Workshop on Asynchronous Many-Task Systems 2023

Venue:
Center of Computation & Technology
Louisiana State University

February 26-28, 2023

This workshop is sponsored by

- Louisiana State University Center for Computation & Technology
- 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

Scientific committee

Technical program chair

- Patrick Diehl, Louisiana State University (USA)
- Hartmut Kaiser, Louisiana State University (USA)
- Peter Thoman, University of Innsbruck (Austria)

Technical program

- Bita Hasheminezhad, NASA Ames Research Center (USA)
- Irina Demeshko, NVIDIA
- Brad Richardson, Sourcery Institute (USA)
- Patricia Grubel, Los Alamos National Laboratory (USA)
- Kevin Huck, University of Oregon (USA)
- Pedro Valero Lara, Oak Ridge National Laboratory (USA)
- Dirk Pflüger, University of Stuttgart (Germany)
- Metin H. Aktulga, Michigan State University (USA)
- Roman Iakymchuk, Sorbonne Universite (France)
- Huda Ibeid, Intel
- Ben Bergen, Los Alamos National Laboratory (USA)
- Dirk Pleiter, KTH Royal Institute of Technology (Sweden)
- Didem Unat, Koç University (Turkey)
- Keita Teranishi, Sandia National Laboratories (USA)
- Abhinav Bhatele, University of Maryland, College Park (USA)
- Daisy Hollman, Google
- Gregor Daiß, University of Stuttgart (Germany)
- Najoude Nader, Louisiana State University (USA)
- Rod Tohid, Louisiana State University (USA)
- Dominic Marcello, Louisiana State University (USA)
- Sebastian Ohlmann, Max Planck Computing & Data Facility (Germany)

Logistics

- Karen Jones and Jennifer Claudet, Louisiana State University

Welcome Address

Contents

Industrial talk

An Azimuth to RISC-V HPC

27^{th}

Chris Taylor
Tactical Computing Lab

The RISC-V community has ratified several ISA extensions creating a viable path for the co-design, implementation, and production of RISC-V HPC hardware. In light of availability of a path to RISC-V HPC hardware, Tactical Computing Labs will present an introduction to the RISC-V, highlight portions of the ISA specification most relevant to the HPC community, present an overview of the current state of RISC-V HPC software, and identify avenues for the HPC runtime system community to engage with the RISC-V community.

Talks

Note that blue names indicate virtual talks.

TBA

15^{th}
9:00 AM–10:00 AM

Michelle Strout
HPE

15^{th}
10:30 AM–11:00 AM

Active Memory Architecture for Asynchronous Graph Processing

Thomas Sterling
Indiana State University

The leading-edge of High Performance Computing is challenged by the end of Moore's Law and new applications' demands in, among other areas, "AI" as the term is being currently employed in the common lexicon. Special Purpose Devices and, in some cases, entire new class of systems have attracted significant investment in recent years with the intent of meeting the rapidly growing demand (at least in appearance) for supervised machine learning platforms. More generally, for both dynamic numeric and informatic problems, the basic data structure may not be sparse matrices but rather time-varying and irregular graphs. AMR and N-body numeric problems and unsupervised machine learning, contextual natural language processing, searches, sorting, hypothesis testing, decision making, and a host of NP complete problems requiring non-deterministic but convergent solutions make up a wide-array of present and future workflows requiring new large-scale solutions. An innovative class of memory-centric architectures are emerging as a research focus to address both dimensions of the design and operation space. One such is the "Active Memory Architecture" under development as an example of a novel memory-centric system incorporating non von Neumann architecture structures, semantic constructs, graph and runtime related overhead primitive mechanisms, and runtime resource management and task scheduling methods. This advanced technical strategy has been sponsored by NASA and is supported by IARPA/ARO. The fundamental principles and planned methods being undertaken with the intent of modeling, simulation, and evaluation will be presented along with a completed FPGA-based graph accelerator prototype.

Command Horizons: Coalescing Data Dependencies while Maintaining Asynchronicity

15^{th}
11:00 AM–11:30 AM

Peter Thoman
The University of Padua

Co-Authors: Philip Salzmann

The highest tiers of performance and efficiency in contemporary large-scale HPC systems are generally achieved by accelerator clusters, which are commonly programmed with low-level or vendor-specific approaches such as MPI+CUDA. The Celerity runtime system provides a data-flow-centric high-productivity API for implementing HPC applications on such clusters, based on the established SYCL industry standard. It is designed to alleviate the development and maintenance burdens inherent in distributed memory systems as well as those introduced by accelerator programming.

