Workshop on Asynchronous Many-Task Systems 2024

Venue:
Student Union Building, Room 169
University of Tennessee, Knoxville

February 14-16, 2024

This workshop is sponsored by

- University of Tennessee, Knoxville (UTK)
- Louisiana State University (LSU)
- Oak Ridge National Laboratory (ORNL)
- Tactical Computing Labs

Abstract

As our compute capacity grows, science simulations are not only becoming bigger, but more complex. Simulations are carried out at multiple scales and using multiple kinds of physics at once. Boundaries are irregular, grids are irregular, computational domains can be dynamic and complex. In such scenarios, the ideal way to parallelize often cannot be statically determined. At the same time, hardware is becoming more heterogeneous and difficult to program. Increasingly, scientists are turning to asynchronous, dynamic parallelism in order to make the best use of increasingly challenging hardware. As a result, numerous frameworks, platforms, and specialized languages have sprung up to answer this need.

The objectives of this workshop are to bring together experts in asynchronous many-task frameworks, developers of science codes, performance experts, and hardware vendors to discuss the state-of-the-art techniques needed to program, analyze, benchmark, and profile these codes to achieve maximum performance possible from modern machines. This workshop will promote a dialogue between these communities, and help identify challenges and opportunities for advancement in all the disciplines they represent.

Organizing committee

- Patrick Diehl, Louisiana State University (USA)
- Pedro Valero-Lara, Oak Ridge National Laboratory (USA)
- George Bosilca, NVIDIA (USA)
- Joseph Schuchart, University of Tennessee, Knoxville (USA)

Scientific committee

- Alex Aiken, Stanford (USA)
- Erwin Laure, Max Planck Computing & Data Facility (Germany)
- Christoph Junghans, Los Alamos National Laboratory (USA)
- Bryce Adelstein Lelbach, NVIDIA (USA)
- Laxmikant V. Kale, University of Illinois at Urbana-Champaign (USA)
- Brad Chamberlain, HPE and University of Washington (USA)

Technical program chair

- Patrick Diehl, Louisiana State University (USA)
- Pedro Valero-Lara, Oak Ridge National Laboratory (USA)

- George Bosilca, NVIDIA (USA)
- Joseph Schuchart, University of Tennessee, Knoxville (USA)

Technical program

- Brad Richardson, Sourcery Institute (USA)
- Kevin Huck, University of Oregon (USA)
- Dirk Pflüger, University of Stuttgart (Germany)
- Metin H. Aktulga, Michigan State University (USA)
- Huda Ibeid, Intel
- Dirk Pleiter, KTH Royal Institute of Technology (Sweden)
- Didem Unat, Koç University (Turkey)
- Keita Teranishi, Sandia National Laboratories (USA)
- Gregor Daiß, University of Stuttgart (Germany)
- Najoude Nader, Louisiana State University (USA)
- Weile Wei, Lawrence Berkeley National Laboratory (USA)
- Jeff Hammond, NVIDIA (Finland)
- Hartmut Kaiser, Louisiana State University (USA)
- J. Ram Ramanujam, Louisiana State University (USA)
- Steven R. Brandt, Louisiana State University (USA)
- Narasinga Rao Miniskar, Oak Ridge National Laboratory (USA)
- Markus Rampp, Max Planck Computing and Data Facility (Germany)
- Sumathi Lakshmiranganatha, Los Alamos National Laboratory (USA)
- Nikunj Gupta, Amazon (USA)
- Jonas Posner, University of Kassel (Germany)

Logistics

- Parsa Aminim, Halpern-Wight Inc. (USA)
- Joseph Schuchart and Thomas Herault, University of Tennessee, Knoxville (USA)

Welcome Address

Greetings,
It gives me great pleasure to extend a warm welcome to all participants of the Workshop on Asynchronous Many-task Systems and Applications, scheduled from February 14 to 16, 2024. The event is being hosted by the Innovative Computing Lab at the University of Tennessee, Knoxville, on the UT campus located in Knoxville, Tennessee. I am delighted to announce our support and co-sponsorship of this workshop, alongside other contributors such as Tactical Computation Labs, LSU and ORNL.
I would like to express my appreciation for the exceptional efforts of the workshop organizers: the local organizers, Dr. Joseph Schuchart and Dr. George Bosilca, as well as the workshop chairs Dr. Patrick Diehl and Dr. Pedro Valero-Lara. The primary objective of this workshop is bringing together experts in asynchronous many-task frameworks, scientific code developers, performance engineering specialists, and hardware vendors and to create an environment where discussions and new ideas flourish. Our aim is to facilitate discussions on cutting-edge techniques necessary for the development, analysis, benchmarking, and profiling of task-based applications to achieve optimal performance on modern architectures. This workshop serves as a platform for fostering dialogue among these communities, identifying challenges, and exploring opportunities for advancement across various disciplines.
A special thanks goes to our distinguished keynote speakers: Dr. Sean Treichler from NVIDIA, Dr. Jeff Vetter from Oak Ridge National Laboratory, and Prof. Robert J. Harrison from Stony Brook University. Their expertise in the field promises valuable insights, and we are grateful for their willingness to share them with us. Finally, I extend my gratitude to all participants. I sincerely hope that you find enjoyment and benefit from the unique discussions and sessions planned over the next two and a half days.
With best wishes,
Jack Dongarra

