# From Piz Daint to the Stars: Simulation of Stellar Mergers using High-Level Abstractions

Gregor Daiß
IPVS, University of Stuttgart
Gregor.Daiss@ipvs.uni-stuttgart.de

Parsa Amini*
CCT, Louisiana State University
parsa@cct.lsu.edu

John Biddiscombe*
Swiss National Supercomputing
Centre
biddisco@cscs.ch

Patrick Diehl*
CCT, Louisiana State University
pdiehl@cct.lsu.edu

Juhan Frank
Louisiana State University
frank@phys.lsu.edu

Kevin Huck
University of Oregon
khuck@cs.uoregon.edu

Hartmut Kaiser*
CCT, Louisiana State University
hkaiser@cct.lsu.edu

Dominic Marcello*
CCT, Louisiana State University
dmarcello@phys.lsu.edu

David Pfander
IPVS, University of Stuttgart
David.Pfander@ipvs.uni-stuttgart.de

Dirk Pflüger
IPVS, University of Stuttgart
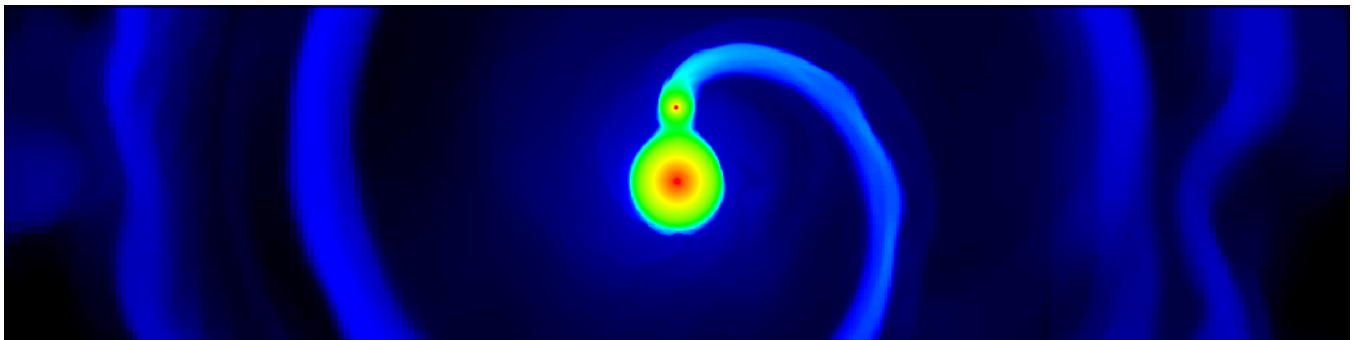dirk.pflueger@ipvs.uni-stuttgart.de

**Figure 1: The Octo-Tiger model of V1309 Scorpii 20 orbits after the simulation begins. V1309 Scorpii is a contact binary that merged into a single star in 2008 in a process known as a luminous red nova. It was the first star to provide conclusive evidence that contact binary systems end their evolution in a stellar merger [58], see also Section 3.**

## ABSTRACT

We study the simulation of stellar mergers, which requires complex simulations with high computational demands. We have developed Octo-Tiger, a finite volume grid-based hydrodynamics simulation code with Adaptive Mesh Refinement which is unique in conserving both linear and angular momentum to machine precision. To face the challenge of increasingly complex, diverse, and heterogeneous HPC systems, Octo-Tiger relies on high-level programming abstractions.

We use HPX with its futurization capabilities to ensure scalability both between nodes and within, and present first results replacing MPI with libfabric achieving up to a 2.8x speedup. We extend Octo-Tiger to heterogeneous GPU-accelerated supercomputers, demonstrating node-level performance and portability. We show scalability up to full system runs on Piz Daint. For the scenario's maximum resolution, the compute-critical parts (hydrodynamics and gravity) achieve 68.1% parallel efficiency at 2048 nodes.

*The STE||AR Group, stellar-group.org

## CCS CONCEPTS

• **Applied computing** → **Astronomy**; • **Computer systems organization** → **Heterogeneous (hybrid) systems**; • **Software and its engineering** → *Ultra-large-scale systems*.

## KEYWORDS

binary star merger, high-level abstractions, accelerators, libfabric, HPX, asynchronous, futures

## 1  INTRODUCTION

Astrophysical simulations are among the classical drivers for exascale computing. They require multiple scales of physics and cover vast scales in space and time. Even the next generation of high-performance computing (HPC) systems will be insufficient to solve more than a fraction of the many conceivable scenarios.

However, new HPC systems come not only with ever larger processor counts, but increasingly complex, diverse, and heterogeneous hardware. This raises challenges especially for large-scale HPC simulation codes and requires going beyond traditional programming models. High-level abstractions are required to ensure that codes are portable and can be run on current HPC systems without the need to rewrite large portions of the code.

We consider the simulation of stellar phenomena based on the simulation framework Octo-Tiger. In particular, we study the simulation of time-evolving stellar mergers (Fig. 1). The study of binary star evolution from the onset of mass transfer to merger can provide fundamental insight into the underlying physics. In 2008, this phenomenon was observed with photometric data, when the contact binary V1309 Scorpii merged to form a luminous red novae [58]. The vision of our work is to model this event with simulations on HPC systems. Comparing the results of our simulations with the observations will enable us to validate the model and to improve our understanding of the physical processes involved.

Octo-Tiger is an HPC application and relies on high-level abstractions, in particular, HPX and Vc. While HPX provides scheduling and scalability, both between nodes and within, Vc ensures portable vectorization across processor-based platforms. To make use of GPUs we use HPX's CUDA integration in this work.

In previous work, we have demonstrated scalability on Cori, a Cray XC40 system installed at the National Energy Research Scientific Computing Center (NERSC) [27]. However, the underlying node-level performance was rather low, and we were only able to simulate for few time steps. Consequently, we had started to study node-level performance, achieving 408 GFLOPS on the 64 cores of the Intel Knights Landing manycore processor [45]. Using the same high-level abstractions as on multicore systems, this led to a speedup of 2 compared to a 24-core Intel Skylake-SP platform.

In this work, we make use of the same CPU level abstraction library Vc [31] for SIMD vector parallelism as in the previous study, but extend Octo-Tiger to support GPU-based HPC machines. We show how the critical node-level bottleneck, the fast multipole method (FMM) kernels, can be mapped to GPUs. Our approach utilizes GPUs as co-processors, running up to 128 FMM kernels on each one simultaneously. This was implemented using CUDA streams and uses HPX's futurization approach for lock-free, low-overhead scheduling. We demonstrate the performance-portability of Octo-Tiger for a set of GPU and processor-based HPC nodes.

To scale more efficiently to thousands of nodes, we have integrated a new libfabric communication backend into HPX where it can be used transparently by Octo-Tiger – the first large scientific application to use the new network layer. The libfabric implementation extensively uses one-sided communication to reduce the overhead compared to a standard two-sided MPI-based backend. To demonstrate both our node-level GPU capabilities as well as our improved scalability with libfabric, we show results for full-scale runs on Piz Daint running the real-world stellar merger scenario of V1309 Scorpii for a few time-steps. Piz Daint is a Cray XC40/XC50 equipped with NVIDIA's P100 GPUs at the Swiss National Supercomputing Centre (CSCS). For our full system runs we used up to 5400 out of 5704 nodes. This is the first time an HPX application was run on a full system of a GPU-accelerated supercomputer.

In Sec. 2 we briefly discuss related approaches. We describe the stellar scenario in more detail in Sec. 3, the important parts of the overall software framework and the high-level abstractions they provide in Sec. 4. In turn, Sec. 5 shows the main contributions of this work, describing both the new libfabric parcelport and the way we utilize GPUs to accelerate the execution of critical kernels. In Sec. 6.1, we present our node-level performance results for NVIDIA GPUs, Intel Xeons and an Intel Xeon Phi platform. Section 6.2 describes our scaling results, showing that we are able to scale with both an MPI communication backend and a libfabric communication backend of HPX. We show that the use of libfabric strongly improves performance at scale.

## 2  RELATED WORK

There are several studies that investigate the structure of mass loss in V1309 Scorpii through computer simulation. One approach to modeling this system is smoothed-particle hydrodynamics (SPH). Notable SPH applications include StarSmasher [1, 2] (a fork of StarCrash [21]) and an unpublished code developed by a collaboration of researchers from Princeton University, Columbia University, and Osaka University [43, 44]. An alternative approach is to use the finite volume method to simulate mass transfer. Examples of such applications include Athena [53, 55] and its rewrite named Athena++ [34, 35, 54]. Lastly, Enzo [10] is a project that implements finite volume hydrodynamics along with a collisionless N-body module that can be used to simulate binary systems where one component is taken to be a point mass. With the exception of SPH codes using direct summation for

gravity, Octo-Tiger is unique among three-dimensional self-gravitating hydrodynamics codes in that it simultaneously conserves both linear and angular momentum to machine precision. SPH codes using direct summation for gravity are limited to only a few thousand particles, making Octo-Tiger the better choice for high resolution simulations.

Adaptive multithreading systems such as HPX expose concurrency by using user-level threads. Some other notable solutions that take such an approach are Uintah [22], Chapel [11], Charm++ [30], Kokkos [19], Legion [6], and PaRSEC [9]. Note that we only refer to distributed memory capable solutions, since we focus here on large distributed simulations. Different task-based parallel programming models, e.g. Cilk Plus, OpenMP, Intel TBB, Qthreads, StarPU, GASPI, Chapel, Charm++, and HPX, are compared in [57]. Our requirements (distributed, task-based, asynchronous) are met by few, out of which HPX has the highest technology readiness level according to this review. It is furthermore the only one with a future-proof C++ standard conforming API and allows us to support the libfabric networking library without changing application code. For more details, see Sec. 4.1.

There are several particle-based FMM implementations utilizing task-based programming available. The approach described in [33] uses the Quark runtime environment [60], the implementation in [3, 4] uses StarPu [5], whilst [12] uses OpenMP [14], and [62] compares Cilk [8], HPX-5, and OpenMP tasks [17]. Our choice of HPX for the task-based runtime system is motivated by the same findings as the above mentioned review and the need to implement specialized kernels for energy conservation that require coupling between different parts of the solver.

While conservation of linear momentum to machine precision is possible with existing FMM implementations, Octo-Tiger employs a novel extension to the FMM that also ensures conservation of angular momentum to machine precision (see Sec. 4.2). Another extension requires a solution for the time-derivative of the gravitational field to ensure conservation of total energy. The coupling of the gravitational derivative with the hydrodynamics solver in turn requires the use of a volume-based FMM code (making integration of a particle-based code very challenging); HPX's futurization technique makes this coupling straightforward while maintaining efficient hardware utilization. Additionally, the planned addition of radiation transport and other solvers in the future can also take advantage of the unique futurization properties of HPX. None of the other available task-based FFM implementations examined, such as PVFMM, ExaFMM-alpha, minifmm, or DASHMM met the requirements for integration into Octo-Tiger. The best fitting and scalable of the alternative candidates would be the volume-based PVFMM code; however, it uses Chebyshev polynomials of higher degree, which results in a significantly higher flops/cell rate than our implementation which assumes locally homogeneous densities. In Octo-Tiger, it would be possible to use the surrounding leaf cells to compute higher order multipole moments at the leaf cell level, resulting in a higher computational density

and better GPU performance (as for PVFMM). However, it is not clear how to ensure the conservation of all momenta for polynomials of higher degree. For the reasons cited here, we have developed new FMM kernels, compatible with HPX for this work.