A core feature of Celerity is the declarative specification of any given task's resource requirements with so-called "range mappers", which can be provided either in terms of existing patterns, or more freely constructed with functors. Based on only this information, the Celerity system asynchronously builds a task and distributed command graph at runtime, transparently splitting kernels across multiple nodes and performing the required data transfers.

In order to implement this automatic data dependency computation and command generation, the runtime system needs to precisely track the state of each distributed data buffer and its content in the system. This imposes challenges on the algorithms and data structures employed, particularly when scaling deeper – i.e. more time steps – and with access patterns that generate new data and require a growing subset of it.

In this talk, we will present Task Horizons, a concept implemented in the Celerity system which allows capping the depth complexity axis of data structures with a freely configurable trade-off between structural complexity, and the level of asynchronicity possible during execution.

15^{th}
11:30 AM–12:00 PM

Performance Portability using Standard C++ with SYCL

Hugh Delaney
Codeplay Software

The proliferation of accelerators, in particular GPUs, over the past decade is impacting the way software is being developed. Most developers who have been using CPU based machines are now considering how it's possible to improve the performance of applications by offloading execution to many core processors. Many emerging disciplines including AI, deep neural networks and machine learning have shown that GPUs can increase performance by many times compared to CPU-only architectures. New hardware features such as "tensor cores" are also starting to emerge to address specific problems including mixed precision computing. The new challenge for developers is figuring out how to develop for heterogeneous architectures that include GPUs made by different companies. Currently the most common way to develop software for GPUs is using the CUDA programming model but this has pitfalls. CUDA uses non-standard C++ syntax and semantics, is a proprietary interface, and can only be used to target Nvidia GPUs. Alternatives include HIP which offers another proprietary programming interface only capable of targeting AMD GPUs.

This presentation will demonstrate how standard C++ code with SYCL can be used to achieve performance portability on processors from multiple vendors including Nvidia GPUs, AMD GPUs and Intel GPUs. The SYCL programming interface is a royalty free and industry defined open standard designed to enable the latest features of accelerators. Using an open source project, we'll show how standard C++ syntax and semantics are used to define the SYCL kernel and memory management code required to offload parallel execution to a range of GPUs. Further to this, we'll explain how easy it is to compile this C++ code using a SYCL compiler so that it can be run on Nvidia, AMD and Intel GPUs and compare this execution performance with the same code written using proprietary CUDA and HIP environments.

Khronos SYCL heterogeneous programming for future codesign with RISC-V, HPC, and AI

15^{th}
12:00 PM–12:30 PM

Michael Wong
Codeplay Software

SYCL is the Khronos heterogeneous programming language that is selected for multiple DOE Exascale programming model including Aurora, Perlmutter, and Frontier.

By focusing on open ISO languages such as C++ with the addition of heterogeneous offloads to support the newest accelerator hardware, this talk aims to showcase how SYCL can also become an open standard for HPC and AI. In that respect, Kokkos, and HPX have both been built on top of SYCL.

Technology trends require software/hardware co-design for HPC and AI systems. Open-standard software programming models such as Khronos SYCL and oneAPI enables this co-design and allows us to support initiatives such for parallelism and acceleration for Exascale computing. In future, we plan to support RISC-V® processors as accelerators for HPC and Datacenter and Cloud Computing.

15^{th}
2:00 PM–2:30 PM

Qthreads: A Lightweight Threading Library for the Many-core Era

Jan Ciesko
Sandia National Laboratories

Co-Authors: Stephen Olivier and Ronald Brightwell

Qthreads is a parallel programming library based on user-level threading. It offers a rich API for thread management and synchronization. The API enables expression of producer-consumer relationships and of parallel patterns. Locality-aware optimizations are available through a set of task schedulers with optional work stealing. Qthreads is used in several downstream projects, including HPE's runtime system for Chapel, and serves as a platform for pioneering novel programming model features, resource-aware scheduling techniques, and interoperability. In this talk, we give an introduction to Qthreads and explain relevant concepts and design choices of the interface and implementation. Further, we discuss a representative use case. Finally, we provide an overview of the current research areas and conclude the talk with an invitation to engage with the community. SNL is managed and operated by NTESS under DOE NNSA contract DE-NA0003525.

