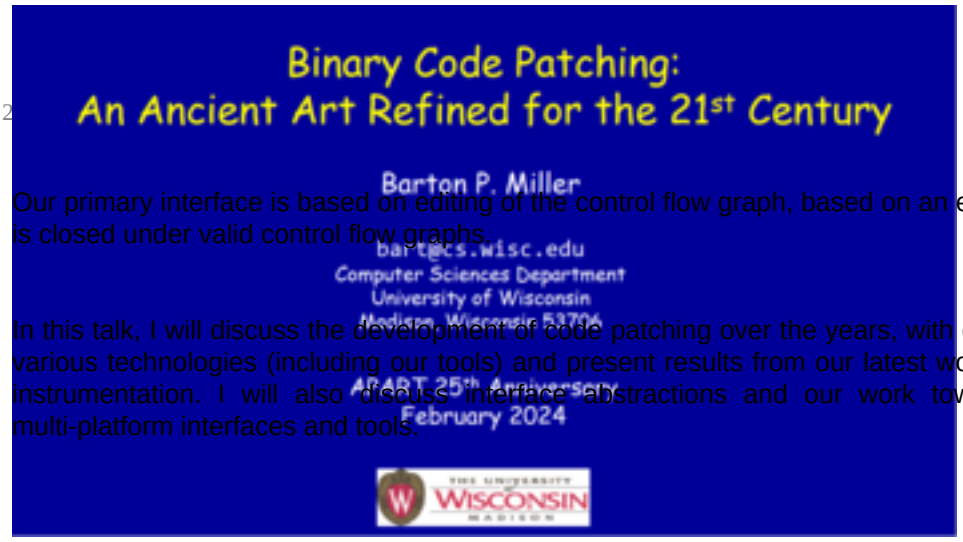
In the early days, it was a bold technique used by serious programmers to avoid the long recompile/reassemble and link steps. Code patching required an intimate knowledge of the instruction set and its binary representation. Great advances have been made in simplifying the use of code patching, making it less error prone and more flexible. "Binary rewriters" were a great advance in the technology for modifying a binary before its execution. Early tools, such as OM, EEL, and Vulcan, enabled the building of tools for tracing, simulation, testing, and sandboxing.

Moving beyond static patching, we developed "dynamic instrumentation", the ability to patch code into a running program. Dynamic instrumentation provided the ability to adapt the code to the immediate need, dynamically control overhead costs. We applied this technology to both user programs and operating system kernels producing the Dyninst and Kerninst tool kits. This technology formed the foundation of the Paradyn Performance Tools.

Dynamic code patching continued to get more aggressive. We developed "self-propelled instrumentation", which inserts instrumentation code that propagates itself along the program's control flow as the program executes. At its best, this technique can provide very low overhead, detailed instrumentation in support of fault isolation and identification of intermittent performance issues.

More recently, we have addressed a wide variety of issues related to binary code patching including analyzing and patching defensive and obfuscated malware, parallelizing the binary code parsing process to quickly patch huge (GB+) binaries, and efficient analysis and instrumentation of GPU binaries.

Key to both static and dynamic patching are the interfaces. There is a difficult balance between providing an interface that abstracts the details of the code, often using control- and data-flow graphs and instruction categories, and an interface that exposes the details of the instruction set.




Our primary interface is based on editing of the control flow graph, based on an editing algebra that is closed under valid control flow graphs.

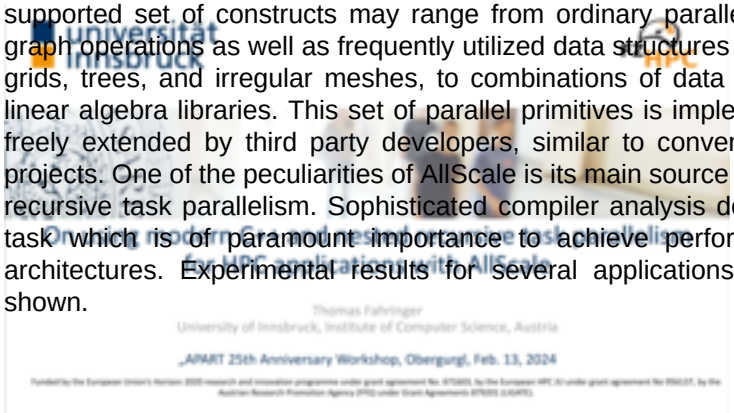
In this talk, I will discuss the development of code patching over the years, with examples from the various technologies (including our tools) and present results from our latest work in selfpropelled instrumentation. I will also discuss interface abstractions and our work towards the goal of multi-platform interfaces and tools.