## 3 SCENARIO: STELLAR MERGERS

In September 2008, the contact binary, V1309 Scorpii, merged to form a luminous red novae (LRN) [58]. The Optical Gravitational Lensing Experiment (OGLE) observed this binary prior to its merger, and six years of data show its period decreasing. When the merger occurred, the system increased in brightness by a factor of about 5000. Mason et al. [39] observed the outburst spectroscopically, confirming it as a LRN. This was the first observed stellar merger of a contact binary with photometric data available prior to its merger.

Possible progenitor systems for V1309 Scorpii, consisting initially of zero-age main sequence stars with unequal masses in a relatively narrow range, were proposed by Stepien in [50]. As the heavier of the two stars first begins to expand into a red giant, it transfers mass to its lower mass companion, forming a common envelope. The binary's orbit shrinks due to friction, and the mass transfer slows down as the companion becomes the heavier of the two stars but continues to grow at the expense of the first star. Eventually this star also expands, with both stars now touching each other forming a contact binary. Stepien et. al. sampled the space of physically possible initial masses, finding that initial primary masses of between $1.1M_\odot$ and $1.3M_\odot$ and initial secondary masses between $0.5M_\odot$ and $0.9M_\odot$ produced results consistent with observations prior to merger. The evolution described above results in an approximately $1.52 - 1.54M_\odot$ primary and a $0.16 - 0.17M_\odot$ secondary with helium cores and Sun-like atmospheres. It is theorized that the merger itself was due to the Darwin instability. When the total spin angular momentum of a binary system exceeds one third of its orbital angular momentum, the system can no longer maintain tidal synchronization. This results in a rapid tidal disruption and merger. Octo-Tiger uses its Self-Consistent Field module [20, 23] to produce an initial model for V1309 to simulate this last phase of dynamical evolution. The stars are tidally synchronized, and the stars have a common atmosphere. The system parameters are chosen such that the spin angular momentum just barely exceeds one third of the orbital angular momentum. Octo-Tiger begins the simulation just as the Darwin instability sets in (Fig. 1).

## 4 SOFTWARE FRAMEWORK

### 4.1 HPX

We have developed the Octo-Tiger application framework [52] in ISO C++11 using HPX [24–26, 28, 29, 51]. HPX is a C++ standard library for distributed and parallel programming built on top of an Asynchronous Many Task (AMT) runtime system. Such AMT runtimes may provide a means for helping programming models to fully exploit available parallelism on complex emerging HPC architectures. The HPX methodology described here includes the following essential components:

- An ISO C++ standard conforming API that enables wait-free asynchronous parallel programming, including futures, channels, and other primitives for asynchronous execution.
- An Active Global Address Space (AGAS) that supports load balancing via object migration and enables exposing a uniform API for local and remote execution.
- An active-message networking layer that enables running functions close to the objects they operate on. This also implicitly overlaps computation and communication.
- A work-stealing lightweight task scheduler that enables finer-grained parallelization and synchronization and automatic load balancing across all local compute resources.
- APEX, an in-situ profiling and adaptive tuning framework.

The design features of HPX allow application developers to naturally use key parallelization and optimization techniques, such as overlapping communication and computation, decentralizing control flow, oversubscribing execution resources, and sending work to data instead of data to work. As a result Octo-Tiger achieves exceptionally high system utilization and exposes very good weak- and strong scaling behaviour. HPX exposes an asynchronous, standards conforming programming model enabling Futurization, with which developers can express complex dataflow execution trees that generate billions of HPX tasks that are scheduled to execute only when their dependencies are satisfied [27]. Also, Futurization enables automatic parallelization and load-balancing to emerge. Additionally, HPX provides a performance counter and adaptive tuning framework that allows users to access performance data, such as core utilization, task overheads, and network throughput; these diagnostic tools were instrumental in scaling Octo-Tiger to the full machine.

This paper demonstrates the viability of the HPX programming model at scale using Octo-Tiger, a portable and standards conforming application. Octo-Tiger fully embraces the C++ Parallel Programming Model, including additional constructs that are incrementally being adopted into the ISO C++ Standard. The programming model views the entire supercomputer as a single C++ abstract machine. A set of tasks operates on a set of C++ objects distributed across the system. These objects interact via asynchronous function calls; a function call to an object on a remote node is relayed as an active message to that node. A powerful and composable primitive, the future object represents and manages asynchronous execution and dataflow.

A crucial property of this model is the semantic and syntactic equivalence of local and remote operations. This provides a unified approach to intra- and inter-node parallelism based on proven generic algorithms and data structures available in today's ISO C++ Standard. The programming model is intuitive and enables performance portability across a broad spectrum of increasingly diverse HPC hardware.

## 4.2 Octo-Tiger

Octo-Tiger simulates the evolution of mass density, momentum, and energy of interacting binary stellar systems from the start of mass transfer to merger. It also evolves five passive scalars. It is a three-dimensional finite-volume code with Newtonian gravity that simulates binary star systems as self-gravitating compressible inviscid fluids. To simulate these fluids we need three core components: (1) a hydrodynamics solver, (2) a gravity solver that calculates the gravitational field produced by the fluid distribution, and (3) a solver to generate an initial configuration of the star system.

The passive scalars, expressed in units of mass density, are evolved using the same continuity equation that describes the evolution of the mass density. They do not influence the flow itself, but are rather used to track various fluid fractions as the system evolves. In the case of V1309, these scalars are initialized to the mass density of the accretor core, the accretor envelope, the donor core, the donor envelope, and the common atmosphere between the two stars. The passive scalars are useful in post-processing. For instance, to compute the temperature we require the mass and energy densities as well as the number density. The latter is not evolved in the simulation, but can be computed from the passive scalars assuming a composition for each fraction (e.g. helium for both cores, and a solar composition for the remaining fractions).

The balance of angular momentum plays an important role in the orbital evolution of binary systems. Three-dimensional astrophysical fluid codes with self-gravity do not typically conserve angular momentum. The magnitude of this violation is dependent on the particular problem and resolution. Previous works have found relative violations as high as $10^{-3}$ per orbit [16, 38, 41]. This error, accumulated over several dozen orbits, becomes significant enough to influence the fate of the system. Octo-Tiger conserves both linear and angular momenta to machine precision. In the fluid solver, this is accomplished using a technique described by [18], while the gravity solver uses our own extension to the FMM.

Octo-Tiger's main datastructure is a rotating Cartesian grid with adaptive mesh refinement (AMR). It is based on an adaptive octree structure. Each node is an $N^3$ sub-grid (with $N = 8$ for all runs in this paper) containing the evolved variables, and can be further refined into eight child nodes. Each octree node is implemented as an HPX component. These octree nodes are distributed onto the compute nodes using a space filling curve. For further information about implementational details we refer to [45] and [37].

The first solver that operates on this octree is a finite volume hydrodynamics solver. Octo-Tiger uses the central advection scheme of [32]. The piece-wise parabolic method (PPM) [13] is used to compute the thermodynamic variables at cell faces. A method detailed by [38] is used to conserve total energy in its interaction with the gravitational field. This technique involves applying the advection scheme to the sum of gas kinetic, internal, and potential energies, resulting in conservation of the total energy. Numerical precision of internal energy densities can suffer greatly in high mach flows, where the kinetic energy dwarfs the gas internal energy. We use the dual-energy formalism of [10] to overcome this issue: We evolve both the gas total energy as well as the entropy. The internal energy is then computed from one or the other depending on the mach number (entropy for high

mach flows and total gas energy for low mach ones). The angular momentum technique described by [18] is applied to the PPM reconstruction. It ads a degree of freedom to the reconstruction of velocities on cell faces by allowing for the addition of a spatially constant angular velocity component to the linear velocities. This component is determined by evolving three additional variables corresponding to the spin angular momentum for a given cell.

The gravitational field solver is based on the FMM. Octo-Tiger is unique in conserving both linear and angular momentum simultaneously and at scale using modifications to the original FMM algorithm [36, 37].

Finally, we assemble the initial scenario using the Self-Consistent Field technique alongside the FMM solver. Octo-Tiger can produce initial models for binary systems that are in contact, semi-detached, or detached [37]. Calculated only once, the computational demands of this solver will be negligible for full-size runs.

We used a test suite of four verification tests, recommended by Tasker et al. [56] for self-gravitating astrophysical codes, to verify the correctness of our results. The first two are purely hydrodynamic tests: the Sod shock tube and the Sedov-Taylor blast wave. Both have analytical solutions which we can use for comparisons. The third and fourth test are a globular star cluster in equilibrium and one in motion. In each case, the equilibrium structure should be retained. Because Octo-Tiger is intended to simulate individual stars self-consistently, we have substituted a single star in equilibrium at rest for the third test and a single star in equilibrium in motion for the fourth test.

## 4.3  The FMM hotspot

The most compute-intensive task is the calculation of the gravitational field using the FMM, since this has to be done for each of the fluid-solver time-steps. Note that our FMM variant differs from approaches such as the implementation used in [61]. While being distributed and GPU-capable, their FMM is operating upon particles. Our FMM variant operates on the grid cells directly since each grid cell has a density value which determines its mass, and thus its gravitational influence on other cells. We further differ from other (cell-based) FMM variants used for computing gravitational fields by conserving not only linear momentum, but also angular momentum, down to machine precision using the changes outlined in [36]. Due to its computational intensity, we will take a closer look at the FMM and its kernels in this section.

The FMM algorithm consists of three steps. First, it computes the multipole moments and the center-of-masses of the individual cells. This information is then used to calculate Taylor-series expansions coefficients in the second and third steps. These coefficients can in turn be used to approximate the gravitational potential in a cell, which can then be used by the hydrodynamics solver [37].

The first of the three FMM steps requires a bottom up traversal of the octree datastructure. The fluid density of the cells of the highest level is the starting point. The multipole moments of every other cell are then calculated using the multipole moments of its child cells. We can additionally compute the center of mass for each refined cell. While this step includes a tree-traversal, it is not very compute intensive.

In the second FMM step (same-level), we use the multipole moments and the center-of-masses to calculate how much the gravity in each cell is influenced by its neighboring cells on the same octree level. How many cells are considered as "neighboring" is determined by the so-called opening criteria [37]. However, their number is constant on each level. The result of these interactions is a Taylor series expansion of interactions. This is the most compute-intensive part.

In the third FMM step, the gravitational influence of cells outside of the opening criteria is computed, and the octree is traversed top-down. The respective Taylor series expansion of the parent node is passed to the child nodes and accumulated.

In the first and third step we calculate interactions between either child nodes and their respective parents or vice-versa. Since a refined node only has 8 children, the number of these interactions is limited. In the second step, the number of same-level interactions per cell that need to be calculated is much higher. For our choice of parameters, each cell interacts with 1074 of its close neighbors, assuming they exist.

The second FMM step (same-level interactions) is by far the most compute-intense part. Originally, it required about 70% of the total scenario runtime and was thus the core focus of previous optimizations. Originally, lookup of close neighbor cells was performed using an interaction list, and data was stored in an array-of-struct format. In order to improve cache-efficiency and vector-unit usage, we changed it to a stencil-based approach and are now utilizing a struct-of-arrays datastructure. Compared to the old interaction-list approach, this led to a speedup of the total application runtime between 1.90 and 2.22 on AVX512 CPUs and between 1.23 and 1.35 on AVX2 CPUs [15]. Furthermore, we achieved node-level scaling as well as performance portability between different CPU architectures through the usage of Vc [15, 45]. After these optimizations, the FMM required only about 40% (depending on the hardware) of the total scenario runtime with its compute kernels reaching a significant fraction of peak on multiple platforms as we will demonstrate in Sect. 6.1.