Kokkos' Role for Performance Portability in AMT Systems

15^{th}
2:30 PM–3:00 PM

Christian Trott
Sandia National Laboratories

Kokkos is at its core a data parallel C++ Programming Model designed for Performance Portability. Originally developed at Sandia National Laboratories, it is now maintained by an open-source development team spanning multiple US National Laboratories. Today, Kokkos is commonly used in high performance computing applications and libraries in conjunction with MPI. This mix of MPI and Kokkos establishes a clear separation of concerns: Kokkos is used to expose fine grained data parallelism, which can be mapped portably to all common HPC hardware architectures, while MPI deals with coarse grained parallelism - in particular at an inter-node level. This talk will discuss how Kokkos could - and likely should - play the same role for asynchronous many-task models. To that end the presentation will review the high-level design of Kokkos integration into Legion, HPX, and Uintah, and point out opportunities of leveraging capabilities in the larger Kokkos EcoSystem such as portable tools and math library support.

15^{th}
3:00 PM–3:30 PM

HPX - A C++ Library for Parallelism and Concurrency

Hartmut Kaiser
Louisiana State University

With the advent of modern computer architectures characterized by – amongst other things – many-core nodes, deep and complex memory hierarchies, heterogeneous subsystems, and power-aware components, it is becoming increasingly difficult to achieve best possible application scalability and satisfactory parallel efficiency. The community is experimenting with new programming models that rely on finer-grain parallelism, and flexible and lightweight synchronization, combined with work-queue-based, message-driven computation. The recently growing interest in the C++ programming language in industry and in the wider community increases the demand for libraries implementing those programming models for the language.

In this talk, we present HPX – A C++ Standards Library for Parallelism and Concurrency that is built around lightweight tasks and mechanisms to orchestrate massively parallel (and – if needed – distributed) execution. We also implement a full set of standard parallel algorithms and related asynchronous extensions to those. The library enables an asynchronous execution model that uses the concept of (Standard C++) futures to make data dependencies explicit, employs explicit and implicit asynchrony to hide latencies and to improve utilization, and manages finer-grain parallelism with a work-stealing scheduling system enabling automatic load balancing of tasks. Lately we have been experimenting with the new sender/receiver model that is currently being discussed as part of the C++ standardization process.

HPX is a C++ library exposing a higher-level parallelism API that is fully conforming to the existing C++11/14/17/20 standards and is aligned with the ongoing standardization work. This API and programming model has shown to enable writing highly efficient parallel applications for heterogeneous resources with excellent performance and scaling characteristics. This talk will highlight the implemented extensions to the C++ standard parallel algorithms and shows recent performance results.

PGAS: A View from Berkeley

16^{th}
9:00 AM–10:00 AM

Damian Rouson
Lawrence Berkeley National Laboratory

Partitioned Global Address Space (PGAS) programming models, languages, and libraries offer HPC software developers a set of abstractions that facilitate parallel communication and computation, including remote memory access and remote procedure calls. This talk will give a high-level overview of PGAS research and development at Berkeley Lab, covering our contributions to the PGAS software ecosystem, including our work on unit tests for the PGAS features in the LLVM Flang Fortran compiler; creating the Caffeine coarray Fortran parallel runtime library [1]; producing the UPC++ PGAS template library [2]; and developing the GASNet-EX exascale networking middleware that supports a range of PGAS languages, libraries, and frameworks [3]. The talk will also highlight the use of the aforementioned technologies in task-scheduling frameworks, including FEATS [4], Legion [5], and DepSpawn [6].

Portable Uintah framework for heterogeneous, asynchronous many-task runtime systems for Exascale Architectures

16th
10:30 AM–11:00 AM

Abhishek Bagusetty
Argonne National Laboratory

Co-Authors: John Holmen and Martin Berzins

The Uintah Computational Framework is being prepared to make portable use of current and emerging exascale systems, initially the DOE Aurora system through the Aurora Early Science Program. While Uintah's tasks have been ported to Kokkos for performance portability, Uintah's task scheduling infrastructure is not yet wholly portable. This paper describes the evolution of Uintah's task scheduling approach to be ready for such architectures. As a part of this, Uintah's Unified Scheduler, which uses raw CUDA, has been ported to HIP and SYCL to help better understand what abstractions are needed to develop a wholly portable task scheduler. This paper extends recent work by exploring performance portability on AMD and Intel GPUs. Results are shown for a benchmark executing workloads representative of typical Uintah application across ALCF and OLCF systems. The latest state of Uintah's support for Kokkos and challenges relating to preparing single-source code for AMD-, Intel-, and NVIDIA-based GPUs are also discussed