[Video recording] [Slides]

## On using modern C++ and nested recursive task parallelism for HPC applications with AllScale

**Thomas Fahringer**
**University of Innsbruck**

On using modern C++ and nested recursive task parallelism for HPC applications with AllScale Contemporary parallel programming approaches often rely on well-established parallel libraries and/or language extensions to address specific HW resources that can lead to mixed parallel programming paradigms. In contrast to these approaches, AllScale proposes a C++ template-based approach to ease the development of scalable and efficient general-purpose parallel applications. Applications utilize a pool of parallel primitives and data structures for building solutions to their domain-specific problems. HPC experts who provision high level, generic operators and data structures for common use cases, design these parallel primitives. The supported set of constructs may range from ordinary parallel loops, over stencil and distributed graph operations as well as frequently utilized data structures including (adaptive) multidimensional grids, trees, and irregular meshes, to combinations of data structures and operations like entire linear algebra libraries. This set of parallel primitives is implemented using pure C++ and may be freely extended by third party developers, similar to conventional libraries in C++ development projects. One of the peculiarities of AllScale is its main source of parallelism that is based on nested recursive task parallelism. Sophisticated compiler analysis determines the data needed for every task which is of paramount importance to achieve performance across a variety of parallel architectures. Experimental results for several applications implemented with AllScale will be shown.




[Video recording] [Slides]

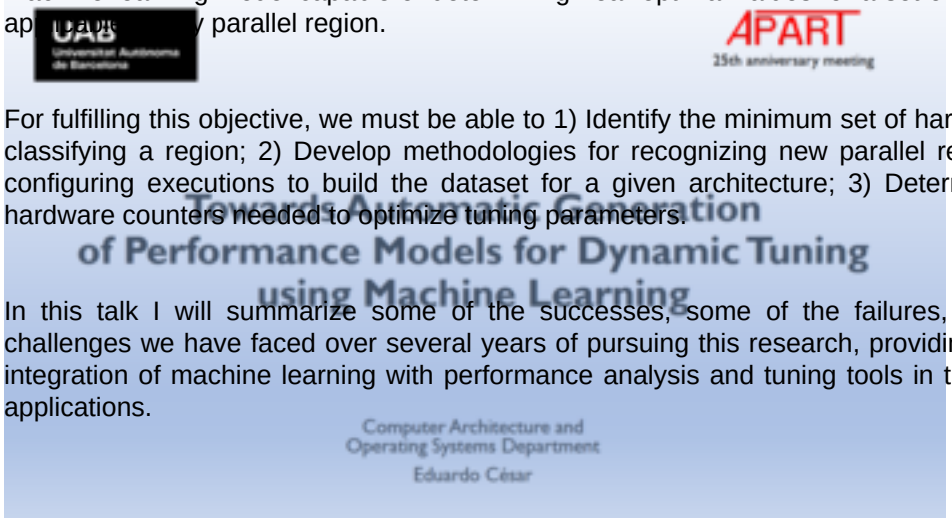
## Towards Automatic Generation of Performance Models for Dynamic Tuning using Machine Learning

**Eduardo Cesar**
**Universitat Autonoma de Barcelona**

Incorporating machine learning into automatic performance analysis and tuning tools is a promising approach for addressing the increasing heterogeneity of current High-Performance Computing

(HPC) applications. Our fundamental hypothesis posits that hardware performance counters effectively characterize parallel regions on multicore CPUs and GPUs.

We propose creating a dataset with the values of the most relevant hardware counters obtained from the execution of representative regions. This dataset will then be utilized to train a machine-learning model capable of determining near-optimal values for a set of tuning parameters ap[illegible] parallel region.




For fulfilling this objective, we must be able to 1) Identify the minimum set of hardware counters for classifying a region; 2) Develop methodologies for recognizing new parallel region patterns and configuring executions to build the dataset for a given architecture; 3) Determine the essential hardware counters needed to optimize tuning parameters.

