Best Practices in Running Collaborative GPU Hackathons: Advancing Scientific Applications with a Sustained Impact

Chandrasekaran, S.; Juckeland, G.; Lin, M.; Otten, M.; Pleiter, D.; Stone, J. E.; Lucio-Vega, J.; Zingale, M.; Foertter, F.;

Originally published:

July 2018

Computing in Science & Engineering 20(2018)4, 95-106

DOI: https://doi.org/10.1109/MCSE.2018.042781332

The OLCF GPU Hackathon Series: The Story Behind Advancing Scientific Applications with a Sustained Impact

Sunita Chandrasekaran*, Guido Juckeland^, Meifeng Lin[#], Matthew Otten**, Dirk Pleiter[$], John. E. Stone[‖], Juan Lucio-Vega[*], Michael Zingale[%] and Fernanda Foertter[##]

* University of Delaware, USA; (schandra, jluciove}@udel.edu
^ Helmholtz-Zentrum Dresden-Rossendorf, Germany; g.juckeland@hzdr.de
[#]Brookhaven National Laboratory, USA; mlin@bnl.gov
**Cornell University, USA; mjo98@cornell.edu
[$]Forschungszentrum Jülich, Jülich Supercomputing Centre, Germany; d.pleiter@fz-juelich.de
[‖]University of Illinois at Urbana-Champaign, USA; johns@ks.uiuc.edu
[%] Stony Brook University, USA; michael.zingale@stonybrook.edu
[##]NVIDIA, USA; ffoertter@nvidia.com

Abstract

The OLCF GPU Hackathons are a one-week code-development/learning event to better enable attendees to utilize GPUs. It only took three years to grow from a ``Let's give this a try"-event to a repeatedly copied format with several spin-offs that inspired HPC centers around the world. Sticking to a few fundamental principles---work on your own code, learn from your mentors just what you need and when you need it, stay flexible in achieving your goal---the week-long hackathon format created at Oak Ridge Leadership Computing Facility (OLCF) has been just the spark needed by many groups of scientists to light the fire of a wider GPU adoption in leading-edge as well as university-scale HPC environments. Most interestingly, the format enabled both ends of the experience spectrum---graduate students vs. postdoc fellows---the same kind of progress and chance of success.

Keywords: B.1.4.b Languages and compilers, D.1 Programming Techniques, M.12.0.b Case Studies in Scientific Applications

1. Introduction

How can experts prepare future computational scientists for the upcoming challenges in heterogeneous parallel computing when every University and college curriculum still teaches sequential programming concepts? How can well established domain scientists refactor their applications to keep up with the current hard- and software trends? The radical changes in the compute node architectures in the last decade, which have been introduced by adding hardware accelerators, have brought established programmers back to square one in using these novel devices. In that regard it is also an opportunity where students with their classroom experiences,

scientists and programmers with their domain knowledge and Graphic Processor Units (GPUs) programming environment developers can come together and learn from one another.

In 2014, the Oak Ridge Leadership Compute Facility (OLCF) was faced with the question of increasing the number of projects successfully utilizing the GPUs in the fast supercomputer available for research. It became obvious that there was a missing component between the six applications that were chosen by the OLCF's Center for Accelerated Application Readiness (CAAR) and the general trainings both from OCLF and the vendors. Fernanda Foertter, before moving to NVIDIA recently, was leading several of these hackathon efforts since 2016. She suggested a combination of agile programming and intense mentoring over the course of one week to close this middle ground -- and the first such GPU Hackathon took place in the fall of the same year. Foertter led several of these hackathons since 2016.

The first hackathon featured seven teams that all met in one location for one week to jointly work on their individual applications. The success of the teams as well as the extremely positive and encouraging feedback from the participants, proved that the proposed education format was exactly what was needed to introduce new HPC concepts to experienced programmers. Since then the format has grown both in itself---2017 featured a total of five OCLF supported GPU Hackathons---as well as in similar formats. Other sites, e.g. the Barcelona Supercomputing Center, run similarly structured hackathons and a shorter format---the Mini-Hackathon---has been created as well.

It is a sad truth that not all scientific applications are written such that they can directly benefit from a GPU. Irregular memory access patterns or low arithmetic intensity prevent a full utilization of the features that make a GPU fast. On the other hand, such applications are also suffering on the CPU side since they are not exploiting the modern, performance critical features of a CPU (multi-core, SIMD short vectors). In retrospect, after refactoring an application during a hackathon to run on a GPU, the codes typically also show a dramatic CPU performance boost.

This article explains the training format adopted for the OLCF GPU Hackathon, presents trends observed, and discusses reasons for successes and failures of teams using selective case studies from over 15 hackathons that have taken place so far. It also summarizes the outcome and takeaway for the participants and looks at the Hackathon format from an educator's perspective. Anyone could take this hackathon format and adopt it in their respective institutions.

2. The GPU Hackathon Format

The OLCF hackathon format is centered around very few, but firm principles:

- A team has to sign up with at least three members working on the same code. This is to ensure that there is a broader developer base behind a project.
- The team must come up with the goal they want to reach with the hackathon.
- Their goal must include GPU usage of some kind -- after all, this is what the GPU Hackathons are all about.