FleCSI: Scalability Studies and Challenges

16^{th}
11:00 AM–11:30 AM

Sumathi Lakshmiranganatha
Los Alamos National Laboratory

Co-Authors: Benjamin Bergen, Andrew Reisner, and Jonathan Pietarila Graham

The Flexible Computational Science Infrastructure (FleCSI) is a compile-time configurable framework for multi-physics applications development. FleCSI provides an abstraction layer to control different runtime interfaces, e.g., Legion, MPI and HPX for distributed-memory execution, and Kokkos and Kitsune for shared-memory execution. This presentation discusses the results of some scalability studies of iterative solvers developed on FleCSI comparing the Legion and MPI backends. We will share some of the challenges we encountered in completing this work.

16^{th}
11:30 AM–12:00 PM

Recent FleCSI Developments

Ben Bergen
Los Alamos National Laboratory

FleCSI is a task-based runtime abstraction layer that provides a single-source programming model for shared and distributed-memory parallelism. FleCSI also provides several user-configurable data structures that can be combined to create custom interfaces for multi-physics application development. This talk will provide an overview of recent improvements and new capabilities, including a new tracing interface, support for arbitrary color-to-process mappings, and support for upcoming accelerated system architectures.

Framework for Extensible, Asynchronous Task Scheduling (FEATS) in Fortran

16^{th}
12:00 PM–12:30 PM

Brad Richardson
Archaeologic, Inc.

Co-Authors: Damian Rouson, Harris Snyder, and Robert Singleterry

Modern Fortran empowers developers to express parallel algorithms without directly referencing lower-level parallel programming models. Fortran's parallel programming features place Fortran within the Partitioned Global Address Space (PGAS) class of languages. Prior to the introduction of the parallel features of modern Fortran, most parallel scientific programs hardwired compiler directives (pragmas), run-time library procedure calls, and compiler-specific language extensions directly into the Fortran source code. Examples include Message Passing Interface (MPI) procedure calls, OpenMP compiler directives and compiler-specific CUDA Fortran. For data-parallel problems, application developers typically find it straightforward to implement their own parallel algorithms. Software that perform complex, heterogeneous, staged calculations, however, pose a much greater challenge. Such applications require careful coordination of the calculations of task dependencies as prescribed by directed acyclic graphs. In these cases, rolling one's own solution proves difficult and finding a customizable framework to extend becomes attractive.

To the best of our knowledge, FEATS is the first framework of its kind in modern Fortran. This paper discusses the techniques used in implementing FEATS. It describes the methods used for coordinating execution of tasks between images. It also describes the methods used to communicate data between tasks. The paper presents the positive and negative aspects of the approach, along with the beneficial features and shortcomings of the Fortran language. We describe what a user of the framework must provide, and how this is done, including presenting a working example.

Efficiency and raw speed vs generality and versatility in parallel programming models

16^{th}
2:00 PM–2:30 PM

Laxmikant (Sanjay) Kale
University of Illinois Urbana-Champaign

When we design parallel programming systems (languages, libraries and abstractions in general), we are faced with a multitude of use-cases, some arising from applications and some arising from community micro-benchmark suites. Initially, the focus of a developer is to support a narrow set of these. As the utility of the system broadens and additional applications come in to its ambit, the system needs to add new features and generalize existing features. Such additions tend to have a detrimental impact on performance. This challenge is especially acute for asynchronous task-based systems for a variety of reasons. This talk will deal with challenges of dealing with this tradeoff and potential solution strategies for it. It will build upon our experience with the Charm++ software stack, which also includes associated abstractions including Adaptive MPI.

Lessons Learned From Writing Task-Based Libraries

16^{th}
2:30 PM–3:00 PM

Alex Aiken
Stanford

This talk will summarize experiences over the last several years in writing task-based libraries for deep learning, tensor algebra, and solving sparse linear systems, among others. Because many performance-related decisions in task-based programs are late-bound, meaning they are left abstract in the program itself and only fixed when the program is run, it is possible to change these decisions easily, enabling rapid iteration after a code is written to find the combination of decisions that work best for a particular execution environment. This ability to productively modify performance-related decisions and the ability to easily change the partitioning of data have turned out to be key to achieving state-of-the-art and highly portable performance.