Due to the presence of AMR, there are four different cases of same-level interactions: 1) multipole-monopole interactions between cells of a refined octree node (multipoles) and cells of a non-refined octree node (monopoles); 2) multipole-multipole interactions; 3) monopole-monopole interactions; and 4) monopole-multipole interactions. This yields four kernels per octree-node. Their input data are the current node's sub-grid as well as all sub-grids of all neighboring nodes as a halo (ghost layer). The kernels then compute all interactions of a certain type and add the result to the Taylor coefficients of the respective cells in the sub-grid. We were able to combine the multipole-multipole and the multipole-monopole kernels into a single kernel, yielding three compute kernels in our implementation.

As the monopole-multipole kernel consumes only about 2% of the total runtime, we ignore it in the rest of this work.

The remaining two compute kernels, 1)/2) and 3), are the central hotspots of the application and will henceforth be called FMM kernels. Each kernel launch applies a 1074 element stencil for each cell of the octree's sub-grid. As we have $N^3 = 512$ cells per sub-grid, this results in 549 888 interactions per kernel launch. Depending on the interaction type, each of those interactions requires a different number of floating point operations to be executed. For monopole-monopole interactions we execute 12 floating point operations per interaction, and for multipole-multipole/monopole interaction 455 floating point operations. More information about the kernels can be found in [45]; however, the number of floating operations per monopole interaction differs slightly there as we combined the two monopole-X kernels there.

## 5 IMPROVING OCTO-TIGER USING HIGH-LEVEL ABSTRACTIONS

Running an irregular, adaptive application like Octo-Tiger on a heterogeneous supercomputer like Piz Daint presents challenges: The pockets of parallelism contained in each octree node must be run efficiently on the GPU, despite the relatively small number of cells in each sub-grid. The GPU implementation should not degrade parallel efficiency through overheads such as work aggregation, CPU/GPU synchronization, or blocked CPU threads. Furthermore, we expect the implementation to behave as before, with the exception of faster GPU execution of tasks.

In this section, we first present our implementation and integration of FMM GPU kernels into the task flow using HPX CUDA futures as a high-level abstraction. We then introduce the libfabric parcelport and show how this new communication layer improves scalability of Octo-Tiger by taking advantage of HPX's communications abstractions.

### 5.1 Asynchronous Many Tasks with GPUs

As our FMM implementation is stencil-based and uses a struct-of-arrays datastructure, the FMM kernels introduced in Section 4.3 are very amenable to GPU execution. Each kernel executes a 1074 element stencil on the 512 cells of the 8x8x8 sub-grid of an octree node, calculating the gravitational interactions of each cell with its 1074 neighbors. We parallelize over the cells of the sub-grid, launching kernels with 8 blocks, each containing 64 CUDA threads which execute the whole stencil for each cell. The stencil-based computation of the interactions between two cells is done the same way as on the CPU. In fact, since we use Vc datatypes for vectorization on the CPU, we can simply instance the same function template (that computes the interaction between two cells) with scalar datatypes and call it within the GPU kernel. GPU-specific optimizations are done in a wrapper around this cell-to-cell method and the loop over the stencil elements. This wrapper includes the usual CUDA optimizations such as shared and constant memory usage.

Thus far we have used standard CUDA to create rather normal kernels for the FMM implementation. However, these kernels alone suffer from two major issues: As it stands, the execution of a GPU kernel would block the CPU thread launching it, no other task would be scheduled or executed whilst it runs. As Octo-Tiger relies on having thousands of tasks available simultaneously for scalability, this presents a problem. The second issue is obvious when looking at the size of the workgroups and the number of blocks for each GPU kernel launch mentioned above. The GPU kernels do not expose enough parallelism to fully utilize a GPU such as the NVIDIA P100 using only small workgroups and 8 blocks per kernel. To solve these two issues, we provide an HPX abstraction for CUDA streams.

For any CUDA stream event we create an HPX future that becomes ready once operations in the stream (up to the point of the event/future's creation) are finished. Internally, this is created using a CUDA callback function that sets the future ready [24]. This seemingly simple construct allows us to fully integrate CUDA kernels within the HPX runtime, as it provides a synchronization point for the CUDA stream that is compatible with the HPX scheduler. It yields multiple immediate advantages:

- Seamless and automatic execution of kernels and overlapping of CPU/GPU tasks;
- overlapping of computation and communication as some HPX tasks are related to the communication with other compute nodes; and
- CPU/GPU data synchronization - completed GPU kernels triggering the scheduler, signal access to buffers that can be used/copied.

Furthermore, the integration is mostly non-invasive since a CUDA kernel invocation now equates to a function call returning a future. The rest of the kernel implementation and the (asynchronous) buffer handling uses the normal CUDA API, thus the GPU kernels themselves can still be hand-optimized. Nonetheless, this integration alone does not solve the second issue: The kernels are too fine-grained to fully utilize the GPUs. Conventional approaches to solve this include work aggregation and execution models where CUDA kernels can call other kernels and coalesce execution.

Unfortunately, work aggregation schemes, as described in [42], do not fit our task-based approach. Individual kernels should finish as soon as possible in order to trigger dependent ones, such as communication with other nodes or the third FMM step; delays in launching these may lead to a degradation of parallel efficiency. Recursively calling other GPU kernels as in [59] poses a similar problem as we would traverse the octree on the GPU, making communication calls more difficult. Furthermore, we would like to run code on the appropriate device; tree traversals on the CPU, and processing of the octree kernels on the GPU.

Here, however, we can exploit the fact that the execution of GPU kernels is just another task to the HPX runtime system: We launch a multitude of different GPU kernels concurrently on different streams with each CPU thread handling multiple CUDA streams, and thus multiple GPU kernels concurrently. Normally, this would present problems for CPU/GPU synchronization as GPU results are needed for other CPU tasks. But the continuation passing style of

program execution in HPX, chaining dependent tasks onto futures, makes this trivial. When a GPU kernel output (or data transfer) that has not yet finished is needed for a task, the runtime assigns different work to the CPU and schedules the dependent tasks when the GPU future becomes ready. When the number of concurrent GPU tasks running matches the total number of available CUDA streams (usually 128 per GPU), new kernels are instead executed as CPU tasks until a CUDA stream becomes empty again.

In summary, the octree is traversed on the CPU, with tasks spawned asynchronously for kernels on the GPU or CPU returning futures for each. Any tasks that require results from previous ones are attached as continuations to the futures. The CPU is continuously supplied with new work (including communication tasks) as futures complete. Since all CPU threads may participate in traversal and steal work from each other, we keep the GPU busy by nature of the sheer number of concurrent GPU kernels submitted.

Octo-Tiger is the first application to use HPX CUDA futures. It is in fact an ideal fit for this kind of GPU integration: Parallelization is possible only within individual timesteps of the application, and a production run simulation will require tens of thousands of them, making it is essential to maximize parallel efficiency (as well as proper GPU usage), particularly as each timestep might run for a fraction of a second on the whole machine overall. The fine-grained approach of GPU usage presented here fits these challenges perfectly.

In Section 6 we show how this model performs. We run a real-world scenario for a few timesteps to both show that we achieve a significant fraction of GPU peak performance during the execution of the FMM, as well as scalability on the whole Piz Daint machine, each of the 5400 compute nodes using a NVIDIA P100 GPU. Thus, Octo-Tiger also serves as a proof as concept, showing that large, tree-based applications containing pockets of parallelism can efficiently run fine-grained parallelism tasks on the GPU without compromising scalability with HPX.

## 5.2 Active messages and libfabric parcelport

The programming model of HPX does not rely on the user matching network sends and receives explicitly as one would do with MPI. Instead, active messages are used to transfer data and trigger a function on a remote node; we refer to the triggering of remote functions with bound arguments as actions and the messages containing the serialized data and remote function as parcels [7]. A halo exchange, for example, written using MPI involves a receive operation posted on one node and a matching send on another. With non-blocking MPI operations, the user may check for readiness of the received data at a convenient place in the code and then act appropriately. With blocking ones, the user must wait for the received data and can only continue as soon as it arrives.

With HPX, the same halo exchange may be accomplished by creating a future for some data on the receiving end, and having the sending end trigger an action that sets the future *ready* with the contents of the parcel data. Since futures in HPX are the basic synchronization primitive for work,

the user may attach a continuation to the receive data to start the next calculation that depends on it. The user does not therefore have to perform any test for readiness of the received data: When it arrives, the runtime will set the future and schedule whatever work depends upon it automatically. This combines the convenience of both a blocking receive to trigger work, with an asynchronous receive that allows the runtime to continue whilst waiting.

The asynchronous send/receive abstraction in HPX has been extended with the concept of a *channel* that the receiving end may fetch futures from (for $N$ timesteps ahead if desired) and the sending end may push data into as it is generated. Channels are set up by the user similar to MPI communicators; however, the handles to channels are managed by AGAS (Sect. 4.1). Even when a grid cell is migrated from one node to another during operation, the runtime manages the updated destination address transparently, allowing the user code to send data to the relocated grid with minimal disruption. These abstractions greatly simplify user level code and allow performance improvements in the runtime to be propagated seamlessly to all places that use them.

The default messaging layer in HPX is built on top of the asynchronous two-sided MPI API and uses Isend/Irecv within the parcel encoding and decoding steps of action transmission and execution. HPX is designed from the ground up to be multi-threaded, avoid locking/waiting, and instead suspend tasks and execute others as soon as any blocking activity takes place. Although MPI supports multi-threaded applications, it has its own internal progress/scheduling management and locking mechanisms that interfere with the smooth running of the HPX runtime. The scheduling in MPI is in turn built upon the network provider's asynchronous completion queue handling and multi-threaded support which may also use OS level locks that suspend threads (and thus impede HPX progress).

The HPX parcel format is more complex than a simple MPI message, but the overheads of packing data can be kept to a minimum [7] by using remote memory access (RMA) for transfers. All user/packed data buffers larger than the eager message size threshold are encoded as pointers and exchanged between nodes using one-sided RMA put/get operations. Switching HPX to use the one-sided MPI RMA API is no solution as this involves memory registration/pinning that is passed through to the provider level API, causing additional (unwanted) synchronization between user code, MPI code, and the underlying network/fabric driver. By-passing MPI and using the network API directly to improve performance was seen as a way of decreasing latency, improving memory management, simplifying the parcelport code, and better integrating the multi-threaded runtime with the communications layer. Libfabric was chosen as it has an ideal API that is supported on many platforms, including Cray machines via the GNI provider [46].

The purely asynchronous API of libfabric blends seamlessly with the asynchronous internals of HPX. Any task scheduling thread may poll for completions in libfabric and set futures to received data without any intervening layer. A one-to-one

mapping of completion events to ready futures is possible for some actions, and dependencies for those futures can be immediately scheduled for execution. We expose pinned memory buffers for RMA to libfabric via allocators in the HPX runtime, so that internal data copying between user buffers (halos for example) and the network is minimized. When dealing with GPUs capable of multi TFlop performance, even delays of the order of microseconds in receiving data and subsequent task launches translates to a significant loss of compute capability. Note that with the HPX API it is trivial to reserve cores for thread pools dedicated to background processing of the network separate from normal task execution to further improve performance, but this has not yet been attempted with the Octo-Tiger code.

Our libfabric parcelport uses only a small subset of the libfabric API but delivers very high performance as we demonstrate in Sect. 6.2. It should be stressed that the improvements we see in throughput are more a result of switching from two to one-sided communication, rather than abandoning MPI. Similar gains could probably be made using the MPI RMA API, but this would require a much more complex implementation.