Since 2014 the GPU Hackathon Series has expanded to over five OLCF endorsed hackathons a year which are spread throughout the year and around the globe. The GPU Hackathons are formally treated as OLCF training events [16] and are part of OLCF's outreach to create exciting

proposals that use ``Titan"---the veteran GPU computing HPC system at ORNL [23]. Another fixture of the GPU Hackathons is the large pool of mentors that support the teams. Each team will be assigned (at least) two mentors such that they will typically have a domain expert and a GPU programming expert to guide them towards reaching their set goal.

2.1 Proposal Submission

Interested teams can submit their proposed work through an online submission form where they are asked to describe their targeted application, state their goal for the hackathon, and briefly introduce the team members and their levels of expertise. Every Hackathon site has one or two local organizers that lead the review process for the submissions. The reviewers are the pool of local organizers from all Hackathons since they have the experience of how previous, potentially similar submissions worked out. Furthermore, the pool of potential mentors also reviews the proposals to provide feedback on the technical possibilities and science domain relevance. Due to the increased popularity of the GPU Hackathons the acceptance rate has dropped to about 50% in 2017, albeit all sites increasing the number of supported teams to about 10 per site.

Following are some of the strong points that have led to a hackathon proposal to be accepted:

- The code is open source, so that the community impact is high. If teams have no license for the code, they are encouraged to choose an open source license. If the license issue cannot be resolved quickly, the team is encouraged to reapply at a later Hackathon.
- If the task the team is aiming to accomplish has not been accomplished by other software projects available that are already GPU enabled.
- There are teams whose chance of success in achieving their set goal or the impact of the software advancement is judged to be higher. The former is typically the case if some or all team members already have some parallel programming expertise. The latter emphasizes that the Hackathon also works with teams from the ground up, if the impact of a GPU enabled code is very promising for the targeted science domain.

2.2 Preparation

The selected teams are introduced to their mentors about one month prior to the actual Hackathon week. The team and the mentors then discuss a preparation strategy. This includes:

- Receiving access to the compute system(s) to be used during the hackathon week so that the teams can ensure their application runs as it is.
- Online training courses on GPU programming basics (e.g. OpenACC [24] introduction, Compute Unified Device Architecture (CUDA) [25] basics), so that teams have some fundamental understanding of the techniques used during the Hackathon week.
- In case of a GPU port of a non-GPU accelerated application, the teams are encouraged to profile the code so that the hot spots of the code are known.
- The organizers typically also set up a git or subversion (svn) repository so that the teams and their mentors both have access to the source code and a revision system for changes during the week of the Hackathon.

2.3 Hackathon Week

The teams meet up with their mentors in person for one week at the hosting site. They share one table in one room which holds all teams, so that team members can easily communicate with one

another. From a software development point of view the week could be best described as a series of three one-day sprints with a planning day in front and a wrap-up day at the end. Other agile programming techniques were also employed such as the daily scrum sessions where all teams are forced to summarize their current status and strategy. Scrum sessions typically enable the team to optimize their deliverables. A scrum master looks to resolve any impediments or distractions faced by the development team.

While the chair and the mentors can provide a rough guideline as to how fast teams should make progress over the week, the main strength of the Hackathon format is the flexibility for every team to move at their own pace. The mentors can teach required techniques exactly when the team members need them, and the teams apply the learned items directly to their code. Hence, they also have the gratification of improving something they care about instead of yet another Jacobi solver [18]. The Hackathon chair is responsible for keeping teams on track to their goal for the week which can sometimes also lead to the suggestion of abandoning the current strand of work and focus on a new approach.

The constant self-reflection enforced by the scrum sessions makes sure that the teams set themselves rather small work packages which also leads to a more direct feedback on their progress. The measured speed-up compared to the original code is a very good progress indicator, which also leads to a (healthy) competitive atmosphere in the room.

2.4 Post-Hackathon Activities

After the Hackathon week the teams and mentors typically work together remotely for a period of time (approximately 3 months) to tie up the loose ends from the hackathon rush. Furthermore, the local organizers encourage the teams to report about their code improvements at suitable conferences and workshops. Lastly, the pool of Hackathon attendees is at the same time a pool of potential future mentors. This way we would be able to run more and more hackathons in a sustainable manner.

3. Hackathon Trends

Over 60 scientific codes have been ported to heterogeneous platforms via these hackathons.

Team Profile: Hackathons have had teams with senior scientists and expert programmers ---to teams comprising of all students starting to learn to use GPUs ---to teams consisting of a balanced mix of faculty, students and staff. There have also been teams with only domain scientists clearly seeking help from computer scientists and have managed to use GPUs on Day 1.

Collaborative Model: Typically, in a team, the domain scientist explains the code, the programmer breaks down the problem into several pieces and start refactoring the code to adapt to architectures, a tool developer points out if the issue is with the code or the tool itself and a mentor drives the team with proposed solutions and prototypes. This also shows that participants of all kinds are stakeholders in advancing science.

HPC Applications: A variety of applications spanning Astrophysics, Quantum Chemistry, Molecular Dynamics, Biophysics, Computational Fluid Dynamics, Climate modeling and more

have been ported to GPUs in these hackathons. The lines of code of the applications have ranged from 1000 to 100,000. Section 4 highlights some of the cherry-picked case studies.

Programming Languages/Models: The teams typically bring along applications written in C/C++ or Fortran, at an increasing rate also Python and sometimes Kokkos [1] too. Some codes use libraries like Thrust [19]. Sometimes there are applications that are already demonstrating reasonable speedup and show potential to achieve more and sometimes the code is a bare sequential code begging for acceleration on powerful GPUs.