Contents

Welcome Address **v**

Industrial talk **1**
Orienteering RISC-V for HPC (*Chris Taylor*) 1

Session Chairs **3**

Talks **5**
Deep Codesign in the Post-Exascale Computing Era (*Jeffrey Vetter*) . . . 5
Optimizing Parallel System Efficiency: Dynamic Task Graph Adaptation with Recursive Tasks (*Nathalie Furmento*) 6
On scheduling languages for tasking execution models (*Jose M Monsalve Diaz*) 7
Evolving APGAS Programs: Automatic and Transparent Resource Adjustments at Runtime (*Jonas Posner*) 8
Evaluating PaRSEC for Advanced Matrix Computations in Climate and Weather Prediction (*Qinglei Cao*) 9
Experiences Porting Distributed Applications to Asynchronous Tasks: A Multidimensional FFT Case-study (*Alexander Strack*) 10
DLA-Future: a task-based linear algebra library which provides a GPU-enabled distributed eigensolver. (*Raffaele Solcà*) 11
A Suite of Codes for Benchmarking and Testing Mutex-Based Parallel Systems (*Max Morris*) 12
Speaking Pygion: Experiences Writing an Exascale Single Particle Imaging Code (*Alex Aiken*) 13
Taskflow: A General-purpose Task-parallel Programming System (*Tsung-Wei Huang*) 14
Too Much Parallelism, and the Need for Compiler Support in Asynchronous Many-Task Systems (*Jean Treichler*) 15
Futures for dynamic dependencies – Parallelizing the H-LU Factorization (*Rüdiger Nather*) 16
Scalable Block-Sparse Matrix Multiplication Using Template Task Graphs (*Joseph Schuchart*) 17
A Peek into Multi-Vendor and Multi-device Heterogeneity and Portability of MatRIS using IRIS Task based Runtime (*Mohammad Alaul Haque Monil*) 18

An abstraction for distributed stencil computations using Charm++ (*Aditya Bhosale*) . . . 19
HPX in the HPC Cloud: Evaluating Overheads of Cloud resources for HPX/Kokkos using an astrophysics application (*Patrick Diehl*) . . . 20
Distributed Asynchronous Contact Mechanics with DARMA/vt (*Nicholas Morales*) 21
Rethinking Programming Paradigms in the QC-HPC Context (*Elaine Wong*) 22
MADNESS: A task-based application and runtime (*Robert Harrison*) . . . 23
Only Pay for What You Need: Registration-Based Queueing in Asynchronous Many-Task Systems (*Zane Fink*) . . . 24
Dynamic Tuning of Core Count to Maximize Performance in Object-based Runtime systems (*Kavitha Chandrasekar*) . . . 25
The Asynchronous Low-level Programming Interface (ALPI): Enabling Portable Task-Aware Libraries Across Different Runtime Systems (*Kevin Sala*) . . . 26
IRIS Reimagined: Advancements in Intelligent Runtime System for Task-Based Programming (*Narasinga Rao Miniskar*) . . . 27

Posters **29**
A Lightweight Communication Interface for Asynchronous Many-Task Systems (*Omri Mor*) . . . 29
MatRIS 1.0: A Portable abstraction of Math Library using IRIS Task based Runtime for Multi-device and Multi-Vendor Heterogeneity (*Mohammad Alaul Haque Monil*) . . . 29

Additional information **31**
Addresses . . . 31
Restaurants . . . 31

Author Index **33**

Industrial talk

Orienteering RISC-V for HPC

15^{th}

Chris Taylor
Tactical Computing Lab

RISC-V is making an inroads to commodity and HPC computing. This presentation provides an introduction to the current state of the RISC-V HPC environment. The presentation specifically introduces the RISC-V instruction set architecture, available hardware platforms, the state of HPC RISC-V software, venues to track RISC-V HPC related work, and additional information about the RISC-V foundation working groups, technical groups, and special interest groups.

Session Chairs

- Keynote 1, TBD
- Session 1 (Wed 10:30 – 12:20), TBD
- Session 2 (Wed 1:30 – 3:00), TBD
- Session 3 (Wed 3:30 – 4:30), TBD
- Keynote 2, TBD
- Session 3 (Thu 10:30 to 12:00), TBD
- Session 4 (Thu 1:30 to 3:30), TBD
- Keynote 3, TBD
- Session 5 (Fr 10:30 to 12:30), TBD

Talks

Note that blue names indicate virtual talks.

Deep Codesign in the Post-Exascale Computing Era

14^{th}
9:00 AM–10:00 AM

Jeffrey Vetter
Oak Ridge National Laboratory

DOE has just deployed its first Exascale system at ORNL, so now is an appropriate time to revisit our Exascale predictions from over a decade ago and think about post-Exascale. We are now seeing a Cambrian explosion of new technologies during this 'golden age of architectures,' making codesign of architectures with software and applications more critical than ever. In this talk, I will revisit the Exascale trajectory, survey post-Exascale technologies, and discuss their implications for both system design and software. As an example, I will describe Abisko, a new microelectronics codesign project, that focuses on designing a chiplet for analog spiking neural networks using novel neuromorphic materials like electrochemical random access memory.