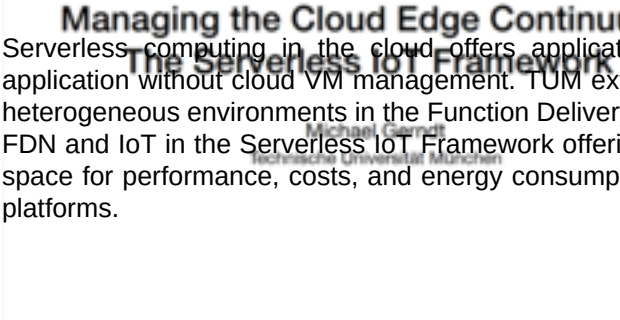
In this talk I will summarize some of the successes, some of the failures, and some of the challenges we have faced over several years of pursuing this research, providing insights into the integration of machine learning with performance analysis and tuning tools in the context of HPC applications.

[Video recording] [Slides]

# Wednesday, February 14, 2024

## Managing the Cloud Edge Continuum: The Serverless IoT

**Michael Gerndt**
**Technical University of Munich**




Serverless computing in the cloud offers application developers to concentrate on the cloud application without cloud VM management. TUM extended the concept of serverless computing to heterogeneous environments in the Function Delivery Network (FDN). Ongoing works, integrate the FDN and IoT in the Serverless IoT Framework offering function invocation optimization in time and space for performance, costs, and energy consumption across IoT devices, edge and cloud FaaS platforms.

[Video recording] [Slides]

## MPIWasm: Executing WebAssembly on HPC Systems

**Mohak Chadha**
**Technical University of Munich**

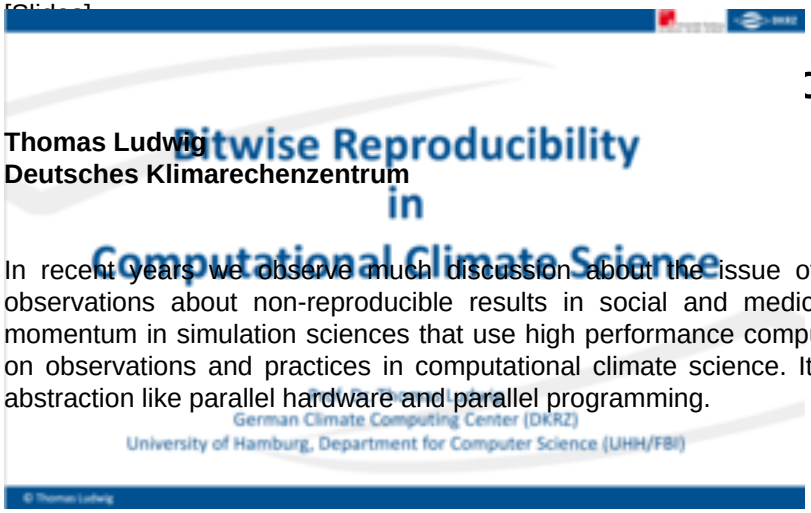
MPIWasm is a Wasm runtime that enables the execution of applications that utilize the Message Passing Interface (MPI) standard on HPC systems. It builds on Wasmer and supports the execution of MPI-based HPC application modules on both x86_64 and aarch64 processor architectures. To facilitate its adoption and suitability in HPC environments, it supports the high-performance execution of HPC application modules and has low overhead for MPI calls throughzero-copy memory operations. The former is achieved by leveraging the LLVM compiler for translating Wasm instructions to native machine code AoT, while the latter is achieved by transparently translating between the host and the Wasm module's linear memory address space. In addition, MPIWasm supports high-performance networking interconnects such as Infiniband that are used by MPI libraries for inter-rank communication by linking against the host MPI library at runtime and providing a translation layer between the Wasm modules and the host library. Our experiments on a production HPC system demonstrate that MPIWasm delivers competitive native application performance.

[Slides]

## Bitwise Reproducibility in Computational Climate Science

**Thomas Ludwig**
**Deutsches Klimarechenzentrum**

In recent years we observe much discussion about the issue of reproducibility. Triggered by observations about non-reproducible results in social and medical sciences the topic gained momentum in simulation sciences that use high performance computing. The talk will concentrate on observations and practices in computational climate science. It will focus on lower levels ob abstraction like parallel hardware and parallel programming.