Communication: For fast interactions between and across team members these hackathons have switched to Slack-based communication channel. In such intense hacking setting, there is little room for slow email-based communications. Sometimes not all members of a scientific code can be present at the hackathon due to travel restrictions, for example and the teams need to collectively interact with another that may be in different geographic locations. Using slack has allowed participants to communicate between team members and mentors even after hackathons.

Figure 1 captures the trend – though anectodal and via surveys - of a variety of emotions that the participants go through during the 5-day hackathon time period. Early in the week, the teams are typically eager to get started and then run into all sorts of problems such as their sequential code not even compiling or executing and if it does, they show inaccurate results. After the middle of the week, teams are hopeful and enlightened to learn that the problems have a solution and are already running their codes on GPUs. Towards the end of the week, the teams retrospect different lessons learned in that week and share the outcome with each other further elaborated in Section 6. Teams either leave quite delighted with their progress or leave with at the very least the next set of possibilities to explore.

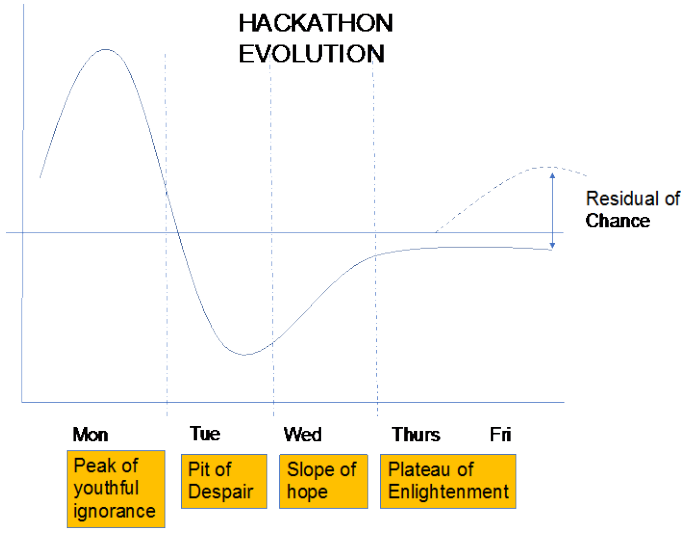

Figure 1. The various stages the hackathon participants find themselves in over the course of the week

Definition of Success: We observe different types of success stories:

1. Application speedup: In terms of speedup we observe teams and codes achieved preliminary speedup when compared to a serial code or to an OpenMP code on multicore.
2. Knowledge gain: Teams quickly learn to use GPUs using OpenACC and are determined to continue to take the effort to move code to GPUs after leaving the hackathon event.
3. Publications: Students and advisors are motivated to publish their work and present their work at conferences thus imparting knowledge on how to program GPUs without struggling to learn CUDA. Some of the publications resulted out of hackathon efforts include [34-42].

One team, NekCEM (Nekton for Computational ElectroMagnetics) [30], from one of the first hackathons even won a 2016 R&D100 award (https://www.rd100conference.com/awards/winners-finalists/6546/nekcemnek5000-scalable-high-order-simulation-codes/) as well as an ECP award (https://www.exascaleproject.org/project/ceed-center-efficient-exascale-discretizations/).

4. Case Studies

This section will highlight some of the case studies out of 60 applications ported to large scale systems via these hackathons. Each case study is a short story from the local hackathon organizer or the team lead of the application discussing the effort, challenges and outcome.

Computing Platforms Used

Participants in these hackathons have used large heterogeneous clusters such as TITAN (K20x) and Summit-Dev (NVIDIA Pascal100) at ORNL, a new 108 compute node (K80) GPU-accelerated institutional cluster at Brookhaven National Laboratory (BNL) among others. Hackathons in Europe were held at the Swiss National Supercomputing Centre (CSCS, Switzerland), Technische Universtat Dresden (TUD, Germany) and Jülich Supercomputing Centre (JSC, Germany). These hackathons are popularly termed as Eurohacks. Teams at Eurohacks access clusters at JSC, CSCS, and TUD. JSC clusters including JURECA cluster [20] with nodes consisting of K80 GPU and Haswell processors as well as the OpenPOWER JURON system [21] with nodes consisting of IBM POWER8 processors and NVLink attached P100 GPUs. CSCS includes machines such as Piz Daint [43] that consist of Cray nodes with NVIDIA Tesla K20x and SandyBridge CPUs. The TUD cluster TAURUS [45] is similar in setup to the JURECA cluster from JSC. The software harness of these machines includes a number of compiler tools, profilers, performance analysis tools and other scientific libraries.

4.1 AstroGPU: Astrophysics

The BNL hackathon hosted a team that comprised of present and past students (at different levels of GPU coding experience) from Stony Brook University (SBU). The team used the structured adaptive mesh refinement library AMReX [4], applicable to the astrophysics codes Maestro [5] and Castro [6]. By stripping down version of Castro from 100K lines of code to only 8K lines of code, StarLord [29], the team found it much easier to explore different ideas. StarLord and AMReX use C++ for the memory management, parallelism, and drivers and Fortran for the compute kernels. The choice was made to use CUDA Fortran for this work, since it seemed to work well with both new and old Fortran codes.

Prior to the hackathon, the team spent sufficient time cleaning up interfaces in StarLord and

splitting some kernels into multiple pieces to allow them to start offloading the work piece by piece to the device. Over the course of the hackathon week, the team achieved about 20x faster on a single GPU compared to a single CPU core on SummitDev---the precursor for the ``Summit'' system recently unveiled at ORNL.