14th
10:30 AM–11:00 AM

Optimizing Parallel System Efficiency: Dynamic Task Graph Adaptation with Recursive Tasks

Nathalie Furmento
INRIA

The efficiency of both heterogeneous and homogeneous parallel systems can be significantly enhanced by using task-based programming models. One such model, the Sequential Task Flow (STF) model, only allows task graphs with static tasks sizes. Indeed, while this model proves effective in managing task graphs efficiently, the task granularity must be chosen when submitting the graph.

However, finding the ideal task granularity to fully exploit a parallel system is a challenging problem.

Firstly, for heterogeneous systems, the optimal task size varies across different processing units. Secondly, even for homogeneous systems, the best granularity also depends on the idleness of the computing units, for example, either when many tasks are ready, or when there is a lack of parallelism.

To address these issues, we use an extension of StarPU's STF model, which allows tasks to transform into subgraphs at runtime – referred to as recursive tasks. We have enhanced recursive tasks, by introducing the concept of a "splitter", a tool positioned between task submission and scheduling. This tool makes just-in-time decisions to transform a task into a subgraph with the appropriate granularity.

We provide an early evaluation on homogeneous shared-memory systems and an ongoing work-in-progress for heterogeneous architectures. These first results demonstrate that the just-in-time adaptation of the task graph opens up new opportunities for increased performance.

On scheduling languages for tasking execution models

14^{th}
11:00 AM–11:30 AM

Jose M Monsalve Diaz
Argonne National Laboratory

Tasking execution models are valuable for achieving parallel, heterogeneous, and distributed computing performance. Thanks to the influence of dataflow execution models, tasking enables finer-grain control of the parallelism compared to other execution models such as SPMD. However, most tasking implementations combine the program execution model with its application programming interface into a single definition. More often than not, the API is implemented as part of a programming model that relies on von Neumann execution. Such is the case of the most popular tasking models such as OpenMP, OmpSs, Legion, etc.

This work describes the importance of separating roles between the scheduling language and the computation language. We demonstrate that these two roles are present in all execution models, with some being more explicit than others, but that an explicit distinction could enable further optimizations to be applied to tasking models. We then define a scheduling language and demonstrate how this language can be obtained from the OpenMP tasking execution model. Finally, we list open questions that will need to be answered in order to define an appropriate scheduling language for parallel, heterogeneous, and distributed program execution models.

14^{th}
11:30 AM–12:00 AM

Evolving APGAS Programs: Automatic and Transparent Resource Adjustments at Runtime

Jonas Posner
University of Kassel

In the rapidly evolving field of High-Performance Computing (HPC), the need for resource adaptivity is paramount, particularly in addressing the dynamic nature of irregular computational workloads. A key area of adaptivity lies within programming models, which typically offer limited support.

Fully adaptive programs are both malleable–capable of dynamically adjusting resources in response to external job scheduler requests–and evolving–autonomously deciding when and how to adjust resources, e.g., through automated decision-making. Previous adaptivity approaches typically relied on iterative workloads and required complex code modifications.

Asynchronous Many-Task (AMT) programming is emerging as a powerful alternative. In AMT, computations are split into fine-grained tasks, allowing transparent task relocation by the runtime system and unlocking significant potential for efficient adaptivity.

This work-in-progress proposes an extension to the existing AMT APGAS that recently incorporated malleability. Our extension adds evolving capabilities providing automatic and transparent resource adjustments to meet changing computational workloads at runtime. While our easy-to-use abstractions require only minimal code additions, adjustments such as process initialization and termination are managed automatically. Our extension is validated via a load-balancing library for irregular workloads.

We propose two heuristics for automatic computational load detection: one that uses CPU loads provided by the OS, and another that exploits detailed insights into task loads. We evaluate our approach using a novel synthetic benchmark that starts with a single task evolving into two irregular trees connected by a long sequential branch. Preliminary results are promising, indicating that the task-load-based heuristic outperforms the CPU-based one in responsiveness and effectiveness.

Evaluating PaRSEC for Advanced Matrix Computations in Climate and Weather Prediction

14^{th}
1:30 PM–2:00 PM

Qinglei Cao
Saint Louis University

Task-based runtime systems, characterized by their dynamic execution models and optimized resource management, are at the forefront of a computational revolution. They enable the development of more intricate and adaptable algorithms, essential in the field of computational science. This paper provides an in-depth exploration of the PaRSEC task-based runtime system, particularly focusing on its versatility in managing a variety of matrix computations. More specifically, we examine PaRSEC's role in enhancing the efficiency of processing dense, low-rank, mixed-precision, and sparse matrix operations, which are crucial in scientific applications such as climate and weather prediction - the primary focus of this study. Through experimentation and analysis, we showcase PaRSEC's ability to significantly boost computational efficiency and scalability across a range of computationally intensive and less intensive tasks on various hardware architectures. Our findings not only underscore the potential of PaRSEC in advancing sustainable, efficient, and accurate environmental modeling and forecasting, but also emphasize the growing necessity of task-based runtime systems in supporting the next generation of matrix algebra libraries.