It is a significant contribution of this work that we have demonstrated that an application may benefit from significant performance improvements in the runtime without changing a single line of the application code. This has been achieved utilizing abstractions for task management, scheduling, distribution, and messaging. It is generally true of any library that improvements in performance will produce corresponding improvements in code using it. But switching a large codebase to one-sided or asynchronous messaging is usually a major operation that involves redesigns of significant portions to handle synchronization between previously isolated (or sequential) sections. The unified futurized and asynchronous API of HPX provides a unique opportunity to take advantage of improvements at all levels of parallelism throughout a code as all tasks are naturally overlapped. And network bandwidth and latency improvements reduce waiting not only for remote data, but the effects of improved scheduling of all messages (synchronization of remote tasks as as well as direct data movement) directly impacts and improves on-node scheduling and thus benefits all tasks.

## 6 RESULTS

The initial model of our V1309 simulation includes a $1.54M_\odot$ primary and a $0.17M_\odot$ secondary. Each have helium cores and solar composition envelopes, and there is a common envelope surrounding both stars. The simulation domain is a cubic grid with edges $1.02 \times 10^3 R_\odot$ long. This is about 160 times larger than the initial orbital separation, providing space for any mass ejected from the system. The sub-grids are $8 \times 8 \times 8$ grid cells. The centers of mass of the components are $6.37 R_\odot$ apart. The grid is rotating about the z-axis with a period of 1.42 days, corresponding to the initial period of the binary. For the level 14 run, both stars are refined down to 12 levels, with the core of the accretor and donor refined

| | | | |
|---|---|---|---|
| HPX | 45f3d80 | Vc | 1.4.1 |
| Boost | 1.68.0 | hwloc | 2.0.3 |
| GCC | 7.3.0 | tcmalloc/gperftools | 2.7 |
| Cray-MPICH | 7.7.2 | HDF5 | 1.10.4 |
| Silo | 4.10.2 | libfabric | 1.7.0 |
| CUDA | 9.2 | cmake | 3.12.0 |

**Table 1: Software dependencies of Octo-Tiger (d6ad085).**

to 13 and 14 levels respectively. The 15, 16, and 17 level runs are successively refined one more level in each refinement regime. At the finest level, each grid cell is $7.80 \times 10^{-3} R_\odot$ in each dimension for level 14, down to $9.750 \times 10^{-4} R_\odot$ for level 17. Although available compute time allowed us only to simulate a few time-steps for this work, this is exactly the production scenario we aim for. For all obtained results, the software dependencies in Table 1 were used to build Octo-Tiger (d6ad085) on the various platforms.

### 6.1 FMM Node-Level Performance

In the following, we will take a closer look at the performance of the FMM kernels, discussed in Sect. 4.3 and 5.1, on both GPUs and different CPU platforms. We will first explain how we made measurements and then discuss the results.

*6.1.1 Measuring the Node-Level Performance.* Measuring the node-level results for the FMM solver alone presents several challenges. Instead of a few large kernels, we are executing millions of small FMM kernels overall. Additionally, one FMM kernel alone will never utilize the complete device. On the CPU, each FMM kernel is executed by just one core. We cannot assume that the other cores will always be busy executing an FMM kernel as well. On the GPU, one kernel will utilize only up to 8 Streaming Multiprocessors (SM). The NVIDIA P100 GPU contains 56 of these SMs, each of which is analogue to a SIMD-enabled processor core.

In order to see how well we utilize the given hardware with the FMM kernels, we focus not on the performance of a single kernel. We rather focus on the overall performance while computing the gravity during the GPU-accelerated FMM part of the code.

In order to calculate both the GFLOP/s and the fraction of peak performance, we need to know the number of floating point operations executed while calculating the gravity, as well as the time required to do so. The first piece of information is easy to collect. Each FMM kernel always executes a constant number of floating point operations. We count the number of kernel launches in each HPX thread and accumulate this number until the end of the simulation. We can further record whether a kernel was executed on the CPU or the GPU. Due to the interleaving of kernels and the general lack of synchronization points between the gravity solver and the fluid solver, the amount of runtime spent in the FMM solver is more difficult to obtain. To measure it, we run the simulation multiple times; first, on the CPU without any GPUs. We collect profiling data with perf to get an estimation of the fraction of the runtime spent within the FMM kernels and thus the gravity solver. With this information we

| Utilized Hardware | Execution | Total scenario runtime | FMM | | |
|---|---|---|---|---|---|
| | | | runtime | GFLOP/s | fraction of peak |
| Intel®Xeon™E5-2660 v3 , 2.4 GHz, 10 Cores | CPU-only | 2950s | 1228s | 125 GFLOP/s | 30% |
| with 1x NVIDIA®V100 (PCI-E) | 1 GPU | 1790s | 68s | 2271 GFLOP/s | 32% |
| with 2x NVIDIA®V100 (PCI-E) | 2 GPU | 1770s | 48s | 3185 GFLOP/s | 22% |
| Intel®Xeon™E5-2660 v3 , 2.4 GHz, 20 Cores | CPU-only | 1601s | 614s | 250 GFLOP/s | 30% |
| with 1x NVIDIA®V100 (PCI-E) | 1 GPU | 1086s | 100s | 1516 GFLOP/s | 22% |
| with 2x NVIDIA®V100 (PCI-E) | 2 GPU | 1017s | 30s | 5188 GFLOP/s | 37% |
| Intel®Xeon™Phi 7210 , 1.3 GHz, 64 Cores | | 1774s | 334s | 459 GFLOPS/s | 17% |
| **One Piz Daint Node** | | | | | |
| Intel®Xeon®E5-2690v3 , 2.6GHz, 12 Cores | CPU-only | 2415s | 980s | 157 GFLOP/s | 31% |
| with 1x NVIDIA®P100 (PCI-E) | 1 GPU | 1592s | 158s | 973 GFLOP/s | 21% |

**Table 2: FMM kernel node-level performance on various platforms. On platforms with GPUs we compare the performance with and without GPUs. The theoretical peak performance used for calculating the fraction of peak performance corresponds to the utilized device.**

calculate the fraction of the runtime spent outside the gravity solver. Afterwards we repeat the run – without perf – and multiply its total runtime with the earlier obtained runtime fractions to get both the time spent in the gravity solver and the time spent in other methods. With this information, as well as the counters for the FMM kernel launches, we can now calculate the GFLOP/s achieved by the CPU when executing the FMM kernels. To get the same information for the GPUs, we include them in a third run of the same simulation. Using the GPUs, only the runtime of the gravity solver will improve since the rest of the code does not benefit from them. Thus, by subtracting the runtime spent outside of the FMM kernels in the CPU-only run from the total runtime of the third run, we can estimate the overall runtime of the GPU-enabled FMM kernels and with that the GFLOP/s we achieve overall during their execution.

For all results in this work, we employ the same V1309 scenario and double precision calculations. The level 14 octree discretization considered here will serve as the baseline for scaling runs.

*6.1.2 Results.* The results of our node-level runs can be found in Tab. 2. Switching to a stencil-based approach for the FMM instead of the old interaction-lists, the fraction of time spent in the two main FMM kernels shrank considerably. On the Intel Xeon E5-2660 v3 with 20 cores, they now only make up 38% of the total runtime. On the Intel Xeon Phi 7210 this difference is even higher, with the FMM only making up 20% of the total runtime. This is most likely due to the fact that the other less optimized parts of Octo-Tiger make fewer use of the SIMD capabilites that the Xeon Phi offers and are thus running a lot slower. This reduces the overall fraction of the FMM runtime compared to the rest of the code.

Nevertheless, we achieve a significant fraction of peak performance on all devices. On the CPU-side, the Xeon Phi 7210 achieves the most GFLOP/s within the FMM kernels. Since it lowers its frequency to 1.1 GHz during AVX-intensive calculations, the real achieved fraction of peak performance may

be significantly higher than 17%. We have assumed the base (unthrottled) clock rate shown in the table for calculating the theoretical peak performance of the CPU devices. Other than running a specific Vc version that supports AVX512 on Xeon Phi, we did not adapt the code. However, we attain a reasonable fraction of peak performance on this difficult hardware. On the AVX2 CPUs we reach about 30%.

We tested GPU performance of the FMM kernels in multiple hardware configurations; we used either 10 or 20 cores in combination with either one or two V100 GPUs. Using two V100 GPUs, an insufficient number of cores affects performance. With 20 cores and two GPUs we achieve 37% of the combined V100 peak performance. Reducing to 10 cores, the performance drops to 22% of the peak. Then, the GPUs get starved of work, since the 10 cores have a lot of tasks to work on and cannot launch enough kernels on the GPU.

Simultaneously, when utilizing one V100 GPU managed by 10 cores, we achieve 32% of peak performance on the GPU. But using one V100 with 20 CPU cores, the performance decreases, achieving only 22% peak: The number of threads used to fill the CUDA streams of the GPU directly affects the performance. This effect can be explained by the way we handle CUDA streams. Each CPU thread manages a certain number of CUDA streams. When launching a kernel, a thread first checks whether all of the CUDA streams it manages are busy. If not, the kernel will be launched on the GPU using an idle stream. Otherwise, the kernel will be executed on the CPU by the current CPU worker thread. Executing an FMM kernel on the CPU takes significantly longer than on the GPU, as one CPU kernel will be executed on one core. In a CPU-only setting all cores are working on FMM kernels of different octree nodes.

With 20 cores and one V100, the CPU threads first fill all 128 streams with 128 kernel launches. Launching the next kernels, the GPU has not finished yet, and the CPU threads start to work on FMM kernels themselves. This leads to starvation of the GPU for a short period of time, as the CPU threads are not launching more work on the GPU in

|      | Piz Daint |
|------|-----------|
| CPU  | $1 \times$ Intel®Xeon™E5-2690 v3, 2.60GHz, 12 cores |
| GPU  | $1 \times$ NVIDIA®Tesla®P100 |
| RAM  | 64 GB |
| IC   | Cray Aries routing and communications ASIC |

**Table 3: Configuration of Piz Daint.**

the meantime. Having two V100 offsets the problem, as the cores are less likely to work on the FMM themselves: It is more likely that there is a free CUDA stream available. We analyzed the number of kernels launched on the GPU to provide further data on this. Using 20 cores and one V100 we launch 97.4995% of all multipole-multipole FMM kernels on the GPU. Using 10 cores and one V100 this number increases to 99.9997%. Considering that a CPU FMM execution on one core takes longer than on the GPU and that during this time no other GPU kernels are launched in the meantime, the small difference in percentage can cause a large performance impact. This is a current limitation of our implementation and will be addressed in the next version of Octo-Tiger: There is no reason not to launch multiple FMM kernels in one stream if there is no empty stream available. This would lead to 100% of the FMM kernels launched on the GPU independent of the CPU hardware.

Since Piz Daint is our target system, we also evaluated performance on one of its nodes, using 128 CUDA streams. For comparison, 99.5207% of all multipole-multipole FMM kernels were launched on the GPU. We achieve about 21% of peak performance on the GPU. In summary, we were able to demonstrate that the uncommon approach of launching many small kernels is a valid way to utilize the GPU.

## 6.2 Scaling results

All of the presented distributed scaling results were obtained on Piz Daint at the Swiss National Supercomputing Centre. Table 3 lists the hardware configuration of Piz Daint.