4.2 JuRor: Real Time Fire Propagation Simulation

The JuRor team is part of a German research project called ``ORPHEUS'' [28] which targets improved safety in underground facilities in case of fire. One part of the project is the real-time simulation of the propagation of the fire in the underground structure to guide fire fighters to where they are needed most. The simulation was compressed to a miniapp of about 2,500 lines of C++11 code. The weakly compressible Navier-Stokes equations are numerically solved using finite differences in a fractional step method using an orthogonal projection. The time marching scheme is split into the three major steps: advection, diffusion and pressure calculation. The advection equation is calculated using a semi-Lagrangian scheme, the diffusion equation solver is based on an implicit Jacobi scheme, and the pressure equation is handled by a (geometric) multi-grid method. The team used the week to add OpenACC directives to the code in the most computationally intense parts early on and spend most time in optimizing data movement, results published in [3]. JuROr remains one of the codes of the new institute for Civil Safety Research at the Research Lab in Jülich [44].

4.3 PALM4GPU: Simulating Turbulent Flows

PALM is a finite-difference large-eddy simulation (LES) model to study turbulent atmospheric and oceanic flow [17]. It is used both for basic research (e.g. turbulent transport above heterogeneous terrain, effect of turbulence on the growth of cloud droplets) and applied research questions (e.g. flow within and behind wind farms, effects of turbulence on airplanes during takeoff and landing, turbulent wind fields in cities). The flow model is coupled with a Lagrangian particle model (LPM). PALM consists of about 100,000 lines of Fortran 95-based code with some Fortran 2003 extensions and external libraries such as NetCDF [26] and FFTW [27] are used. Parallelization is based on two-dimensional domain decomposition using MPI and hybrid parallelization with MPI and OpenMP is also available. Even though the goal seemed realistic, over the course of the hackathon the team struggled heavily with this legacy code. The nature of the ``organically'' grown code base---a result of multiple students' theses plus grant-projects---caught up during the refactoring for adding OpenACC directives. The team struggled with incorrect results that have lain dormant in the code and for a lack of a constant code maintainer have stopped the effort after the hackathon [2]. Nevertheless, PALM is still widely used, albeit, only in a hybrid MPI+OpenMP setup.

4.4 Kinetic Model Builder

A team comprising of Ph.D. and Master's students from the Chemical and Biomolecular Engineering, University of Delaware (UDEL), with no prior GPU programming experience learned a great deal by participating in the hackathon. They learned to use profilers that indicated that LU decomposition within their Ordinary Differential Equations (ODE) solver code, if run on GPU could give them performance gains on large systems. The team although did not achieve a noticeable speedup while at the hackathon, they continued to build their code on the university's local cluster post the hackathon and witnessed great performance gains on a GPU. The LU decomposition workload of an explicit ODE solver was transferred to the GPU. In a complex

molecular system of 904 species, for which LU decomposition is performed on a 904x904 matrix at every time step within the ODE solvers algorithm (~5000 time steps), the GPU ODE solver implementation (after profiling) took 5.9s compared to 23.2 seconds on the CPU and also demoed better scalability for larger matrix sizes. These results indicate that the solvers will scale even better on larger systems.

4.5 NekCEM: Higher Order Computational Electromagnetics

NekCEM is a spectral element discontinuous Galerkin solver for electromagnetics featuring geometric flexibility, spectrally accurate numerical convergence, with an efficient parallel MPI and, with the help of the Hackathons, an OpenACC implementation. Prior to the hackathon, the NekCEM code had limited OpenACC and GPU support, all implemented by a former student. The hackathon was a chance for many members of the NekCEM team to quickly attain the knowledge needed to implement and optimize OpenACC code. Though there are resources available explaining OpenACC and how to efficiently use it (including examples), they do not cover the myriad of compiler bugs and OpenACC bugs. Many of these can be solved with enough searching, but it is quite discouraging to make only minor changes and then spend 30 minutes to find a solution. Working off of the knowledge of the mentors and experts at the hackathon, these problems can quickly be solved, allowing for the implementation to continue unimpeded. For the NekCEM team, the Hackathon was key to getting an efficient OpenACC version of our Maxwell solver, resulting in more than a 2.5x speedup (compared to 16 MPI ranks) on over 16k nodes of Titan [7]. The GPU performance was used as a part of their successful R&D100 application.

4.6 CoMD: Molecular Dynamics (MD) Exascale Proxy App

The BNL hackathon hosted a team that included a diverse group of students spanning computational biology and computer science backgrounds from UDEL and UHouston, with a key team member participating solely remotely. The team's application code was CoMD, a molecular dynamics exascale proxy application developed by the DoE Exascale Co-Design Center for Materials in Extreme Environments [8]. The team's two major goals were to create and evaluate a GPU-accelerated version of CoMD using OpenACC, and to investigate extending the CoMD implementation to incorporate support for Particle Mesh Ewald (PME) [31] long-range electrostatics. The original CoMD application was written in C, with some existing adaptations for MPI, OpenMP, and CUDA. Since the internals of CoMD were unfamiliar to several team members, and there was a general disparity in individual programming expertise, this presented an early challenge that led to a bifurcation of the team into a subgroup focused on OpenACC GPU acceleration of CoMD Lennard-Jones (L-J) force calculation loops, and a subgroup focused on adaptation of CoMD to support PME.