14th
2:00 PM–2:30 PM

Experiences Porting Distributed Applications to Asynchronous Tasks: A Multidimensional FFT Case-study

Alexander Strack
University of Stuttgart

Parallel algorithms relying on synchronous parallelization libraries often experience adverse performance due to global synchronization barriers. Asynchronous many-task runtimes offer task futurization capabilities that minimize or remove the need for global synchronization barriers. Task futurization can improve overall algorithmic performance. However, some applications are better suited than others for migration to an asynchronous many-task model. This paper conducts a case study of the multidimensional Fast Fourier Transform to identify which applications will benefit from the asynchronous many-task model. Our principle focus is the popular FFTW library. We use the asynchronous many-task model HPX and a one-dimensional FFTW backend to implement multiple versions using different HPX features and highlight overheads and pitfalls during migration. Furthermore, a new HPX backend has been implemented and added to FFTW. The case study analyzes shared-memory scaling properties between our HPX-based parallelization and FFTW with its pthreads, OpenMP, and HPX backends. We examine the performance tradeoff of FFTW's algorithmic planning. The case study also compares FFTW's backends paired with MPI to a purely HPX-based implementation in a distributed environment.

DLA-Future: a task-based linear algebra library which provides a GPU-enabled distributed eigensolver.

14^{th}
2:30 PM–3:00 PM

Raffaele Solcà
ETH Zurich

DLA-Future implements an efficient GPU-enabled distributed eigenvalue solver using a software architecture based on the latest C++ std::execution concurrency proposals. The state-of-the-art linear algebra implementations LAPACK and ScaLAPACK were designed for legacy systems and employ the so-called fork join parallelism, which can perform inefficiently on modern architectures. The benefits of task-based linear algebra implementations are significant. The reduction of the number of synchronization points and the ease of overlapping computation with communication are two of the main benefits that lead to improved performance. In specific cases (which depend on the problem sizes, dependencies and number of independent operations), the ability to schedule multiple algorithms concurrently yields a noticeable reduction of time-to-solution.

We present the implementation of DLA-Future and the results on different types of systems starting from Piz Daint multicore and GPU partitions (Intel Broadwell/Haswell, Nvidia P100 GPUs), moving to more recent architectures available in ALPS (AMD Zen 2 CPUs, Nvidia A100 GPUs) and LUMI (AMD MI250x GPUs). The benchmark results are divided into two categories. The first contains a comparison of DLA-Future against widely used eigensolver implementations. The second category showcases the performance of the eigensolver in real applications. We present results generated with CP2K and SIRIUS, where DLA-future support was easily added thanks to the C-API provided, which is drop-in compatible with the ScaLAPACK interface.

14th
3:30 PM–4:00 PM

A Suite of Codes for Benchmarking and Testing Mutex-Based Parallel Systems

Max Morris
Louisiana State University

We present a collection of parallel codes in various programming languages covering common uses and implementations of mutual exclusion. This provides us with a basis for evaluating existing parallel systems for speed and efficiency, and a robust test suite for evaluating the correctness of new parallel systems. We also present benchmark results comparing the performance of the std:: and hpx:: libraries. We will make the suite publicly available and make available a process for accepting new submissions.

Speaking Pygion: Experiences Writing an Exascale Single Particle Imaging Code

14^{th}
4:00 PM–4:30 PM

Alex Aiken
Stanford University

The goal of the SpiniFEL project was to write, from scratch, a single particle imaging code for exascale supercomputers. The original vision was to have two versions of the code, one in MPI and one in Pygion, a Python-based interface to the Legion task-based runtime. We describe the motivation for the project, some of the programming challenges we encountered along the way, what worked and what didn't, and why only the Pygion code eventually ran at scale.

Taskflow: A General-purpose Task-parallel Programming System

14^{th}
4:30 PM–5:00 PM

Tsung-Wei Huang
University of Wisconsin at Madison

The Taskflow project addresses the long-standing question: "How can we make it easier for C++ developers to write efficient parallel programs with high productivity?" Modern computing applications rely on many parallel computing resources to achieve new performance milestones that were previously out of reach. However, programming these parallel computing resources is not an easy task because there are many technical details, such as abstraction, scheduling, concurrency, and load balancing. Many of these technical details are known difficult to program correctly.

To overcome this challenge, Taskflow develops a simple and powerful task graph programming model to enable efficient implementations of parallel decomposition strategies. Our programming model empowers users with both static and dynamic task graph constructions to implement various computational patterns, including in-graph control flow, composition, and on-the-fly tasking. Taskflow also introduces an efficient work-stealing scheduling algorithm that can efficiently balance the number of available workers with dynamically generated task parallelism. We have applied Taskflow to many industrial applications and achieved significant performance improvement through task graph parallelism. Taskflow is being used by many academic and industrial parallel computing applications, such as AMD Xilinx, Nvidia GameWorks, Tesseract Robotics, OSSIA sequencer, and so on.

Structure of the Talk: The takeaway of the talk consists of the following five items:

1. Express your parallelism in the right way
2. Program task graph parallelism using Taskflow
3. Program dynamic task graph parallelism using Taskflow
4. Overcome the scheduling challenges
5. Demonstrate the efficiency of Taskflow

We will start by explaining the importance to express the parallelism in the right way. Then, we will introduce two essential programming paradigms of Taskflow, static task graph parallelism and dynamic task graph parallelism. Next, we will discuss how Taskflow overcomes critical scheduling challenges in running static and dynamic task graphs. Finally, we will present successful industrial use cases of Taskflow.