For the scalability analysis of Octo-Tiger different levels of refinement of the V1309 scenario were run, as shown in Tab. 4. A level 13 restart file, which takes less than an hour to generate on an Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz, was used as the basis for all runs. For all levels the restart file for level 13 was read and refined to higher levels of resolution through conservative interpolation of the evolved variables. The number of nodes was increased in powers of two $1, 2, 4, \ldots$ up to 4096 nodes with a maximum of 5400 which corresponds to the full system on Piz Daint. All runs utilized 12 CPU cores on each node, i.e. up to $64, 800$ cores for the full-system run. The simulations started at level 14, the smallest that fits on a single Piz Daint node with respect to memory while still consisting of an acceptable number of sub-grids to expose sufficient parallelism. The number of nodes was increased by a power of two until the scaling saturated due to too little work per node. Higher refinement levels were then run on the largest overlapping node counts to produce the graph shown in Fig. 2, where the speedup is calculated with respect to the number of processed sub-grids per second on one node at level

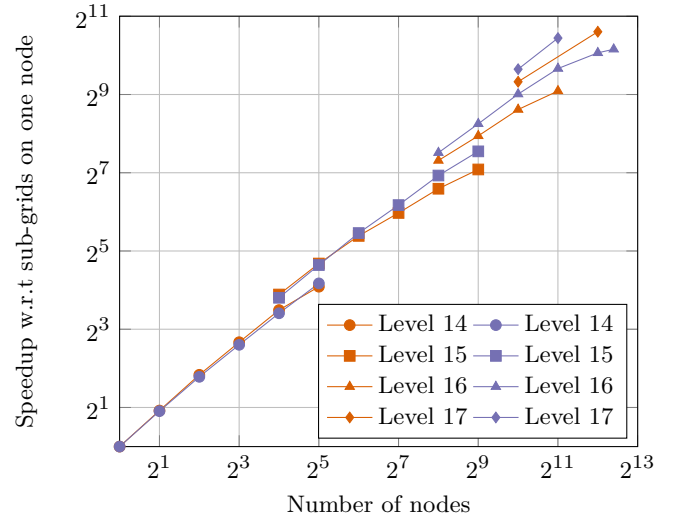| Level of refinement | sub-grids | memory usage (GB) |
|---------------------|-----------|-------------------|
| 13 | 5,417 | 8 |
| 14 | 10,928 | 16.37 |
| 15 | 42,947 | 56.92 |
| 16 | $2.24 \cdot 10^5$ | 271.94 |
| 17 | $1.5 \cdot 10^6$ | 2,305.92 |

**Table 4: Number of tree nodes (sub-grids) per level of refinement (LoR) and the memory usage of the corresponding level.**



**Figure 2: Relative speedup with respect to the processed sub-grids on one node for level 14. The red lines show the results using HPX's MPI parcelport and the blue lines using HPX's libfabric parcelport, respectively. Note that for level 16 and level 17 some data points are missing due to restricted node hours for development projects.**

14. The graph therefore shows a combination of weak scaling as the level of refinement increases and strong scaling for each refinement level as the node count increases. Weak scaling is clearly very good, with close to optimal improvements with successive refinement levels. Strong scaling tails off as the amount of sub-grids for each level becomes too small to generate sufficient work for all CPUs/GPUs.

## 6.3 Network performance results

Figure 2 shows the speedup of both libfabric and MPI parcelport on Piz Daint. The libfabric parcelport scales much better than the MPI parcelport and in fact outperforms it by a factor of almost 3 for the largest runs. At level 17 on 1024 nodes, the libfabric version achieves a (weak) scalability of 78.4% of the efficiency of the reference value of level 14 on 1 node; for 2048 nodes the value drops to 68.1%. Where there is enough work to keep processors busy and overlap communication for large runs, impressive scaling can be observed. At level 16 the efficiency values range from 71.4% at 256 nodes down to 21.2% on 5400 nodes where the communication dominates. The performance difference between the number of sub-grids
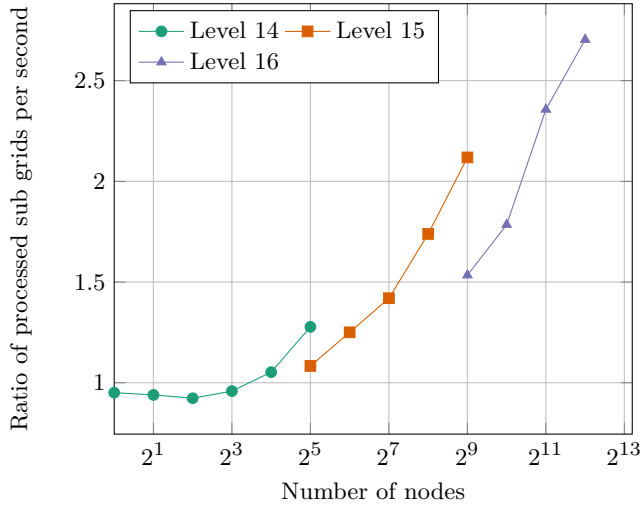
**Figure 3: Ratio of processed sub grids per second between HPXs libfabric and MPI Parcelport on Piz Daint (higher numbers mean libfabric is faster).**

processed per second for the two parcelports increases with higher node counts and refinement level, a sure sign that communication is responsible for causing delays that prevent the processing cores from getting work done. Each increase in the refinement level can, due to AMR, increase the total number of grids by up to a factor of 8; see Tab. 4 for the actual values. This causes a near quadratic increase in the total number of halos exchanged. As the node count increases, the probability of a halo exchange increases linearly, and it is therefore no surprise that reduced communication latency leads to the large gains observed. The improvement in communication is due to all of the following changes:

- Explicit use of RMA for the transfer of halo buffers.
- Lower latency on send and receive of all parcels and execution of RMA transfers.
- Direct control of all memory copies for send/receive buffers between the HPX runtime and the libfabric driver.
- Reduced overhead between receipt of a transfer/message completion event and subsequent setting of a *ready* future.
- Thread-safe lock-free interface between the HPX scheduling loop and the libfabric API with polling for network progress/completions integrated into the HPX task scheduling loop.

It is important to note that the timing results shown are for the core calculation steps that exchange halos, and the figures do not include regridding steps or I/O that also make heavy use of communication. Including them would further illustrate the effectiveness of the networking layer: Start-up timings of the main solver at refinement level 16 and 17 were in fact reduced by an order of magnitude using the libfabric parcelport, increasing the efficiency of refining the initial restart file of level 13 to the desired level of resolution. Note further that some data points at level 16 and 17 for large runs are missing as the start-up time consumed the limited node hours available to their execution.

The communication speedups shown have not separately considered the effects of thread pools and the scheduling of network progress on the rates of injection or the handling of messages. When running on Piz Daint with 12 worker threads executing tasks, any thread might need to send data across the network. In general, the injection of data into send queues does not cause problems unless many threads are attempting to do so concurrently and the send queues are full. The receipt of data, however, must be performed by polling of completion queues. This can only take place in-between the execution of other tasks. Thus, if all cores are busy with work, no polling is done, and if no work is available, all cores compete for access to the network. The effects can be observed in Fig. 3 where the libfabric parcelport causes a slight reduction in performance for lower node counts. With GPUs doing most of the work, CPU cores can be reserved for network processing, and the job of polling can be restricted to a subset of cores that have no other (longer running) tasks to execute. HPX supports partitioning of a compute node into separate thread pools with different responsibilities; the effects of this will be investigated further to see whether reducing contention between cores helps to restore the lost performance.

## 7 CONCLUSIONS AND FUTURE WORK

As the core contributions of this paper, we have demonstrated node-level and distributed performance of Octo-Tiger, an astrophysics code simulating a stellar binary merger. We have shown excellent scaling up to the full system on Piz Daint and improved network performance based on the libfabric library. The high-level abstractions we employ, in particular HPX and Vc, demonstrate how portability in heterogeneous HPC systems is possible. This is the first time an HPX application was run on a full system of a GPU-accelerated supercomputer. This work also has several implications for parallel programming for future architectures. The asynchronous many-task runtime systems like HPX are a powerful, viable, and promising addition to the current landscape of parallel programming models. We show that it is not only possible to utilize these emerging tools to perform on the largest scales, but also that it might even be desirable to leverage the latency hiding, finer-grained parallelism and natural support for heterogeneity that the asynchronous many-task model exposes.

In particular, we have significantly increased node-level performance of the originally most compute hungry part of Octo-Tiger, the gravitational solver. Our optimizations have demonstrated excellent node-level performance on different HPC compute nodes with heterogeneous hardware, including multi-GPU systems and KNL. We have achieved up to 37% of the peak performance on two NVIDIA V100 GPUs, and 17% of peak on a KNL system. To achieve high node-level performance for the full simulation, we will also port the remaining part, the hydrodynamics solver, to GPUs.

The distributed scaling results have been obtained within a development project on Piz Daint and thus with severely limited compute time. The excellent results presented in this

paper have already built the foundation for a production proposal that will enable us to target full-resolution simulations with impact on physics.

Despite the significant performance improvement replacing MPI with libfabric, there are more networking improvements under development that have not been incorporated into Octo-Tiger yet. This includes the use of user-controlled RMA buffers that allow the user to instruct the runtime that certain memory regions will be used repeatedly for communication (and thus amortize memory pinning/registration costs). Integration of such features into the channel abstraction may prove to reduce latencies further and is an area we will explore.

With respect to the astrophysical application, we have already developed a radiation transport module for Octo-Tiger based on the two moment approach adapted by [48]. This will be required to simulate the V1309 merger with high accuracy. What remains is to fully debug and verify this module and to port the implementation to GPUs.

Finally, our full-scale simulations will be able to predict the outcome of mergers that have not yet happened: These simulations will useful for comparison with future "red nova" contact-binary merger events. Two contact-binary systems have been suggested as future mergers, KIC 9832227 [40, 49] and TY Pup [47]. Other candidate systems will be discovered with the new all-sky surveys such as the Zwicky Transient Facility (ZTF) and the Large Synoptic Survey Telescope (LSST).

## ACKNOWLEDGMENTS

## REFERENCES

[1] [n. d.]. Red Giant and Main Sequence Binary (V1309 Sco). https://www.sharcnet.ca/~jnandez/simulations.html. Accessed: 2019-03-14.

[2] [n. d.]. StarSmasher - a Smoothed Particle Hydrodynamics code. https://jalombar.github.io/starsmasher/. Accessed: 2019-03-14.

[3] Emmanuel Agullo, Berenger Bramas, Olivier Coulaud, Eric Darve, Matthias Messner, and Toru Takahashi. 2016. Task-based FMM for heterogeneous architectures. *Concurrency and Computation: Practice and Experience* 28, 9 (2016), 2608–2629.

[4] Emmanuel Agullo, Bérenger Bramas, Olivier Coulaud, Martin Khannouz, and Luka Stanisic. 2016. *Task-based fast multipole method for clusters of multicore processors*. Ph.D. Dissertation. Inria Bordeaux Sud-Ouest.

[5] Cédric Augonnet, Samuel Thibault, Raymond Namyst, and Pierre-André Wacrenier. 2011. StarPU: a unified platform for task scheduling on heterogeneous multicore architectures. *Concurrency and Computation: Practice and Experience* 23, 2 (2011), 187–198.

[6] Michael Bauer, Sean Treichler, Elliott Slaughter, and Alex Aiken. 2012. Legion: Expressing locality and independence with logical regions. In *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE, 1–11.

[7] John Biddiscombe, Thomas Heller, Anton Bikineev, and Hartmut Kaiser. 2017. Zero Copy Serialization using RMA in the Distributed Task-Based HPX runtime. In *14th International Conference on Applied Computing*. IADIS, International Association for Development of the Information Society.

[8] Robert D Blumofe, Christopher F Joerg, Bradley C Kuszmaul, Charles E Leiserson, Keith H Randall, and Yuli Zhou. 1996. Cilk: An efficient multithreaded runtime system. *Journal of parallel and distributed computing* 37, 1 (1996), 55–69.

[9] George Bosilca, Aurelien Bouteiller, Anthony Danalis, Mathieu Faverge, Thomas Hérault, and Jack J Dongarra. 2013. Parsec: Exploiting heterogeneity to enhance scalability. *Computing in Science & Engineering* 15, 6 (2013), 36–45.