MatRIS: A fully agnostic solution for BLAS and LaPACK codes using the IRIS runtime

16^{th}
3:00 PM–3:30 PM

Pedro Valero-Lara
Oak Ridge National Laboratory

BLAS & LaPACK are crucial and core computation components in high performance computing and machine learning applications. Historically, hardware vendors and researchers have provided optimized math kernels for specific architectures. For this reason, the application developers have to decide which library and architecture to use and re-implement thousands of lines of code to port and optimize their codes to other architectures. Moreover, following the trend of heterogeneity, hardware manufacturers and vendors are releasing new architectures and their proprietary libraries that can harness the best possible performance for commonly linear algebra kernels. However, tuned kernels for one architecture are not portable to others. Moreover, the coexistence of different architectures in a single node made orchestration difficult. To address these challenges, we introduce MatRIS, a portable framework for BLAS & LaPACK functionalities. MatRIS ensures a separation between linear algebra algorithms and vendor library kernels using IRIS runtime. Such abstraction at the algorithm level makes implementation completely vendor-library and architecture agnostic. MatRIS uses IRIS runtime to dynamically select the vendor-library kernel and suitable processor architecture at runtime. We demonstrate that MatRIS can fully utilize different heterogeneous systems by launching and orchestrating different vendor-library kernels without any change in the source code.

This research reports the following contributions:

1- Improve portability and productivity for BLAS & LaPACK codes by separating the algorithm description (application details) from the implementation, tasks mapping (hardware features), and vendor library kernels.

2- Efficient utilization of different heterogeneous systems with a large number of computing components without changing one line of code.

3- Performance study of the MatRIS implementation on three different heterogeneous systems; one NVIDIA DGX-1 system with 1× Intel CPU and 4× NVIDIA, one node of the fastest TOP5001 supercomputer today, the ORNL's supercomputer Frontier, with 1× AMD CPU (64 cores) and 8× AMD GPUs, and one extreme heterogeneous system with 1× AMD CPU, and 8× GPUs (4× NVIDIA GPUs + 4× AMD GPUs).

Heterogeneous distributed runtimes for fine-granularity tasks: A possible revolution in parallel programming

17^{th}
9:00 AM–10:00 AM

George Bosilca
University of Tennessee, Knoxville

Challenges introduced by highly hybrid many-cores architectures have a lasting impact on the portability and performance of applications, partially due to traditional programming paradigms. These programming paradigms lack the flexibility and capabilities required to deal with large amounts of potential parallelism and a dynamic hybrid execution environment, putting the performance and scalability of applications at risk. Advances in task-based runtime have shown to provide a plausible solution to this problem, one that not only increase the domain scientists' productivity but also deliver codes that are more efficient, more scalable, and more adaptable to various hardware architectures, and show an increased portability potential to transition from one generation of hardware to another. This talk will describe a distributed task-based runtime, PaRSEC, and highlight its data management strategies and features to allow the implementation of highly efficient and scalable algorithms at any scale.

17^{th}
10:30 AM–11:00 AM

HPX and Kokkos: unifying asynchrony and portability on the path towards standardization

Mikael Simberg
Swiss National Supercomputing Centre

Co-Authors: Gregor Daiß

Modern hardware architectures are increasingly parallel, through both massively multicore CPUs and accelerators which often dominate the available compute on the newest nodes. Utilizing this hardware fully is increasingly difficult with traditional programming models. At the same time the hardware landscape has further diversified which increases the burden on application developers to run their applications anywhere. HPX, a tasking runtime with a focus on distributed applications and standards conformance, is ideally suited to tackle the first challenge. Kokkos, a performance portability layer, handles the second. However, using them together has not been straightforward until the integrations presented here. This talk presents the integration of HPX and Kokkos on multiple different levels with: 1. an HPX backend for Kokkos, which is the first asynchronous CPU backend for Kokkos; 2. HPX-Kokkos, a thin interoperability layer which combines primitives provided by HPX and Kokkos; and 3. The integration of all of the former into Octo-Tiger, an astrophysics application. The integration allows applications to make full use of the portability provided by Kokkos and move past the limitations of fork-join parallelism with the tasking provided by HPX. Finally, this talk will present the latest integration of the C++ std::execution proposal for asynchrony into HPX and Kokkos, and how it provides a path towards a standardized solution for portability and asynchrony in applications like Octo-Tiger.

Scheduling Many-task Applications on Multi-clouds and Hybrid Clouds

17^{th}
11:00 AM–11:30 AM

Peter Franz
Louisiana State University

Co-Authors: Peter Franz and Gerald Baumgartner

Cloud computing has been very successful for many types of applications, especially for applications that do not require frequent communication between different cloud nodes, such as MapReduce or graph-parallel algorithms. One of the advantages of cloud computing over a cluster or local supercomputer is its low cost, in particular when using virtual resources. However, for applications with fine-grained parallelism, it shows less than satisfactory performance. Renting dedicated clusters from cloud providers is expensive and virtual resources are heterogeneous in performance and latency.