Too Much Parallelism, and the Need for Compiler Support in Asynchronous Many-Task Systems

15^{th}
9:00 AM–10:00 AM

Jean Treichler
NVIDIA

From the start, Legion was designed to be part of a larger software stack, with a distributed execution runtime (i.e. Realm) below it, but also libraries, frameworks, and DSLs above it. Early successes with Legion came from application code written directly to the Legion API, but more recently the vision of being an enabler for higher levels of abstraction is coming to fruition. I will review several of these efforts, talk about their successes in scaling (whether to larger systems, larger workloads, or larger programmer audiences), and that will lead me into discussion of some key challenges that are becoming more critical as this scaling continues.

Legion, and really all AMT systems, are at risk of becoming victims of their success. If anything, they are too good at extracting parallelism from application code, and this manifests as increasing demands on the runtime portions of these systems at larger scales. Much of the analysis and decision making being performed in real time (and therefore with real overhead) is not actually contributing to improved performance because there was already enough work to keep processors busy and data movement latencies hidden. I'll touch on past and current efforts to attack this at the application level and in runtime implementation, but the best place in the stack to perform optimizations like this (i.e. compilers) is severely underrepresented, and we should fix that.

Futures for dynamic dependencies – Parallelizing the H-LU Factorization

15^{th}
10:30 AM–11:00 AM

Rüdiger Nather
University of Kassel

We present a novel task-parallel algorithm for the LU factorization of hierarchical matrices (H-matrices). These arise from, e.g., boundary element methods. H-matrix arithmetic typically exhibits complicated dependency patterns which pose a challenge for effective parallelization. Existing methods express these dependencies with in/out parameter qualifiers. In this work, we investigate the use of the future construct as an alternative. Futures serve as placeholders for eventually computed values and have been used to express similarly complicated dependencies in other contexts. Unfortunately, support for futures in current runtime systems is insufficient. Therefore, we implemented our own proof-of-concept runtime system atop the C++20 standard. It is designed as a header-only library and utilizes coroutines to model tasks. We use task dependencies not only to determine the order of execution, but also for effective load balancing. Our results show that the approach provides a clear, concise and reasonably efficient way to perform the parallel H-LU factorization.

Scalable Block-Sparse Matrix Multiplication Using Template Task Graphs

15^{th}
11:00 AM–11:30 AM

Joseph Schuchart
University of Tennessee, Knoxville

Block-sparse matrix-matrix multiplication is an important operation in many scientific fields, including quantum physics and general tensor operations. Template Task Graph (TTG) is a programming model for flowgraph-based composition of high-performance algorithms executable on distributed heterogeneous computer platforms. The TTG API abstracts out the details of the underlying task and data flow runtime. This paper describes the implementation of 2.5D SUMMA (Scalable Universal Matrix Multiplication Algorithm) in the Template Task Graph (TTG) programming model. We will discuss the structure of the task graph and the operations involved, including strategies for controling the distribution of matrix blocks. We will provide a performance evaluation of our implementation and briefly discuss our intended approach for future support of accelerators.

15th
11:30 AM–12:00 PM

A Peek into Multi-Vendor and Multi-device Heterogeneity and Portability of MatRIS using IRIS Task based Runtime

Mohammad Alaul Haque Monil
Oak Ridge National Laboratory

MatRIS is a multilevel math library abstraction for scalable and performance-portable sparse/dense BLAS/LAPACK operations. Including IRIS task-based runtime at the bottom level, MatRIS exposes three levels of abstraction to make the tiled algorithms completely architecture agnostic. MatRIS ensures the decomposition and creation of tasks that represent the necessary encapsulation of the optimized kernels from the vendor and open-source math libraries. Once built, MatRIS can select different combinations of accelerators at runtime, which in turn makes it portable to diverse heterogeneous architectures. Using the features of managing heterogeneity of IRIS runtime, MatRIS algorithms exclude the need to specify any orchestration and data transfer. In this study, we provide insight into how serial task abstraction of tiled Cholesky factorization of MatRIS is made portable and scalable in the case of multi-device and multi-vendor heterogeneity on different heterogeneous systems, including a cloud node with NVIDIA and AMD GPUs.