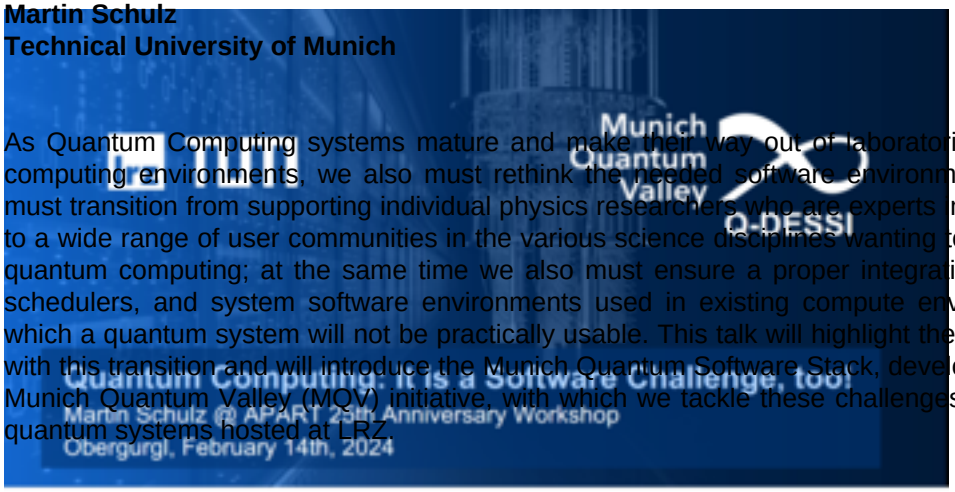



[Video recording] [Slides]

## Quantum Computing: It is a Software Challenge, too!

**Martin Schulz**
**Technical University of Munich**

As Quantum Computing systems mature and make their way out of laboratories into production computing environments, we also must rethink the needed software environments. For one, we must transition from supporting individual physics researchers who are experts in quantum science to a wide range of user communities in the various science disciplines wanting to use the power of quantum computing; at the same time we also must ensure a proper integration into workflows, schedulers, and system software environments used in existing compute environments without which a quantum system will not be practically usable. This talk will highlight the issues associated with this transition and will introduce the Munich Quantum Software Stack, developed as part of the Munich Quantum Valley (MQV) initiative, with which we tackle these challenges for the upcoming quantum systems hosted at LRZ.



[Video recording] [Slides]

# Thursday, February 15, 2024

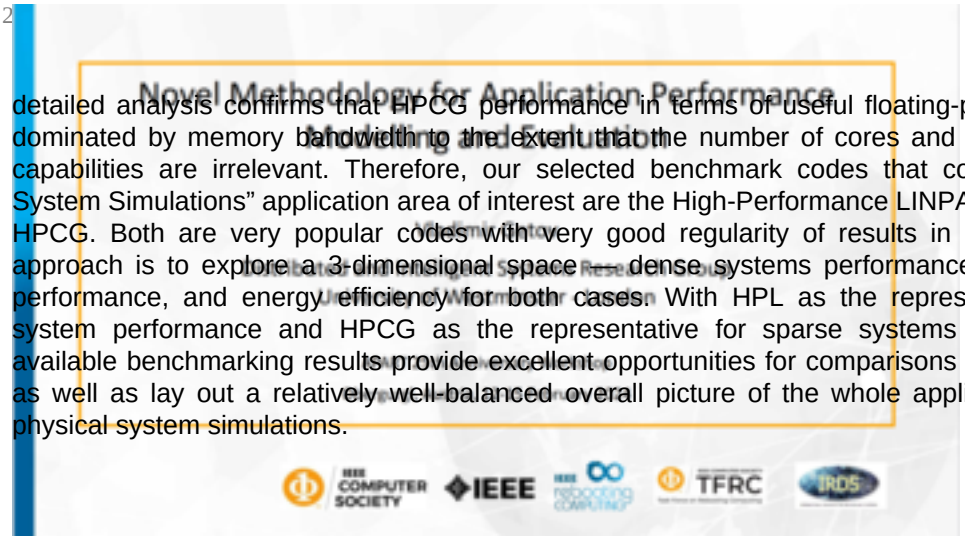
## Novel Methodology for Application Performance Modelling and Evaluation

**Vladimir Getov**
**University of Westminster**

Computer simulation of physical real-world phenomena emerged with the invention of electronic digital computing and has been increasingly adopted as one of the most successful modern methods for scientific discovery. Arguably, the main reasons for this success have been the rapid development of novel computer technologies that has led to the creation of powerful supercomputers, large distributed systems, high-performance computing frameworks with access to huge data sets, and high throughput communications. In addition, unique and sophisticated scientific instruments and facilities, such as giant electronic microscopes, nuclear physics accelerators, or sophisticated equipment for medical imaging are becoming integral parts of those complex computing infrastructures. Subsequently, the term ‘e-science’ was quickly embraced by the professional community to capture these new revolutionary methods for scientific discovery via computer simulations of physical systems. The relevant application codes are typically based on finite-element algorithms, while the computations constitute heavy workloads that conventionally are dominated by floating-point arithmetic. Examples include application areas such as climate modeling, plasma physics (fusion), medical imaging, fluid flow, and thermo-evolution.