[10] Greg L Bryan, Michael L Norman, Brian W O'Shea, Tom Abel, John H Wise, Matthew J Turk, Daniel R Reynolds, David C Collins, Peng Wang, Samuel W Skillman, et al. 2014. Enzo: An adaptive mesh refinement code for astrophysics. *The Astrophysical Journal Supplement Series* 211, 2 (2014), 19.

[11] Bradford L Chamberlain, David Callahan, and Hans P Zima. 2007. Parallel programmability and the chapel language. *The International Journal of High Performance Computing Applications* 21, 3 (2007), 291–312.

[12] Jee Choi, Aparna Chandramowlishwaran, Kamesh Madduri, and Richard Vuduc. 2014. A cpu: Gpu hybrid implementation and model-driven scheduling of the fast multipole method. In *Proceedings of Workshop on General Purpose Processing Using GPUs*. ACM, 64.

[13] P. Colella and P. R. Woodward. 1984. The Piecewise Parabolic Method (PPM) for Gas-Dynamical Simulations. *J. Comput. Phys.* 54 (Sept. 1984), 174–201. https://doi.org/10.1016/0021-9991(84)90143-8

[14] Leonardo Dagum and Ramesh Menon. 1998. OpenMP: An Industry-Standard API for Shared-Memory Programming. *IEEE Comput. Sci. Eng.* 5, 1 (Jan. 1998), 46–55. https://doi.org/10.1109/99.660313

[15] Gregor Daiß. 2018. *Octo-Tiger: Binary Star Systems with HPX on Nvidia P100*. Master thesis. Universität Stuttgart.

[16] Marius Dan, Stephan Rosswog, James Guillochon, and Enrico Ramirez-Ruiz. 2011. Prelude to A Double Degenerate Merger: The Onset of Mass Transfer and Its Impact on Gravitational Waves and Surface Detonations. *Astrophysical Journal (ApJ)* 737, 2, art. id 89 (2011). https://doi.org/10.1088/0004-637X/737/2/89 http://adsabs.harvard.edu/abs/2011ApJ...737...89D.

[17] Bronis R. de Supinski Michael Klemm. 2017. *OpenMP Technical Report 6:Version 5.0 Preview 2*. Technical Report. OpenMP Architecture Review Board.

[18] Bruno Desprésa and Emmanuel Labourasse. 2015. Angular Momentum Preserving Cell-Centered Lagrangian and Eulerian Schemes on Arbitrary Grids. *J. Comput. Phys.* 290 (2015), 28–54. https://doi.org/10.1016/j.jcp.2015.02.032 https://dx.doi.org/10.1016/j.jcp.2015.02.032.

[19] H. Carter Edwards, Christian R. Trott, and Daniel Sunderland. 2014. Kokkos: Enabling manycore performance portability through polymorphic memory access patterns. *J. Parallel and Distrib. Comput.* 74, 12 (2014), 3202 – 3216. https://doi.org/10.1016/j.jpdc.2014.07.003 Domain-Specific Languages and High-Level Frameworks for High-Performance Computing.

[20] Wesley Even and Joel E. Tohline. 2009. Constructing Synchronously Rotating Double White Dwarf Binaries. *The Astrophysical Journal Supplement Series* 184 (Oct 2009), 248–263. https://doi.org/10.1088/0067-0049/184/2/248 arXiv:astro-ph.SR/0908.2116

[21] Joshua Faber, Jamie Lombardi, and Fred Rasio. 2010. StarCrash: 3-d Evolution of Self-gravitating Fluid Systems. *Astrophysics Source Code Library* (2010).

[22] J Davison de St Germain, John McCorquodale, Steven G Parker, and Christopher R Johnson. 2000. Uintah: A massively parallel

problem solving environment. In *Proceedings the Ninth International Symposium on High-Performance Distributed Computing*. IEEE, 33–41.

[23] Izumi Hachisu. 1986. A Versatile Method for Obtaining Structures of Rapidly Rotating Stars. II. Three-dimensional Self-consistent Field Method. *The Astrophysical Journal Supplement Series* 62 (Nov 1986), 461. https://doi.org/10.1086/191148

[24] Thomas Heller, Hartmut Kaiser, Patrick Diehl, Dietmar Fey, and Marc Alexander Schweitzer. 2016. Closing the Performance Gap with Modern C++. In *High Performance Computing (Lecture Notes in Computer Science)*, Michaela Taufer, Bernd Mohr, and Julian M. Kunkel (Eds.), Vol. 9945. Springer International Publishing, 18–31.

[25] Thomas Heller, Hartmut Kaiser, and Klaus Iglberger. 2012. Application of the ParalleX Execution Model to Stencil-Based Problems. *Computer Science - Research and Development* 28, 2-3 (2012), 253–261. https://doi.org/10.1007/s00450-012-0217-1 https://stellar.cct.lsu.edu/pubs/isc2012.pdf.

[26] Thomas Heller, Hartmut Kaiser, Andreas Schäfer, and Dietmar Fey. 2013. Using HPX and LibGeoDecomp for Scaling HPC Applications on Heterogeneous Supercomputers. In *Proceedings of the ACM/IEEE Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA, SC Workshop) (art. id 1)*. https://doi.org/10.1145/2530268.2530269 https://stellar.cct.lsu.edu/pubs/scala13.pdf.

[27] Thomas Heller, Bryce Adelstein Lelbach, Kevin A Huck, John Biddiscombe, Patricia Grubel, Alice E Koniges, Matthias Kretz, Dominic Marcello, David Pfander, Adrian Serio, Juhan Frank, Geoffrey C Clayton, Dirk Pflüger, David Eder, and Hartmut Kaiser. 2019. Harnessing billions of tasks for a scalable portable hydrodynamic simulation of the merger of two stars. *The International Journal of High Performance Computing Applications* (2019). https://doi.org/10.1177/1094342018819744 arXiv:https://doi.org/10.1177/1094342018819744 published online.

[28] Hartmut Kaiser, Thomas Heller, Daniel Bourgeois, and Dietmar Fey. 2015. Higher-level Parallelization for Local and Distributed Asynchronous Task-Based Programming. In *First International Workshop on Extreme Scale Programming Models and Middleware*. 29–37. https://doi.org/10.1145/2832241.2832244 https://stellar.cct.lsu.edu/pubs/executors_espm2_2015.pdf.

[29] Hartmut Kaiser, Thomas Heller, Bryce Adelstein Lelbach, Adrian Serio, and Dietmar Fey. 2014. HPX: A Task Based Programming Model in a Global Address Space. In *Proceedings of the International Conference on Partitioned Global Address Space Programming Models (PGAS) (art. id 6)*. https://doi.org/10.1145/2676870.2676883 https://stellar.cct.lsu.edu/pubs/pgas14.pdf.

[30] Laxmikant V Kale and Sanjeev Krishnan. 1993. CHARM++: a portable concurrent object oriented system based on C++. In *OOPSLA*, Vol. 93. Citeseer, 91–108.

[31] Matthias Kretz. 2015. *Extending C++ for Explicit Data-Parallel Programming via SIMD Vector Types*. Ph.D. Dissertation. Goethe University Frankfurt. https://doi.org/10.13140/RG.2.1.2355.4323 http://publikationen.ub.uni-frankfurt.de/frontdoor/index/index/docId/38415.

[32] Alexander Kurganov and Eitan Tadmor. 2000. New High-Resolution Central Schemes for Nonlinear Conservation Laws and Convection-Diffusion Equations. *J. Comput. Phys.* 160, 1 (2000), 241–282. https://doi.org/10.1006/jcph.2000.6459 https://dx.doi.org/10.1006/jcph.2000.6459.

[33] Hatem Ltaief and Rio Yokota. 2014. Data-driven execution of fast multipole methods. *Concurrency and Computation: Practice and Experience* 26, 11 (2014), 1935–1946.

[34] Morgan MacLeod, Eve C. Ostriker, and James M. Stone. 2018. Bound Outflows, Unbound Ejecta, and the Shaping of Bipolar Remnants during Stellar Coalescence. *The Astrophysical Journal* 868, 2 (dec 2018), 136. https://doi.org/10.3847/1538-4357/aae9eb

[35] Morgan MacLeod, Eve C. Ostriker, and James M. Stone. 2018. Runaway Coalescence at the Onset of Common Envelope Episodes. *The Astrophysical Journal* 863, 1 (aug 2018), 5. https://doi.org/10.3847/1538-4357/aacf08

[36] D. C. Marcello. 2017. A Very Fast and Angular Momentum Conserving Tree Code. *Astronomical Journal* 154, Article 92 (Sept. 2017), 92 pages. https://doi.org/10.3847/1538-3881/aa7b2f arXiv:astro-ph.IM/1706.06989

[37] Dominic C. Marcello, Kundan Kadam, Geoffrey C. Clayton, Juhan Frank, Hartmut Kaiser, and Patrick M. Motl. 2016. Introducing Octo-tiger/HPX: Simulating Interacting Binaries with Adaptive

Mesh Refinement and the Fast Multipole Method. In *Proceedings of the International Conference on Accretion Processes in Cosmic Sources*. http://apcs2016.iaps.inaf.it.

[38] Dominic C. Marcello and Joel E. Tohline. 2012. A Numerical Method for Studying Super-Eddington Mass Transfer in Double White Dwarf Binaries. *The Astrophysical Journal Supplement Series* 199, Article 35 (Apr 2012), 35 pages. https://doi.org/10.1088/0067-0049/199/2/35 arXiv:astro-ph.SR/1404.6208

[39] Mason, E., Diaz, M., Williams, R. E., Preston, G., and Bensby, T. 2010. The peculiar nova V1309 Scorpii/nova Scorpii 2008* - A candidate twin of V838 Monocerotis. *A&A* 516 (2010), A108. https://doi.org/10.1051/0004-6361/200913610

[40] L. A. Molnar, D. M. Van Noord, K. Kinemuchi, J. P. Smolinski, C. E. Alexander, E. M. Cook, B. Jang, H. A. Kobulnicky, C. J. Spedden, and S. D. Steenwyk. 2017. Prediction of a Red Nova Outburst in KIC 9832227. *Astrophysical Journal* 840, Article 1 (May 2017). https://doi.org/10.3847/1538-4357/aa6ba7 arXiv:astro-ph.SR/1704.05502

[41] Patrick M. Motl, Joel E. Tohline, and Juhan Frank. 2002. Numerical Methods for the Simulation of Dynamical Mass Transfer in Binaries. *The Astrophysical Journal Supplement Series* 138, 1 (jan 2002), 121–148. https://doi.org/10.1086/324159

[42] Marc S Orr, Bradford M Beckmann, Steven K Reinhardt, and David A Wood. 2014. Fine-grain task aggregation and coordination on GPUs. *ACM SIGARCH Computer Architecture News* 42, 3 (2014), 181–192.

[43] Ondřej Pejcha, Brian D Metzger, and Kengo Tomida. 2015. Cool and luminous transients from mass-losing binary stars. *Monthly Notices of the Royal Astronomical Society* 455, 4 (2015), 4351–4372.

[44] Ondrej Pejcha, Brian D. Metzger, Jacob G. Tyles, and Kengo Tomida. 2017. Pre-explosion Spiral Mass Loss of a Binary Star Merger. *The Astrophysical Journal* 850, 1 (nov 2017), 59. https://doi.org/10.3847/1538-4357/aa95b9

[45] David Pfander, Gregor Daiß, Dominic Marcello, Hartmut Kaiser, and Dirk Pflüger. 2018. Accelerating Octo-Tiger: Stellar Mergers on Intel Knights Landing with HPX. In *Proceedings of the International Workshop on OpenCL (IWOCL '18)*. ACM, New York, NY, USA, Article 19, 8 pages. https://doi.org/10.1145/3204919.3204938

[46] Howard Pritchard, Evan Harvey, Sung-Eun Choi, James Swaro, and Zachary Tiffany. 2016. The GNI provider layer for OFI libfabric. In *Proceedings of Cray User Group Meeting, CUG*, Vol. 2016.