An abstraction for distributed stencil computations using Charm++

15^{th}
1:30 PM–2:00 PM

Aditya Bhosale
University of Illinois Urbana-Champaign

Python has seen significant adoption by the scientific computing community in recent years. Libraries such as Numpy and SciPy have made it possible to prototype complex physical systems with decent performance and low programming effort. In particular, computations on structured grids that are at the heart of several numerical methods such as finite difference, finite volume, geometric multigrid, etc. are highly regular and have static dependence patterns which can be expressed succinctly using Numpy's slicing notations. Numpy execution is limited to a single thread and can occasionally make calls to high-performance multi-threaded BLAS libraries for certain in-built functions. However, for any arbitrary computations on structured grids, Numpy suffers from significant Python overheads and can potentially create several temporary arrays in the process. Moreover, Numpy doesn't scale to distributed memory machines. Thus, domain scientists often have to reimplement the application in a low-level language after prototyping in Python to get good parallel performance. There have been several libraries developed for distributed array computations in Python to address this issue. These are usually developed as either a compiler for a subset of Python or a drop-in replacement for Numpy. However, these libraries either require a significant rewrite of the source code or often show poor strong scaling performance due to high Python overheads. In this paper, we present CharmStencil, a high-level abstraction for expressing computations on structured grids with a Numpy-like Python frontend and a Charm++ backend. CharmStencil is a part of the broader CharmTyles project which aims to build multiple domain-specific abstractions, with a high-level frontend, combined with a parallel domain-specific library at the backend. We present a client-server model that we use to build our abstraction to preserve the productivity offered by tools such as Jupyter notebooks on the frontend, while the actual computations are executed by a highly efficient Charm++ backend. The frontend employs optimizations such as lazy evaluation and message coalescing to avoid the creation of temporary arrays and minimize communication latency between the frontend and the backend. The client-server model with an asynchronous frontend is able to provide a pipelined execution by overlapping the Python overhead incurred by the frontend with useful computation on the backend, thus making our abstraction highly scalable. Finally, we compare our performance against state-of-the-art distributed array computation libraries and compilers for Python.

HPX in the HPC Cloud: Evaluating Overheads of Cloud resources for HPX/Kokkos using an astrophysics application

15^{th}
2:00 PM–2:30 PM

Patrick Diehl
Louisiana State University

Cloud computing for high performance computing resources is an emerging topic. This service is of interest to researchers who care about reproducible computing, for software packages with complex installations, and for companies or researchers who need the compute resources only occasionally or do not want to run a supercomputer on their own. The connection between HPC and containers is exemplified by the fact that Microsoft Azure's Eagle cloud service machine is number three on the November 23 Top 500 list. For cloud services, the HPC application and dependencies are installed in containers, e.g. Docker, Singularity, or something else, and these containers are executed on the physical hardware. Although containerization leverages the existing Linux kernel and should not impose overheads on the computation, there is the possibility that machine-specific optimizations might be lost, particularly machine-specific installs of commonly used packages. In this paper, we will use an astrophysics application using HPX-Kokkos and measure overheads on homogeneous resources, e.g. Supercomputer Fugaku, using CPUs only and on heterogeneous resources, e.g. LSU's hybrid CPU and GPU system. We will report on challenges in compiling, running, and using the containers as well as performance differences.

Distributed Asynchronous Contact Mechanics with DARMA/vt

15^{th}
2:30 PM–3:00 PM

Nicholas Morales
Sandia National Laboratories

Contact mechanics, or the modeling of the impossibility of interpenetration of solid objects, is fundamental to computational solid mechanics (CSM) applications yet are oftentimes the most challenging in terms of computational efficiency and performance.

These challenges arise from the irregularity and highly dynamic nature of contact problems, particularly with algorithms designed for distributed memory architectures. First among these challenges is the inherent load imbalance when distributing contact load across compute nodes. This imbalance is highly problem dependent, and relates to the surface area of contact manifolds and the volume around them, rather than the distribution of the mesh over compute nodes, meaning the application load can vary drastically over different phases. The dynamic nature of contact problems motivates the use of distributed asynchronous many-tasking (AMT) frameworks to efficiently handle irregular workloads.

In this paper, we present our work on _distBVH_, a distributed contact solution using the DARMA/vt library for asynchronous tasking that is also capable of running on-node Kokkos-based kernels. We explore how _distBVH_ addresses the various challenges of CSM contact problems. We evaluate the use of many of DARMA/vt's dynamic load balancers, including decentralized load balancers such as _TemperedLB_, and demonstrate how our load balancing approach can provide significant performance improvements on various computational solid mechanics benchmarks. Additionally, we show how our approach can take advantage of DARMA/vt for tasking and efficient on-node kernels using Kokkos to scale over hundreds of processing elements.

Rethinking Programming Paradigms in the QC-HPC Context

15^{th}
3:00 PM–3:30 PM

Elaine Wong
Oak Ridge National Laboratory

Programming for today's quantum computers is making significant strides toward modern HPC compatible workflows, but key challenges still face the field. Quantum computing programming languages share some common ground, as well as their emerging runtimes and algorithmic modalities. In this short paper, we propose a refinement of the quantum processing unit (QPU) in order to understand the value it can play in linking QC with HPC and through some hypothetical examples, how its potential for scientific discovery might be realized.

MADNESS: A task-based application and runtime

16^{th}
9:00 AM–10:00 AM

Robert Harrison
Stony Brook University

MADNESS (Multiresolution ADaptive Numerical Environment for Scientific Simulation) started as an environment for fast and accurate numerical simulation in chemistry, but rapidly expanded to include applications in nuclear physics (HF and DFT for nuclei), boundary value problems, solid state physics, and atomic and molecular physics. It is portable from laptops to the largest supercomputers, and is open-source under GPL2 with developers/users in the US, Europe, Japan, and China. MADNESS provides a very high level of composition for science applications in terms of functions and operators rather than coefficients and matrix elements.