Over the years, most of the relevant benchmarking projects have covered predominantly dense physical system simulations, in which high computational intensity carries over when parallel implementations are built to solve bigger problems faster. Since emphasis was on dense problems, this approach resulted in systems with increasing computational performance and was the presumption behind the introduction of the very popular semi-annual Top 500 rankings of supercomputers. However, in the last 10-15 years many new applications with very high economic potential have emerged — such as big data analytics, machine learning, real-time feature recognition, recommendation systems, and even physical simulations — that feature irregular or dynamic solution grids. These applications spend much more of their computation in non-floating-point operations such as address computations and comparisons, with addresses that are no longer very regular or cache-friendly. The computational intensity of such programs is far less than for dense kernels, and the result is that for many real codes today, even those in traditional scientific cases, the efficiency of the floating-point units that have become the focal point of modern core architectures has dropped from the >90% to <5%. This emergence of applications with data-intensive characteristics — e.g. with execution times dominated by data access and data movement — has been recognized recently as the “3rd Locality Wall” for advances in computer architecture.

To highlight the inefficiencies described above, and to identify architectures which may be more efficient, a new benchmark called HPCG (High Performance Conjugate Gradient) was introduced several years ago. HPCG also solves Ax=B problems, but where A is a very sparse matrix so that, on evaluated systems, floating-point efficiency mirrors that seen in full scientific codes. Recent




detailed analysis confirms that HPCG performance in terms of useful floating-point operations is dominated by memory bandwidth to the extent that the number of cores and their floating-point capabilities are irrelevant. Therefore, our selected benchmark codes that cover the “Physical System Simulations” application area of interest are the High-Performance LINPACK (HPL) and the HPCG. Both are very popular codes with very good regularity of results in recent years. Our approach is to explore a 3-dimensional space — dense systems performance, sparse systems performance, and energy efficiency for both cases. With HPL as the representative of dense system performance and HPCG as the representative for sparse systems performance, the available benchmarking results provide excellent opportunities for comparisons and interpretation, as well as lay out a relatively well-balanced overall picture of the whole application domain for physical system simulations.

[Video recording] [Slides]

## SUIT: Secure Undervolting with Instruction Traps

**Frank Mueller**
**North Carolina State University**

SUIT presents a novel hardware-software co-design to reduce the safety margin substantially without compromising reliability or security. By enhancing hardware to trap on infrequent instructions consuming high power, we develop a novel OS/runtime mechanism transparent to the user that dynamically switches power regimes in response to instruction needs. We show that this technique results in energy savings at minimal performance cost on average and occasionally even performance improvements.

[Slides]

## Celerity - Distributed-memory Accelerator Programming Made Easier

**Philipp Gschwandtner**
**University of Innsbruck**

While domain-specific HPC software packages continue to thrive and are vital to many scientific communities, a general purpose high-productivity GPU cluster programming model that facilitates experimentation for non-experts remains elusive

Celerity, a combined API and task-based runtime system for programming distributed-memory GPU-based HPC hardware platforms, seeks to provide the means to scale C++ applications to distributed-memory accelerator clusters with relative ease by leveraging the SYCL domain-specific embedded language. By providing information about the logical and spatial buffer access behavior of kernels, users enable the Celerity runtime system to automatically split work across multiple GPUs. Encoded in an execution graph, correctness of the distributed program is ensured by tracking kernel data dependencies and issuing data transfers when required. This flexible design facilitates the effective utilization of hardware resources without the need for manual scheduling.

To further increase productivity, Celerity provides an easy-to-use API intended to offer frequently encountered higher-order parallel programming patterns such as reductions, stencils and parallel I/O. A fully functional implementation is developed and maintained at the University of Innsbruck, currently used for porting benchmarks and applications. This talk will present the Celerity concept, current results and ongoing research.

[Slides]