[47] T. Sarotsakulchai, S.-B. Qian, B. Soonthornthum, X. Zhou, J. Zhang, D. E. Reichart, J. B. Haislip, V. V. Kouprianov, and S. Poshyachinda. 2018. TY Pup: A Low-mass-ratio and Deep Contact Binary as a Progenitor Candidate of Luminous Red Novae. *Journal of Astrophysics* 156, Article 199 (Nov. 2018), 199 pages. https://doi.org/10.3847/1538-3881/aadcfa arXiv:astro-ph.SR/1807.00478

[48] M. Aaron Skinner and Eve C. Ostriker. 2013. A Two-moment Radiation Hydrodynamics Module in Athena Using a Time-explicit Godunov Method. *The Astrophysical Journal Supplement Series* 206, Article 21 (Jun 2013), 21 pages. https://doi.org/10.1088/0067-0049/206/2/21 arXiv:astro-ph.IM/1306.0010

[49] Q. J. Socia, W. F. Welsh, D. R. Short, J. A. Orosz, R. J. Angione, G. Windmiller, D. A. Caldwell, and N. M. Batalha. 2018. KIC 9832227: Using Vulcan Data to Negate the 2022 Red Nova Merger Prediction. *Astrophysical Journal Letters* 864, Article L32 (Sept. 2018), L32 pages. https://doi.org/10.3847/2041-8213/aadc0d arXiv:astro-ph.SR/1809.02771

[50] K. Stępień. 2011. Evolution of the progenitor binary of V1309 Scorpii before merger. *A&A* 531, Article A18 (Jul 2011), A18 pages. https://doi.org/10.1051/0004-6361/201116689 arXiv:astro-ph.SR/1105.2627

[51] STE||AR Group. 2017. HPX GitHub repository. https://github.com/STEllAR-GROUP/hpx. Available under the Boost Software License 1.0 (a BSD-style open source license).

[52] STE||AR Group. 2017. OctoTiger AMR Framework GitHub repository. https://github.com/STEllAR-GROUP/octotiger. Available under the Boost Software License 1.0 (a BSD-style open source license).

[53] James M Stone, Thomas A Gardiner, Peter Teuben, John F Hawley, and Jacob B Simon. 2008. Athena: a new code for astrophysical MHD. *The Astrophysical Journal Supplement Series* 178, 1 (2008), 137.

[54] Stone, James M. and Gardiner, Thomas A. and Teuben, Peter. 2000. Athena++ radiation GRMHD code. https://princetonuniversity.github.io/Athena-Cversion/. Available under the BSD 3-Clause "New" or "Revised" License.
[55] Stone, James M. and Tomida, Kengo and White, Christopher and Felker, Kyle Gerard. 2016. Athena++ radiation GRMHD code. http://princetonuniversity.github.io/athena/. Available under the BSD 3-Clause "New" or "Revised" License.
[56] Elizabeth J. Tasker, Riccardo Brunino, Nigel L. Mitchell, Dolf Michielsen, Stephen Hopton, Frazer R. Pearce, Greg L. Bryan, and Tom Theuns. 2008. A test suite for quantitative comparison of hydrodynamic codes in astrophysics. *Monthly Notices of the Royal Astronomical Society* 390, 3 (Nov 2008), 1267–1281. https://doi.org/10.1111/j.1365-2966.2008.13836.x arXiv:astro-ph/0808.1844
[57] Peter Thoman, Kiril Dichev, Thomas Heller, Roman Iakymchuk, Xavier Aguilar, Khalid Hasanov, Philipp Gschwandtner, Pierre Lemarinier, Stefano Markidis, Herbert Jordan, et al. 2018. A taxonomy of task-based parallel programming technologies for high-performance computing. *The Journal of Supercomputing* 74, 4 (2018), 1422–1434.
[58] R. Tylenda, M. Hajduk, T. Kamiński, A. Udalski, I. Soszyński, M. K. Szymański, M. Kubiak, G. Pietrzyński, R. Poleski, Ł. Wyrzykowski, and K. Ulaczyk. 2011. V1309 Scorpii: merger of a contact binary. *A&A* 528, Article A114 (April 2011), A114 pages. https://doi.org/10.1051/0004-6361/201016221 arXiv:astro-ph.SR/1012.0163
[59] Jin Wang, Norm Rubin, Albert Sidelnik, and Sudhakar Yalamanchili. 2016. Dynamic thread block launch: a lightweight execution mechanism to support irregular applications on GPUs. *ACM SIGARCH Computer Architecture News* 43, 3 (2016), 528–540.
[60] Asim YarKhan, Jakub Kurzak, and Jack Dongarra. 2011. Quark users' guide: Queueing and runtime for kernels. *University of Tennessee Innovative Computing Laboratory Technical Report ICL-UT-11-02* (2011).
[61] Rio Yokota, L.A. Barba, Tetsu Narumi, and Kenji Yasuoka. 2013. Petascale turbulence simulation using a highly parallel fast multipole method on GPUs. *Computer Physics Communications* 184, 3 (2013), 445 – 455. https://doi.org/10.1016/j.cpc.2012.09.011
[62] Bo Zhang. 2014. Asynchronous task scheduling of the fast multipole method using various runtime systems. In *2014 Fourth Workshop on Data-Flow Execution Models for Extreme Scale Computing*. IEEE, 9–16.

# Appendix: Artifact Description/Artifact Evaluation

## SUMMARY OF THE EXPERIMENTS REPORTED

We ran Octotiger on Piz Daint with HPX and CUDA 9.2, as described in the paper.

## ARTIFACT AVAILABILITY

*Software Artifact Availability:* All author-created software artifacts are maintained in a public repository under an OSI-approved license.

*Hardware Artifact Availability:* There are no author-created hardware artifacts.

*Data Artifact Availability:* All author-created data artifacts are maintained in a public repository under an OSI-approved license.

*Proprietary Artifacts:* None of the associated artifacts, author-created or otherwise, are proprietary.

*List of URLs and/or DOIs where artifacts are available:*

```
Scripts to run and build octotiger
↪   https://github.com/STEllAR-GROUP/OctotigerSC19
```

```
Restart files for the v1309 scenario (Version 1.0)
↪   [Data set]. Zenodo.
↪   \(http://doi.org/10.5281/zenodo.2635581\)
```

```
https://github.com/STEllAR-GROUP/hpx/archive/45f3d80⌋
↪   f96eded3a73aaab490a46dca1d97e903c.tar.gz
```

```
https://github.com/STEllAR-GROUP/hpx/archive/8ec0544⌋
↪   1e4d46892fd934301aa521fa54d505868.tar.gz
```

```
https://github.com/STEllAR-GROUP/octotiger/archive/d⌋
↪   6ad085ad4095f9277aae01869de465429fb14c3.tar.gz
```

```
Octo-Tiger with Xeon Phi support: https://github.com⌋
↪   /STEllAR-GROUP/octotiger/archive/c71a12ad6219a66⌋
↪   6e216b5a37713246b628f0875.tar.gz
```

```
https://github.com/VcDevel/Vc/archive/1.4.1.tar.gz
```

```
Vc with Xeon Phi support for argon-knl:
↪   https://github.com/STEllAR-GROUP/Vc/archive/1570⌋
↪   26d3adf3494922269d11a2e4dede09bf867c.tar.gz
```

```
https://github.com/gperftools/gperftools/releases/do⌋
↪   wnload/gperftools-2.7/gperftools-2.7.tar.gz
```

```
https://github.com/live-clones/hdf5/archive/hdf5-1\_⌋
↪   10\_4.tar.gz
```

```
https://download.open-mpi.org/release/hwloc/v1.11/hw⌋
↪   loc-1.11.12.tar.gz
```

```
https://downloads.sourceforge.net/project/boost/boos⌋
↪   t/1.68.0/boost\_1\_68\_0.tar.bz2
```

```
https://wci.llnl.gov/content/assets/docs/simulation/⌋
↪   computer-codes/silo/silo-4.10.2/silo-4.10.2.tar.⌋
↪   gz
```

## BASELINE EXPERIMENTAL SETUP, AND MODIFICATIONS MADE FOR THE PAPER

*Relevant hardware details:* Piz Daint Model: Cray XC50 Processor: Intel Xeon E5-2690 v3 @ 2.60GHz, 12 cores Memory/node: 64 GB; 16 GB CoWoS HBM2 GPU: NVIDIA Tesla P100 for PICe-Based Servers Compute Capability: 6.0 Peak single precision floating point performance: 9.3 TFLOP/s Peak double precision floating point performance: 4.7 TFLOP/s Peak half precision floating point performance: 18.7 TFLOP/s Single precision CUDA cores (FP32): 3584 Double precision CUDA cores (FP64): 1792 Single-precision CUDA cores (FP32): 64 Double-precision CUDA cores (FP64): 32 INT32 cores: N/A Tensor cores/GPU: N/A Tensor cores/SM: N/A Clock frequency: 1126 MHz Memory Bandwidth: 732 GB/s Memory size (HBM2): 16 GB L2 cache: 4096KB Shared memory size/SM: 48 KB Constant memory: 64 KB Register File Size: 256 KB (per SM) 32-bit Registers: 65536 (per SM) Max registers per thread: 255 Number of multiprocessors (SMs): 56 Warp size: 32 threads Maximum resident warps per SM: 64 Maximum resident blocks per SM: 32 Maximum resident threads per SM: 2048 Maximum threads per block: 1024 Maximum block dimensions: 1024, 1024, 64 Maximum grid dimensions: 2147483647, 65535, 65535 Maximum number of MPS clients: 48 Nodes: 5,704 Interconnect Topology: Aries routing and communications ASIC, and Dragonfly network topology Filesystem: Sonexion 1600, 2.5PB, Peak Performance of 138 GB/s argon-knl (node-level platform) Processor: Intel(R) Xeon Phi(TM) CPU 7210 @ 1.30GHz, 64 cores Memory: 94 GB geev (node-level platform) Processor: Intel(R) Xeon(R) CPU E5-2660 v3 @ 2.60GHz, 20 cores Memory: 252 GB GPU: 2x NVIDIA Tesla V100-PCIE

*Operating systems and versions:* Piz Daint Cray Linux Environment (UNICOS) argon-knl (node-level platform) Linux argon-knl 4.4.0-143-generic #169-Ubuntu SMP x86_64 GNU/Linux geev (node-level platform) Linux geev 3.10.0-957.10.1.el7.x86_64 #1 SMP x86_64 GNU/Linux

*Compilers and versions:* GCC 7.3.0 CUDA 9.2 (with GCC 6.3.0) GCC 6.5.0 (for node-level results)

*Applications and versions:* Octo-Tiger changeset d6ad085ad4095f9277aae01869de465429fb14c3 Octo-Tiger with KNL support (for argon-knl) changeset c71a12ad6219a666e216b5a37713246b628f0875

*Libraries and versions:* HPX changeset 45f3d80f96eded3a73aaab490a46dca1d97e903c HPX changeset 8ec05441e4d46892fd934301aa521fa54d505868 (for Libfabric

support) Boost 1.68.0 CMake 3.12.0 HDF5 1.10.4 tcmalloc from gperftools 2.7 hwloc 2.0.3 Silo 4.10.2 Vc 1.4.1 Vc with Xeon Phi support changeset 157026d3adf3494922269d11a2e4dede09bf867c Cray-MPICH 7.7.2 Libfabric: 1.7.0

*Key algorithms:* Octree AMR The central advection scheme of Kurganov & Tadmor (2000) The Fast Multipole Method (FMM) described by Marcello (2017)

*Input datasets and versions:* The restart files generated by Octo-Tiger are quite large and we uploaded them to Zenodo and obtained a DOI ( http://doi.org/10.5281/zenodo.2635581 ). The most important file on this site is the restart.13.silo file which is used as the starting point of our simulations. This file is passed to Octo-Tiger by the parameter restart_file. The ini files contain other parameters required by Octo-Tiger. These are loaded at the beginning of the program with the parameter config_file. Additionally, one does need to use the parameter extra_regrid if the restart file was generated at a lower level than the intended run.