MADNESS employs adaptive multiresolution algorithms for fast computation with guaranteed precision, and separated representations for efficient computation in many dimensions. To guarantee precision, every function has an independent and dynamically refined "mesh" and composing functions or applying operators can change the mesh refinement. These meshes are represented as 2^d-trees, where d is the dimensionality of the problem. These trees are typically poorly balanced, hence computing with such trees on a parallel machine poses significant challenges that are addressed by the task-based MADNESS parallel runtime.

The MADNESS runtime has evolved into a powerful environment for the composition of a wide range of parallel algorithms, not just on trees, but on any distributed data structures, including the block-sparse tensors in TiledArray. The central elements of the parallel runtime are a) futures for hiding latency and managing dependencies, b) global namespaces with one-sided access so that applications are composed using names or concepts central to the application, c) non-process centric computing through remote method invocation in objects in global namespaces, and d) dynamic load balancing and data redistribution. An application in the MADNESS runtime can be viewed as a dynamically constructed DAG, with futures as edges.

After over 15 years of experience, despite many successes task-based composition on the MADNESS runtime has proven to be challenging to use robustly and has not fully migrated to hybrid architectures. This talk will examine both the successes and failures of the runtime and programming model, and will touch upon central elements in the design of the Template Task Graph (TTG, developed in collaboration with ICL team and Virginia Tech) that is the foundation of the next generation of MADNESS.

16th
10:30 AM–11:00 AM

Only Pay for What You Need: Registration-Based Queueing in Asynchronous Many-Task Systems

Zane Fink
University of Illinois at Urbana-Champaign

Asynchronous Many-Task (AMT) systems offer promising productivity and performance benefits to the MPI+X paradigm. These systems eschew the process-centric view of MPI, instead offering hierarchical execution and scheduling, more closely matching today's heterogeneous systems. These systems execute multiple tasks by picking them from a queue. Because different messages have different purposes and priorities, AMT implementors introduce multiple bespoke message queues. This decision introduces complex trade-offs, such as the order in which queues are checked, that affect application performance. Moreover, most application messages use a subset of the queues available in the AMT system, and application cycles are wasted checking empty queues.

To overcome these issues, we introduce registration-based queueing, where users register the message queues needed for their application, and only these queues are checked during application runtime. Moreover, we introduce an API that lets users control how often each message queue is checked. We implement registration-based queueing in the Charm++ runtime system, and demonstrate its performance impact in a variety of different contexts, demonstrating the substantial benefits of registration-based queueing in AMT systems.

Dynamic Tuning of Core Count to Maximize Performance in Object-based Runtime systems

16^{th}
11:00 AM–11:30 AM

Kavitha Chandrasekar
University of Illinois at Urbana-Champaign

Relatively recent developments in supercomputer nodes such as increasing physical and virtual core counts per node aim to speed up HPC application execution time. However, not all applications benefit from increased thread level parallelism. Additionally, the best performing thread count may not be known apriori as it can vary with application or with input size for a given application. This motivates the need for dynamically tuning the number of threads or cores used by an application, at run-time. However, such tuning of core counts in popular object-based or task-based runtime system is non-trivial since objects or tasks are anchored to processing elements (PEs) for locality. In this work, we identify the steps for adaptive tuning of core count to the most performant configuration, at run-time, for an object-based runtime system, Charm++. We show benefits of dynamic profiling and adaptively selecting core (physical or virtual) count for a variety of applications including compute, memory and cache-intensive applications. Specifically, we show that our mechanism can improve performance by 15% in presence of memory contention, by 40% when considering virtual cores, and by 25% under a power-constrained setting, for proxy applications in Charm++.

16th
11:30 AM–12:00 AM

The Asynchronous Low-level Programming Interface (ALPI): Enabling Portable Task-Aware Libraries Across Different Runtime Systems

Kevin Sala

Barcelona Supercomputing Center

As transistor density improvements plateau, cost-effective high-density chip production becomes challenging. Computer systems are overcoming these limitations by including multiple accelerators. Concurrently, exploiting parallelism in distributed and heterogeneous systems remains challenging. Task-based programming models are a promising alternative to exploit complex systems. However, integrating different communication, offloading and storage APIs within tasks poses performance and deadlock risks. Several Task-Aware libraries, such as TAMPI, TAGASPI, TACUDA, TASYCL and TACUDA, have been developed to integrate blocking and non-blocking APIs with task-based programming models efficiently. In this paper, we introduce the Asynchronous Low-level Programming Interface (ALPI) to enable the interoperability and portability of Task-Aware libraries across various programming models and runtime systems. We present ALPI's implementation within the Nanos6 runtime and nOS-V library, which facilitates the integration of Task-Aware libraries into OmpSs-2 and OpenMP programming models, respectively. This work presents a step toward improving the portability and interoperability of Task-Aware libraries across different programming models and runtime systems.