A solution to running high-performance applications in the cloud is to structure them as many-task applications and to match the performance requirements of tasks to the available performance characteristics of cloud nodes. Since this requires periodic performance and latency measurements, a centralized task scheduler can become a bottleneck for large applications with fine-grained parallelism.

In previous research, we have demonstrated that a decentralized task scheduler can successfully distribute the tasks onto cloud nodes without the bottleneck of a centralized scheduler. The disadvantage of a decentralized scheduler is that it cannot guarantee perfect resource utilization: occasionally nodes can become temporarily idle.

For our scheduling approach, worker nodes are connected in an overlay graph. Each node periodically measures its performance and communication latency with neighbors on the graph and sends this information as well as the length of its task queue to its neighbors. After collecting the performance information from its neighbors, each node then normalizes the measurement results and plots them on a multi-dimensional grid. Spare tasks in the task queue are then sent to better-performing nodes, i.e., to nodes with shorter tasks queues, higher performance, and/or lower latency. This is achieved by comparing the difference between the node's own measurement results and a neighbor's measurement results with the desired direction or vector in which tasks should travel.

In previous research, we have demonstrate that this vector-scheduling works best if the underlying overlay graph reflects the distances between nodes. Clusters of nodes with low communication latency between them (e.g., if they reside in the same CPU or the same rack) are connected in a full graph. Between clusters of more distant nodes (e.g., at different ends of the warehouse-sized data center or at different cloud sites) there are fewer connections on the overlay graph to avoid tasks being shipped back and forth between distant nodes.

We have run experiments in CloudLab with both simulated and actual differences in performance and communication latency. As demo application, we have used sets of matrix multiplications of varying sizes. Our experiments demonstrate that the vector-scheduling approach results in good resource utilization and good load balancing of the tasks on the worker nodes.

Recently, we have implemented support in our cloud platform for scheduling tasks on multi-site clouds as well as on hybrid clouds. For this paper, we will concentrate on measurements demonstrating the feasibility of running many-task applications on multi-clouds and hybrid clouds.

Matrix Multiplication using Hedgehog's data flow graphs on multi-node GPU architectures

17^{th}
11:30 AM–12:00 PM

Nitish Shingde
University of Utah

Co-Authors: Martin Berzins, Timothy Blattner, Walid Keyrouz, and Alexandre Bardakoff

Asynchronous task-based systems offer the possibility of modeling better use of large-scale heterogeneous architectures. One such example is the Uintah framework. At this time hedgehog library of NIST presents excellent single-node performance and low overhead using the data flow graphs approach. This paper combines the two methods and evaluates their performance on multi-node GPU architecture using matrix-matrix multiplication. The algorithm builds on the hedgehogs' previous single-node implementation based on the outer product approach and expands it to multi-node architectures using MPI. To evaluate this approach, we compared the performance results against the SLATE and DPLASMA software which implement generic matrix multiplication in similar environments.

17th
12:00 PM–12:30 PM

Octo-Tiger: An HPX Based Code for Modelling Three-Dimensional Self-Gravitating Fluids

Dominic Marcello
Louisiana State University

Co-Authors: Patrick Diehl, Gregor Daiss, Sagiv Shiber, and Hartmut Kaiser

Octo-Tiger is an AMT based tool for modelling three-dimensional self-gravitating astrophysical fluids. It was designed to be particularly suited for modelling interacting binary star systems. It uses a finite volume technique to model the hydrodynamics and the fast multipole method to model the gravity. It is written entirely in C++ and uses HPX (High Performance ParalleX) for both node-level and distributed parallelism. The hydrodynamics and gravity modules have scalar, CUDA, HIP, and Kokkos (with SIMD support) implementations Octo-Tiger has been executed on a wide range HPC platforms including Japan's Fugaku, NERC's CORI and Perlmutter, ORNL's Summit, and CSCS's PizDaint. Here we will briefly describe Octo-Tiger, show some scientific results from a double white dwarf merger, and highlight our efforts to optimize the code. We will also outline the recent addition of a radiation transport module using explicit time integration and a two-moment closure relation.