Over the course of the hackathon, the speedup obtained for the L-J force kernels was improved substantially, through careful use of directives to cause parallel reductions to be done increasingly efficiently without serialization bottlenecks in complex loop nests, giving an overall GPU speedup of 2x vs. OpenMP on the host CPUs, a result that was also observed in small scale MPI runs. The other subgroup focused on adaptation of CoMD to implement PME faced challenges associated with both algorithmic adaptation issues as well as refactoring of the existing library interfaces for the PME implementation they chose to use. Over the course of the Hackathon, the PME subgroup narrowed the scope of their activities to match their time

constraints, focusing on adaptation of key data structures for efficient access on the GPU, minimizing host-GPU data transfers, and adapting individual loop nests for OpenACC.

Overall, the team as a whole had an excellent real-world experience working with third-party source code. They had to ascertain which were the key program data structures, which routines were performance critical, and they had to make important decisions about when to rewrite existing code rather than just annotating it with directives. The team also, gained hands-on experience with multiple tools for profiling and performance analysis, debugging, revision control, compilers, and team messaging and collaboration.

4.7 VMD: Visual Molecular Dynamics

The National Center for Supercomputing applications (NCSA) hackathon in 2015 encouraged teams to submit projects that would be appropriate for GPU acceleration using the early OpenACC tools available at that time. A team of biomolecular modeling domain scientists from UIllinois proposed a project to use OpenACC for GPU acceleration of existing MD analysis algorithms in the molecular visualization and analysis package VMD [9]. The algorithms of interest were originally implemented in a mixture of C++, Tcl, and Python.

The selected algorithms were known to present some implementation challenges for OpenACC acceleration due to a variety of complex loop nests and parallel reduction patterns, while involving relatively few arithmetic operations per memory reference. The team's use of the early OpenACC toolchain exposed a variety of unexpected compatibility problems between the nascent OpenACC runtime implementation and applications that already contained code that initialized and used CUDA on the GPUs. The team had written a straightforward C/C++ based serial implementations of their target algorithms prior to the start of the hackathon, this made adding OpenACC to the key subroutines of interest easier while at the hackathon. The team discovered that significant kernel launch and data transfer overheads in the early OpenACC runtime implementation were preventing them from achieving their performance goals, so the team changed track and re-implemented the kernels a second time at a lower level using CUDA, gaining performance limited primarily by PCIe host-device transfer bandwidth and GPU global memory bandwidths.

By the end of the hackathon the team had become familiar with both OpenACC and CUDA programming, and the kernels they developed were structured well enough to become useful to the molecular modeling community with a minor amount of additional work. A little over a year after the end of the hackathon, more robust and polished versions of the CUDA versions of the kernels that the team developed were integrated into VMD.

4.8 QUDA: Lattice QCD Library

Simulating the theory of strong interactions, which is called Quantum Chromodynamics (QCD), requires significant compute resources. Scientists from this area have always been keen to adopting compute devices providing exceptional compute performance. GPUs started to be used for accelerating the performance critical kernel i.e. linear solver, well before CUDA or directive-based GPU programming models became available (see, e.g., [10]). As more efficient yet complex solvers become available and as GPUs-based systems continue to evolve, the efforts of implementing such solvers start to become heavy for individual research groups and therefore

community efforts to provide shareable implementations are becoming more important. One such effort goes under the name QUDA [11]. During the hackathon at Jülich Supercomputing Centre in 2017, two different teams of QCD researchers worked on augmenting QUDA. The ``Quantum Tornados" team collected researchers both from the US and Europe to integrate a new multigrid-type solver, namely DDαAMG [12]. During the hackathon the team accomplished the adaptation of the DDαAMG algorithm into QUDA. At the last day the team reported significant improvement in the number of solver steps compared to a multigrid solver that was already available in QUDA. Routines integration required implementation of missing routines and changes in the library interface. This code was also running on multiple GPUs concurrently. The new capabilities of QUDA implemented during the hackathons are actively exploited by researchers in the field of Lattice QCD for large scale simulations, including simulations on Piz Daint, which is currently the largest supercomputer in Europe.

4.9 Asynchronator: Asynchronous Linear Solver

One of the teams, which attended the hackathon at the Jülich Supercomputing Centre (JSC), focused on GPU-enabling a new solver that belongs to an emerging class of highly scalable, fault tolerant methods that are based on asynchronous iterations (see, e.g., [13]) Asynchronicity can help to significantly improve scalability. The team comprised of PhD students from HPC-LEAP (http://www.hpc-leap.eu/). The team systematically explored different technologies like Thrust, CUB [32] or cuBLAS [33]. After fixing issues with compilers, the team compared the performance achieved on 2 CPUs to the performance obtained when adding one GPU. They reported performance improvements for selected kernels in the range of 2.6 to 7.3 (for comparison: by adding the GPU the nominal floating-point performance and memory bandwidth is increased by a factor 11 and 3, respectively). This code is an early state but continues to be actively developed. The code is developed with (performance) portability in mind. The GPU port continues to be maintained.