IRIS Reimagined: Advancements in Intelligent Runtime System for Task-Based Programming

16^{th}
12:00 AM–12:30 PM

Narasinga Rao Miniskar
Oak Ridge National Laboratory

Task based programming models are getting higher traction in scientific computing. IRIS is a portable runtime system exploiting multiple heterogeneous programming systems and can discover available resources, manage multiple diverse programming systems (e.g., CUDA, Hexagon, HIP, Level Zero, OpenCL, and OpenMP) simultaneously. It takes constraints of task dependencies and provide customizable scheduling policies to map the tasks to heterogeneous devices. In this paper, we present new capabilities added to the IRIS to make it highly portable, easy heterogeneous programming, build friendly and performance efficient. The capabilities include the addition of vendor specific kernels support, runtime system with foreign function interface to eliminate to write wrapper/boiler-plate code for heterogeneous kernels, easy and configurable CMake based build environment, and automatic and efficient data transfers and orchestration.

Posters

A Lightweight Communication Interface for Asynchronous Many-Task Systems

Omri Mor
University of Illinois Urbana-Champaign

MP underpins communications in a wide swath of HPC applications. These have functioned very well for the programming paradigms that they were primarily designed for, namely bulk-synchronous applications using explicit message-passing for communication. However, asynchronous many-task runtimes present different communication characteristics than most traditional HPC applications; these communication interfaces are not well-optimized and do not serve the needs and requirements of AMT runtimes. We present an overview of the design of the Lightweight Communication Interface (LCI), a communication library and research tool developed at the University of Illinois to study communication interface and design decisions in the context of multithreaded communication and dynamic compute frameworks, and demonstrate how it can be used to improve the performance of AMT runtimes. We showcase current work with the HPX and PaRSEC frameworks and compare results with a set of task-based latency and bandwidth benchmarks.

MatRIS 1.0: A Portable abstraction of Math Library using IRIS Task based Runtime for Multi-device and Multi-Vendor Heterogeneity

Mohammad Alaul Haque Monil
Oak Ridge National Laboratory

MatRIS is a multilevel math library abstraction for scalable and performance-portable sparse/dense BLAS/LAPACK operations. Including IRIS task-based runtime at the bottom level, MatRIS exposes three levels of abstraction to make the tiled algorithms completely architecture agnostic. MatRIS ensures the decomposition and creation of tasks that represent the necessary encapsulation of the optimized kernels from the vendor and open-source math libraries. Once built, MatRIS can select different combinations of accelerators at runtime, which in turn

makes it portable to diverse heterogeneous architectures. Using the features of managing heterogeneity of IRIS runtime, MatRIS algorithms exclude the need to specify any orchestration and data transfer. In this study, we provide insight into how serial task abstraction of tiled Cholesky factorization of MatRIS is made portable and scalable in the case of multi-device and multi-vendor heterogeneity on different heterogeneous systems, including a cloud node with NVIDIA and AMD GPUs.

Additional information

Addresses

Workshop venue

UT Student Union Room (Room 169)
1502 Cumberland Ave, Knoxville, TN 37916

Hotel

Hilton Knoxville, 501 W Church Ave, Knoxville, TN 37902
(865) 523-2300

Banquet

Calhoun's On The River, 400 Neyland Dr, Knoxville, TN 37902
(865) 673-3355

Restaurants

Walking distance

Knoxville's market square and Gay street offer a wide selection of restaurants and bars so this list is far from exhaustive:

- Tupelo Honey – Southern comfort food with a creative twist, plus craft beers & cocktails.: 1 Market Square, Knoxville, TN 37902 (865-522-0004)
- Cafe4 – Refined Southern classics, wine & beer in a comfortable dining room with a mezzanine coffee shop: 4 Market Square, Knoxville, TN 37902 (865-544-4144)
- Stock & Barrel – Stylish restaurant featuring thoughtfully sourced burgers & an extensive selection of bourbons: 35 Market Square, Knoxville, TN 37902 (865-766-2075)
- Vida – Vibrant restaurant in a historic bank, offering Latin dishes & a swanky cocktail lounge in a vault: The Holston, 531 S Gay St, Knoxville, TN 37902 (865-544-8564)

The Old City

A short walk away from the hotel lies Knoxville's Old City, a vibrant night life area with good food and bar choices:

- Kaizen – Popular place serving steamed buns, Asian dishes & draft beer in a chill space: 127 S Central St, Knoxville, TN 37902 (865-409-4444)
- Boyd's Jig & Reel: This cozy pub serving traditional meals & libations hosts live Scottish, Irish & Appalachian music: 101 S Central St, Knoxville, TN 37902 (865-247-7066)
- Lofty taproom doles out Mexican & Southern-inspired eats & draft beers: 100 Broadway SW, Knoxville, TN 37902 (865-999-5015)

Author Index

Aiken Alex, 13

Bhosale Aditya, 19

Cao Qinglei, 9
Chandrasekar Kavitha, 25

Diehl Patrick, 20

Fink Zane, 24
Furmento Nathalie, 6

Harrison Robert, 23
Huang Tsung-Wei, 14

Miniskar Narasinga Rao, 27
Monil Mohammad Alaul Haque, 18, 29
Monsalve Diaz Jose M, 7
Mor Omri, 29
Morales Nicholas, 21
Morris Max, 12

Nather Rüdiger, 16

Posner Jonas, 8

Sala Kevin, 26
Schuchart Joseph, 17
Solcà Raffaele, 11
Strack Alexander, 10

Taylor
Chris, 1
Treichler Jean, 15

Vetter Jeffrey, 5

Wong Elaine, 22