Posters

Efficient Message Passing Support for Asynchronous Many-Task Systems

Jiakun Yan
University of Illinois Urbana-Champaign

Co-Authors: Marc Snir

The communication behaviors of asynchronous many-task systems are usually irregular and happen in a multithreaded environment. This leads to some mismatches between what task systems want and what their underlying communication libraries offer, leading to inefficiencies such as poor multithreaded performance, unnecessary memory copies and messages, unpredictable background task processing, and inefficient polling for completion. We are developing a low-level communication library, Lightweight Communication Interface (LCI), to explore ways to eliminate these mismatches and provide direct communication support to task systems. LCI's features include (a) flexible communication primitives and signaling mechanisms (b) better-multithreaded performance (c) explicit user control of communication resources. We are working on creating an LCI parcelport for HPX. We will report our current progress in this poster.

Evaluating and Improving Shared Memory Performance of HPX and OpenMP using Task Bench

Ioannis Gonidelis
STE||AR Group

Numerous benchmarking efforts have been made to compare both qualitatively and quantitatively, the large variety of asynchronous runtime systems available in the community. A standardized benchmarking solution, Task Bench, is being deployed in this research to explore the shared memory concurrency performance between two analogous Asynchronous Many-Task (AMT) systems: OpenMP and a C++ standard library for parallelism and concurrency (HPX). In this work, we focus on the shared memory features of HPX and neglect the distributed memory features. Based on the results of this analysis, we expose the characteristics relative to both interfaces and scheduling mechanisms of each system that inhibit

performance, and we present the corresponding improvements applied. Efficiency of the implemented testing ground, proper interfacing for asynchrony, and thorough examination of the scheduling can lead to significant performance gain in both systems and showcase shifting of any original measurements. In this research, we are not only trying to emphasize on the on-node performance of the two AMTs on top of Task Bench, but we are also trying to provide the community with a state-of-the-art guidance on how to minimize overheads in parallel code and emphasize the broadness of potential performance barriers.

Performance analysis and optimization on adapted Rotate algorithm

Chuanqiu He

Louisiana State University

The C++ standard defines many algorithms. Usually, a new C++ standard is updated every three years. HPX library needs to follow the C++ code standardization to support sequential and parallel implementation for all algorithms. But some of the HPX algorithms didn't adapt to the latest C++20 standard. I adapted rotate algorithms using the tag_invoke CPO mechanism to add the correct overloads for the algorithms as mentioned by the C++20 standard. Moreover, further optimize the adapted rotate according to the characteristics of rotate implementation by implementing core partition functionality, which enables left and right elements to be processed to obtain the corresponding proportion of a number of cores as needed. Then compare the speedup of rotate algorithm under different policies (parallel, sequential) and optimized rotate. More specifically, creating a benchmark for rotate algorithm and testing its performance changes when tweaking different parameters (like input size and the number of threads, chunk size). The performance of parallel rotate is much better than that of sequential rotate. And also, the core partition functionality worked, and rotate performance improved by about 15%.

The Guardian Language

Max Morris
Louisiana State University

Co-Authors: Steven R. Brandt

Guardian is a parallel programming language that compiles to Java, designed to work with the Javalin parallel programming library. Javalin provides a framework for parallel programming which is non-blocking, composable, orderable, compatible with fork-join pools, and free of deadlocks, all with minimal overhead. However, developing in Javalin requires an asynchronous continuation style of programming. This paradigm is indirect, involves deep nesting of lambdas, and requires adherence to contracts that are impossible to enforce in plain Java. Writing Javalin code in plain Java can be unintuitive and error-prone. Guardian aims to mend these concerns; the language introduces new, yet familiar syntax on top of Java to allow programming in a style more familiar to Java developers. It also prevents errors and pitfalls by enforcing Javalin's contracts at compile-time, guaranteeing safety.

Configuring performance of standard parallel algorithms using HPX

Karame Mohammadiporshokooh
Louisiana State University

HPX is a fully Asynchronous Many Task (AMT) runtime system extending the C++ programming language. HPX provides lightweight user-level tasks that run on top of kernel threads. In this poster we add configurations that tweak HPX's parallel algorithms and observer how performance is affected by that. Since different types of base algorithms (scan based, iterative, bulk etc) are exposed through the standard parallel algorithms we experiment with selectivity of cache and chunk size in order to achieve an optimal set of parameters for a given set of algorithms. Although this poster concentrates on a particular sample of an iterative based algorithm (adjacent_difference) the results can be extended (deduced) accordingly within this set. The performance measurements presented in this poster deliver useful information to the global community on how to organize the execution of these sets of algorithms in order to align the different execution characteristics (architecture, granularity, chunking etc) in favor of performance