4.10 NASA Langley HPC Incubator CNDE team

The Computational Nondestructive Evaluation (CNDE) team was focused on porting an elastodynamic finite integration technique code to GPUs. The code calculates nine variables on a staggered grid. The starting point of the code was in C++ and was a code version optimized to run on Knights Landing cores. The initial goals for the hackathon were to get the code running on GPUs and to achieve speed increases by running on GPU. The code was successfully ported to GPUs and was tested for a grid size of 1024 x 1024 x 16 and run for 4096 time steps. The initial plan was to use only OpenACC, however the OpenACC version showed a 3x slowdown (30 ms runtime) compared to running on KNL (15 ms runtime). During the hackathon it was determined that improved performance was achieved by using CUDA for more complex programming options (compared to OpenACC). It was also surmised that starting with a more simplified kernel than the code that had been optimized for MIC hardware may have led to better results on the GPU hardware.

5. Hackathon Outcomes and Takeaway

This section discusses some of the outcomes and takeaways:

Outcome:

- Large computing facilities are making their systems available to scientists and programmers thus building a large customer pool.
- Compiler bugs are being identified in software tools and due to the on-site participation of key vendors and developers, these errors are immediately flagged for correction.
- The gap between tool developers and its users are narrowed down. Teams benefit from closer proximity to expert mentors.
- Hardware and software are stress tested by varieties of scientific code bases.
- Next-generation workforce is being trained to use parallel machines and work in a team-based setting.
- Collaborations are established between teams from national labs, academia and organizations leading to innovative ideas.
- Mentors are being created and developed.
- Students gain so much experience and knowledge by participating in these hands-on hackathon event that they sometimes tend to tell the faculty what to do thus creating an 'inverted hierarchy', which further strengthens collaborative effort.
- Knowledge is gained outside of an individual's comfort zone while learning how people work together in teams to achieve a common goal or purpose.

Takeaways:

- Profile the code, if possible ahead of time, to understand performance bottlenecks; various profilers exist such as TAU, Score-P
- Use a local machine if you have one to compile and execute your code where you have full control; this can be very helpful for preliminary experiments
- Start with a smaller kernel in the large code base or start with a miniapp. For example, a team at the BNL hackathon created a stripped-down version of Castro [14] called StarLord that was only 8k lines of code instead of 100k lines of code. The smaller code base made it much easier to perform several experiments.
- Identify a good starting point before applying parallelization concepts. A code efficiently parallelized for X86 may not necessarily be a good starting point for GPUs. So a sequential version could be a better starting point.
- Debugging can be a challenge, especially when you get CUDA kernel errors. Sometimes the simplest techniques work the best. One of the teams adopted the motto 'when in doubt, comment it out'.

6 Hackathon From Educator's Perspective

Observing students' performance in various hackathons point out that there is a huge gap between the strong needs of quality programmers in national labs, NSA, NASA and industries and the lack of relevant and sufficient education in computer science graduate and undergraduate programs. A course on parallel programming as part of the undergraduate or the graduate curriculum in universities/institutes is becoming increasingly important. This would prepare the next-generation workforce to be able to think parallel and use parallel platforms (which even includes a simple laptop that nowadays comes with at the least 2-4 cores plus an integrated graphics card). UDEL since 2016, has made parallel programming a required course for students graduating with a CS undergraduate degree. Similarly, University of Edinburgh has made this

course a required one to receive an MSc degree in HPC. Similarly, ETH, Zurich for BSc degree and TU Darmstadt too. Indiana University sprinkles concepts of parallel programming as part of the Intelligent systems engineering degree. After such a general introduction to the concepts of parallel programming, a GPU hackathon could serve as an intense practical lab session for applying the learned principles to one specific platform.

In additional to parallel programming, there are other important tools that the students need to learn to use and these may not be always taught as part of a curriculum. Software carpentry (https://software-carpentry.org/about/) offers a number of courses that could be leveraged by instructors and students. Courses cover program design, version control, testing, and task automation. Software development is not easy for any domain scientists, as also mentioned in the practical book by Scopatz and Huff [14] that focuses on how to use Python to help collect, analyze, build and publish results for research in physics-based field. Some of the fundamental scientific and technical computing tools that the students should know include Makefile, Python, CMake, BASH, linking, debuggers, Software Licensing, Distribution; Programming Languages (at least one) and Mathematical libraries for linear algebra, Version control. More importantly, these topics should be supported with hands-on training.

A model that supports such hands-on training-based course is Georgia Tech's Vertically Integrated Program (VIP) [22]. The author, Chandrasekaran, from UDEL runs a multi-year VIP-HPC program for the undergraduate students in her university, UDEL [15]. The VIP program enables students to do research with faculty for an extended period of 3 years. Unlike a typical classroom and lecture-based course, this VIP course almost functions like a hackathon. The students work in teams. Each team is assigned a project. Students are given access to dedicated clusters consisting of state-of-the-art hardware, scientific computing tools and compilers. The students detail on their approaches, thoughts, proposals, action items and summaries of their projects into a physical VIP notebook provided to them. The class meets once a week. Students communicate with their peers and instructors via slack communication channel. The students are also paired up with senior research students (Masters or Ph.D. students) thus creating a 'learn from mentor' environment and that seems to facilitate faster and easier progress on projects.

They use Git version control system to commit their codes and at the end of each semester. Each team presents posters and demos their project to faculty from other department, technical managers and recruiters from industries in a poster showcase event. The students' projects are reviewed by peers and faculty. The students receive credits (typically 1 for sophomore and 2 for junior and senior) but this varies from person to person depending on his/her project quality, their aptitude, applicability and approachability to a given problem.

Such a practical training-based model or a similar model could be adopted by universities to encourage students to allocate specific time to apply theoretical knowledge to a real problem and collect credits towards their degree program. This model does not increase the time taken to degree, instead builds a strong foundation at the undergraduate level for students to be aware and knowledgeable of key topics critical in the field of HPC. Such a long-term problem-based learning environment could also be considered instead of capstone course where the students get very limited time (1-2 semesters) to explore and build a complete, functional, high-quality open-source product.

7. Summary and Outlook

The ORNL-led GPU Hackathons have proved to be quite effective in helping the domain scientists jump start their initial GPU porting efforts or getting them on track to further optimize their codes. In this regard, this form of HPC training should be considered for the major computing facilities where the users are expected to use the computing resources efficiently. On the other hand, the Hackathon format could also be generalized to help train the next-generation HPC users, especially the students enrolled in computer science or scientific computing courses, or early-career scientific researchers. One could imagine that as part of the course work, term projects could be given to the students in a one or two-day ``mini-hackathon'' setting, provided sufficient preparation similar to the Hackathon preparation discussed earlier in this article is given. The most crucial point from an education's perspective is offer more hands-on training to students.

Acknowledgements

These hackathons would not have been successful without the help of some very important players: Duncan Poole, Pat Brooks, Julia Levites, Mat Colgrove, Michael Wolfe, Brent Leback, Robert Searles, Kyle Friedline, Andy Novocin among others. Special thanks to Dana Hammond, Cara Campbell Leckey, Elizabeth Gregory and William Schneck of NASA for contributing case studies from their hackathon. J.E. Stone acknowledges support from NIH grant 9P41GM104601.

Biography Text of Corresponding Authors

Sunita Chandrasekaran is an Assistant Professor in Dept. of Computer & Information Sciences at the University of Delaware. Her research areas include exploring applicability of high-level programming models and its language extensions to large scale scientific applications and building validating and verifying testsuite for compilers used for acceptance testing on widely used supercomputers. She is an active member of the OpenMP, OpenACC, and SPEC HPG communities. Dr. Chandrasekaran earned her PhD in computer science engineering from Nanyang Technological University, Singapore, for creating a high-level software stack for FPGAs. Dr. Chandrasekaran co-edited a textbook on OpenACC for Programmers: Concepts and Strategies published in November 2017.

Guido Juckeland is the Head of Computational Science Group at Helmholtz-Zentrum Dresden-Rossendorf (HZDR), Germany. His role includes designing and implementing end-to-end research IT-workflows together with scientists and IT experts at HZDR. His research focuses on better usability and programmability for hardware accelerators and application performance monitoring as well as optimization. He is the vice-chair of the SPEC High Performance Group (HPG), the Secretary of the OpenACC standard organization, and also contributes to the OpenMP tools working group. Dr. Juckeland earned his PhD in computer science from Technische Universität Dresden, Germany, for his work on trace-based performance analysis for hardware accelerators. Dr. Juckeland co-edited a textbook on OpenACC for Programmers: Concepts and Strategies published in November 2017.

References

[1] H. C. Edwards, C. R. Trott, and D. Sunderland, “Kokkos: Enabling manycore performance portability through polymorphic memory access patterns,” Journal of Parallel and Distributed Computing, vol. 74, no. 12, pp. 3202–3216, 2014.
[2] H. Knoop, T. Gronemeier, C. Knigge, and P. Steinbach, Porting the MPI Parallelized LES Model PALM to Multi- GPU Systems – An Experience Report. Cham: Springer International Publishing, 2016, pp. 508–523. [Online]. Available: https://doi.org/10.1007/978-3-319-46079-6 35
[3] A.Ksters, S.Wienke, and L.Arnold,“Performance portability analysis for real-time simulations of smoke propagation using OpenACC,” in Proceedings of the P3MA at ISC, currently being published with Springer, 2017
[4] New block-structured AMR framework (amrex), https:// github.com/AMReX-Codes/amrex, 2016.
[5] A. Nonaka, A. Almgren, J. Bell, M. Lijewski, C. Malone, and M. Zingale, “Maestro: An adaptive low mach number hydrodynamics algorithm for stellar flows,” The Astrophysical Journal Supplement Series, vol. 188, no. 2, p. 358, 2010.
[6] A. Almgren, V. Beckner, J. Bell, M. Day, L. Howell, C. Jog- gerst, M. Lijewski, A. Nonaka, M. Singer, and M. Zingale, “Castro: A new compressible astrophysical solver. i. hydro-dynamics and self-gravity,” The Astrophysical Journal, vol. 715, no. 2, p. 1221, 2010.
[7] M. Otten, J. Gong, A. Mametjanov, A. Vose, J. Levesque, P. Fischer, and M. Min, “An mpi/openacc implementation of a high-order electromagnetics solver with gpudirect communication,” The International Journal of High Performance Computing Applications, vol. 30, no. 3, pp. 320–334, 2016.
[8] J. Mohd-Yusof and N. Sakharnykh, “Optimizing CoMD: A molecular dynamics proxy application study,” in GPU Technology Conference (GTC), 2014.
[9] W. Humphrey, A. Dalke, and K. Schulten, “VMD – Visual Molecular Dynamics,” J. Molecular Graphics, vol. 14, no. 1, pp. 33–38, 1996.
[10] G. I. Egri, Z. Fodor, C. Hoelbling, S. D. Katz, D. Nogradi, and K. K. Szabo, “Lattice QCD as a video game,” Computer Physics Communications, vol. 177, no. 8, pp. 631 – 639, 2007. [Online]. Available: http://www.sciencedirect. com/science/article/pii/S0010465507003025
[11] M. A. Clark, R. Babich, K. Barros, R. C. Brower, and C. Rebbi, “Solving Lattice QCD systems of equations using mixed precision solvers on GPUs,” Comput. Phys. Commun., vol. 181, pp. 1517–1528, 2010.
[12] A. Frommer, K. Kahl, S. Krieg, B. Leder, and M. Rottmann, “Adaptive Aggregation Based Domain Decomposition Multi- grid for the Lattice Wilson Dirac Operator,” SIAM J. Sci. Comput., vol. 36, pp. A1581–A1608, 2014.
[13] A. Frommer and D. B. Szyld, “On asynchronous iterations,” Journal of Computational and Applied Mathematics, vol. 123, no. 1, pp. 201 – 216, 2000, numerical Analysis 2000. Vol. III: Linear Algebra. [Online].
[14] A. Scopatz and K. D. Huff, Effective Computation in Physics, 1st ed. O’Reilly Media, May 2015.
[15] U. of Delaware, “Vertically integrated program - high performance computing,” http://vip.udel.edu/hpc, 2017.
[16] ORNL GPU Hackathons https://www.olcf.ornl.gov/calendar/2018-gpu-hackathons/
[17] PALM model system http://palm.muk.uni-hannover.de
[18] Jacobi Method http://mathworld.wolfram.com/JacobiMethod.html
[19] Thurst library https://thrust.github.io/
[20] JURECA http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JURECA/JURECA_node.html
[21] JURON https://hbp-hpc-platform.fz-juelich.de/?page_id=1073
[22] Georgia Tech VIP Program http://www.vip.gatech.edu/