---

Computational feasibility of simulating radiation induced changes in vasculature and blood flow rates

Maxwell Cole
Louisiana State University

Co-Authors: Wayne Newhauser and Patrick Diehl

Background

One of the most common biological effects of radiation is blood vessel damage, which can lead to deleterious effects such as radiation necrosis or atherosclerotic heart disease. In recent years, several groups have performed computational blood flow simulations in the heart. However, these simulations neglected radiation injury and are limited to the resolution of the imaging modalities, typically around 1-2 millimeters. Other computational methods for modeling radiation dose deposition and biological response had been applied only in small volumes of tissue. To understand the systemic effects of radiation on the entire body, computational methods must surpass greater length and time scales than previously achieved. In this study we aim to simulate radiation damage and analyze the effects of the resulting structural changes on blood flow in a system greater than 34 billion vessels, approximately the size of the human body.

Methods

Vascular Geometry: The vascular geometry of the system is constructed from a fractal algorithm to generate 3-dimensional scalable vessel networks. The vessels are represented as rigid, cylindrical tubes connected at junctions to form a closed network, with symmetric halves comprising of an arterial tree and a venous tree. Radiation Transport: Radiation transport is simulated using an amorphous track-structure method to model dose deposition from protons and secondary ionized electrons, or -rays. The biological response of the impacted vessels is modeled to fit experimental data. Fluid Dynamics: The resulting changes in blood flow are calculated utilizing a special case of the Navier-Stokes equation, known as the Poiseuille equation, for the motion of incompressible Newtonian fluids. The network is cast as a system of linear equations, and iterative Krylov methods are used to solve for the blood flow rates of each vessel. Computational Aspects: Due to the large-scale nature of the project, we must implement high-performance computing techniques on supercomputer clusters. A single vessel in our system is approximately 68 bytes. Therefore, representing the geometry of an entire human body requires over 1 terabyte of memory, greater than typically available on a single compute node. To overcome this, we will be integrating the High Performance ParalleX (HPX) C++ library for increased parallelism and concurrency, developed by our collaborators at the STE||AR group in Louisiana State University's Center for Computation and Technology (CCT). HPX is an Asynchronous Many Task runtime system that has shown superior parallel efficiency in large-scale projects.

Preliminary Results

Preliminary results from our laboratory have shown the computational feasibility of calculating blood flow in 17 billion vessels. We have also shown the feasibility of demonstrating whole-organ vascular injury from radiation. This was accomplished on a vascular network the size of the human brain (9 billion vessels) with dose simulated by an amorphous track-structure model consisting of 2 million protons. We are currently generating further preliminary computational results at the CCT's Rostam supercomputer cluster.

Additional information

Addresses

Workshop venue

Digital Media Center, Center for Computation & Technology, 340 E Parker Blvd., Baton Rouge, LA 70803

Hotel

The Cook Hotel at LSU, 3848 W Lakeshore Dr, Baton Rouge, LA 70808
(225) 383-2665

Banquet

LSU Faculty Club, 101 Tower Dr, Baton Rouge, LA 70803
(225) 578-2356

Restaurants

Walking distance

- The Chimes – Lively campus-area hangout from a local chain featuring a worldwide beer list & hearty bar fare: 3357 Highland Rd, Baton Rouge, LA 70802 (225 383-1754)
- Louie's Cafe – LSU-area fixture dating to 1941 serves a diner menu 24/7 in a classic lunch-counter setting: 3322 Lake St, Baton Rouge, LA 70802 (225 346-8221)
- Highland Coffees – Charming, airy locale with a laid-back vibe for coffee roasted on-site & a variety of baked goods: 3350 Highland Rd, Baton Rouge, LA 70802 (225 336-9773)

Local cuisine

- Parrain's Seafood Restaurant – Local seafood specialist cooking up Louisiana recipes in a rustic space with porch seating: 3225 Perkins Rd, Baton Rouge, LA 70808 (225 381-9922)

- Mike Anderson's - Baton Rouge – Area staple for regional seafood in a spacious, wood-lined setting with a sports-friendly vibe: 1031 W Lee Dr, Baton Rouge, LA 70820 (225 766-7823)
- Stroubes Seafood and Steaks – Chophouse presenting local preparations of meat & seafood in comfortable digs with a lounge: 107 3rd St, Baton Rouge, LA 70801 (225 448-2830)

Author Index