[23] https://www.olcf.ornl.gov/olcf-resources/compute-systems/titan/
[24] OpenACC Specification https://www.openacc.org/
[25] CUDA https://developer.nvidia.com/about-cuda
[26] NetCDF https://www.unidata.ucar.edu/software/netcdf/
[27] FFTW http://www.fftw.org/
[28] ORPHEUS http://www.fz-juelich.de/orpheus/DE/Home/home_node.html
[29] StarLord https://github.com/AMReX-Astro/StarLord
[30] NekCEM http://www.mcs.anl.gov/project/nekcem-nekton-computational-electromagnetics
[31] Darden, T., York, D., & Pedersen, L. Particle mesh Ewald: An N·log (N) method for Ewald sums in large systems. The Journal of chemical physics, 98(12), 10089-10092.
[32] CUB https://nvlabs.github.io/cub/
[33] cuBLAS https://developer.nvidia.com/cublas
[34] Clark, M. A., Strelchenko, A., Vaquero, A., Wagner, M., & Weinberg, E. Pushing Memory Bandwidth Limitations Through Efficient Implementations of Block-Krylov Space Solvers on GPUs. arXiv preprint arXiv:1710.09745, 2017
[35] Boyle, P. A., Clark, M. A., DeTar, C., Lin, M., Rana, V., & Avilés-Casco, A. V. Performance Portability Strategies for Grid C++ Expression Templates. In EPJ Web of Conferences (Vol. 175, p. 09006). EDP Sciences, 2018.
[36] Lukas Mazur, Master's thesis "Applications of the Gradient Flow method in Lattice QCD" Department of High Energy Physics, Bielefeld University, Bielefeld, Germany, 2017
[37] Küsters, A., Wienke, S., & Arnold, L. Performance Portability Analysis for Real-Time Simulations of Smoke Propagation Using OpenACC. In International Conference on High Performance Computing (pp. 477-495). Springer, Cham, 2017
[38] Knoop, H., Gronemeier, T., Knigge, C., & Steinbach, P. Porting the MPI Parallelized LES Model PALM to Multi-GPU Systems–An Experience Report. In International Conference on High Performance Computing (pp. 508-523). Springer, Cham, 2016.
[39] Zingale, M., Almgren, A. S., Sazo, M. G., Beckner, V. E., Bell, J. B., Friesen, B., ... & Willcox, D. E. Meeting the Challenges of Modeling Astrophysical Thermonuclear Explosions: Castro, Maestro, and the AMReX Astrophysics Suite. arXiv preprint arXiv:1711.06203. 2017
[40] Tazzari, M., Beaujean, F., & Testi, L. GALARIO: a GPU accelerated library for analysing radio interferometer observations. Monthly Notices of the Royal Astronomical Society, 476(4), 4527-4542, 2018
[41] Clark, M. A., Strelchenko, A., Vaquero, A., Wagner, M., & Weinberg, E. Pushing Memory Bandwidth Limitations Through Efficient Implementations of Block-Krylov Space Solvers on GPUs. arXiv preprint arXiv:1710.09745, 2017
[42] Tazzari, M., Beaujean, F., & Testi, L. GALARIO: a GPU accelerated library for analysing radio interferometer observations. Monthly Notices of the Royal Astronomical Society, 476(4), 4527-4542, 2018
[43] Piz Daint https://www.cscs.ch/computers/piz-daint/
[44] Real-Time Predictions on GPUs http://www.fz-juelich.de/ias/jsc/EN/Research/ModellingSimulation/CivilSecurityTraffic/FireDynamics/ResearchTopics/RealTime_Predictions/_node.html
[45] System TAURUS https://doc.zih.tu-dresden.de/hpc-wiki/bin/view/Compendium/SystemTaurus