*Paper Modifications:* N/A

*Output from scripts that gathers execution environment information.*

```
Piz Daint:
+ lsb_release -a
LSB Version:        n/a
Distributor ID:     SUSE
Description:        SUSE Linux Enterprise Server 12
↪   SP3
Release:        12.3
Codename:       n/a
+ uname -a
Linux nid03508 4.4.103-6.38_4.0.153-cray_ari_c #1 SMP
↪   Thu Nov 1 16:05:05 UTC 2018 (6ef8fef) x86_64
↪   x86_64 x86_64 GNU/Linux
+ lscpu
Architecture:       x86_64
CPU op-mode(s):     32-bit, 64-bit
Byte Order:         Little Endian
CPU(s):             24
On-line CPU(s) list: 0-23
Thread(s) per core: 2
Core(s) per socket: 12
Socket(s):          1
NUMA node(s):       1
Vendor ID:          GenuineIntel
CPU family:         6
Model:              63
Model name:         Intel(R) Xeon(R) CPU E5-2690 v3
↪   @ 2.60GHz
Stepping:           2
CPU MHz:            2601.000
CPU max MHz:        2601.0000
CPU min MHz:        1200.0000
BogoMIPS:           5199.99
Virtualization:     VT-x
L1d cache:          32K
```

```
L1i cache:          32K
L2 cache:           256K
L3 cache:           30720K
NUMA node0 CPU(s):  0-23
Flags:              fpu vme de pse tsc msr pae mce
↪   cx8 apic sep mtrr pge mca cmov pat pse36 clflush
↪   dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx
↪   pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs
↪   bts rep_good nopl xtopology nonstop_tsc
↪   aperfmperf eagerfpu pni pclmulqdq dtes64 monitor
↪   ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr
↪   pdcm pcid dca sse4_1 sse4_2 x2apic movbe popcnt
↪   tsc_deadline_timer aes xsave avx f16c rdrand
↪   lahf_lm abm ida arat epb invpcid_single pln pts
↪   dtherm spec_ctrl kaiser tpr_shadow vnmi
↪   flexpriority ept vpid fsgsbase tsc_adjust bmi1
↪   avx2 smep bmi2 erms invpcid cqm xsaveopt cqm_llc
↪   cqm_occup_llc
+ cat /proc/meminfo
MemTotal:       65844884 kB
MemFree:        63280600 kB
MemAvailable:   62932780 kB
Buffers:           16836 kB
Cached:           844308 kB
SwapCached:            0 kB
Active:           180276 kB
Inactive:         787852 kB
Active(anon):     136424 kB
Inactive(anon):   691400 kB
Active(file):      43852 kB
Inactive(file):    96452 kB
Unevictable:        8184 kB
Mlocked:            8184 kB
SwapTotal:             0 kB
SwapFree:              0 kB
Dirty:                 0 kB
Writeback:             0 kB
AnonPages:        114796 kB
Mapped:            82904 kB
Shmem:            720248 kB
Slab:             155740 kB
SReclaimable:      23228 kB
SUnreclaim:       132512 kB
KernelStack:        7520 kB
PageTables:         4396 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:    32922440 kB
Committed_AS:    1052200 kB
VmallocTotal:   34359738367 kB
VmallocUsed:           0 kB
VmallocChunk:          0 kB
HardwareCorrupted:     0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
```

```
HugePages_Surp:        0
Hugepagesize:       2048 kB
DirectMap4k:     5362676 kB
DirectMap2M:    12328960 kB
DirectMap1G:    51380224 kB
+ env
LESSKEY=/etc/lesskey.bin
MODULE_VERSION_STACK=3.2.10.6
KSH_AUTOLOAD=1
CRAY_BINUTILS_BIN=/opt/cray/pe/cce/8.7.3/binutils/x8⌋
↪  6_64/bin
PE_LIBSCI_VOLATILE_PRGENV=CRAY GNU INTEL
PE_SMA_DEFAULT_PKGCONFIG_VARIABLES=PE_SMA_COMPFLAG_@⌋
↪  prgenv@
PE_TPSL_64_DEFAULT_GENCOMPS_INTEL_mic_knl=160
SLURM_NODELIST=nid03508
SLURM_CHECKPOINT_IMAGE_DIR=/var/slurm/checkpoint
MANPATH=/users/*REDACTED*/.local/man:/users/*REDACTE⌋
↪  D*/.local/man:/opt/slurm/17.11.12.cscs/share/man⌋
↪  :/opt/cray/pe/mpt/7.7.2/gni/man/mpich:/opt/cray/⌋
↪  pe/perftools/7.0.2/man:/opt/cray/pe/papi/5.6.0.2⌋
↪  /share/pdoc/man:/opt/cray/pe/atp/2.1.2/man:/opt/⌋
↪  cray/alps/6.6.43-6.0.7.0_26.4__ga796da3.ari/man:⌋
↪  /opt/cray/job/2.2.3-6.0.7.0_44.1__g6c4e934.ari/m⌋
↪  an:/opt/cray/pe/pmi/5.0.14/man:/opt/cray/pe/libs⌋
↪  ci/18.07.1/man:/opt/cray/pe/man/csmlversion:/opt⌋
↪  /cray/pe/craype/2.5.15/man:/opt/cray/pe/cce/8.7.⌋
↪  3/man:/opt/cray/pe/modules/3.2.10.6/share/man:/o⌋
↪  pt/slurm/default/share/man:/usr/local/man:/usr/s⌋
↪  hare/man:/opt/cray/share/man:/opt/cray/pe/man
NNTPSERVER=news
PE_PAPI_DEFAULT_ACCEL_FAMILY_LIBS_nvidia=,-lcupti,-l⌋
↪  cudart,-lcuda
PE_PETSC_DEFAULT_GENCOMPILERS_CRAY_sandybridge=8.6
PE_PETSC_DEFAULT_GENCOMPS_CRAY_skylake=86
PE_TPSL_DEFAULT_GENCOMPS_INTEL_x86_skylake=160
PE_CXX_PKGCONFIG_LIBS=mpichcxx
PE_MPICH_GENCOMPILERS_PGI=15.3
SLURM_JOB_NAME=bash
XDG_SESSION_ID=278065
XALT_ETC_DIR=/apps/daint/UES/xalt/0.7.6/etc
HOSTNAME=daint103
CRAY_UDREG_INCLUDE_OPTS=-I/opt/cray/udreg/2.3.2-6.0.⌋
↪  7.0_33.18__g5196236.ari/include
GCC_AARCH64=/opt/gcc-cross-aarch64/6.1.0/aarch64
GCC_X86_64=/opt/gcc/6.1.0/snos
PE_FFTW_DEFAULT_TARGET_mic_knl=mic_knl
PE_LIBSCI_ACC_DEFAULT_PKGCONFIG_VARIABLES=PE_LIBSCI_⌋
↪  ACC_DEFAULT_NV_SUFFIX_@accelerator@
PE_PETSC_DEFAULT_GENCOMPILERS_INTEL_mic_knl=16.0
PE_TPSL_64_DEFAULT_GENCOMPS_INTEL_interlagos=160
PE_TRILINOS_DEFAULT_GENCOMPS_CRAY_x86_64=86
SLURM_TOPOLOGY_ADDR=s21.s9.nid03508
SLURMD_NODENAME=nid03508
XKEYSYMDB=/usr/X11R6/lib/X11/XKeysymDB
CRAY_SITE_LIST_DIR=/etc/opt/cray/pe/modules

LIBRARYMODULES=acml:alps:cray-dwarf:cray-fftw:cray-g⌋
↪  a:cray-hdf5:cray-hdf5-parallel:cray-libsci:cray-⌋
↪  libsci_acc:cray-mpich:cray-mpich2:cray-mpich-abi⌋
↪  :cray-netcdf:cray-netcdf-hdf5parallel:cray-paral⌋
↪  lel-netcdf:cray-petsc:cray-petsc-complex:cray-sh⌋
↪  mem:cray-tpsl:cray-trilinos:cudatoolkit:fftw:ga:⌋
↪  hdf5:hdf5-parallel:iobuf:libfast:netcdf:netcdf-h⌋
↪  df5parallel:ntk:onesided:papi:petsc:petsc-comple⌋
↪  x:pmi:tpsl:trilinos:xt-libsci:xt-mpich2:xt-mpt:x⌋
↪  t-papi
ASSEMBLER_AARCH64=/opt/cray/pe/cce/8.7.3/binutils/cr⌋
↪  oss/x86_64-aarch64/aarch64-linux-gnu/bin/as
PE_NETCDF_DEFAULT_VOLATILE_PKGCONFIG_PATH=/opt/cray/⌋
↪  pe/netcdf/4.6.1.2/@PRGENV@/@PE_NETCDF_DEFAULT_GE⌋
↪  NCOMPS@/lib/pkgconfig
PE_PARALLEL_NETCDF_DEFAULT_VOLATILE_PKGCONFIG_PATH=/⌋
↪  opt/cray/pe/parallel-netcdf/1.8.1.3/@PRGENV@/@PE⌋
↪  _PARALLEL_NETCDF_DEFAULT_GENCOMPS@/lib/pkgconfig
PE_SMA_DEFAULT_COMPFLAG_GNU=-fcray-pointer
PE_TRILINOS_DEFAULT_VOLATILE_PKGCONFIG_PATH=/opt/cra⌋
↪  y/pe/trilinos/12.12.1.0/@PRGENV@/@PE_TRILINOS_DE⌋
↪  FAULT_GENCOMPS@/@PE_TRILINOS_DEFAULT_TARGET@/lib⌋
↪  /pkgconfig
SLURM_PRIO_PROCESS=0
RCLOCAL_BASEOPTS=true
CRAY_BINUTILS_ROOT=/opt/cray/pe/cce/8.7.3/binutils/x⌋
↪  86_64/x86_64-pc-linux-gnu/../
CRAY_FTN_VERSION=8.7.3
PE_ENV=CRAY
PE_HDF5_DEFAULT_GENCOMPILERS_GNU=7.1 6.1 5.3 4.9
PE_MPICH_ALTERNATE_LIBS_dpm=_dpm
PE_SMA_DEFAULT_COMPFLAG=
PE_TPSL_64_DEFAULT_GENCOMPILERS_CRAY_x86_64=8.6
SLURM_SRUN_COMM_PORT=42848
SHELL=/usr/local/bin/bash
TERM=xterm-256color
HOST=daint103
ASSEMBLER_X86_64=/opt/cray/pe/cce/8.7.3/binutils/x86⌋
↪  _64/x86_64-pc-linux-gnu/bin/as
PE_TPSL_DEFAULT_GENCOMPS_CRAY_x86_skylake=86
PKGCONFIG_ENABLED=1
HISTSIZE=
PROJECT=/project/d69/*REDACTED*
PROFILEREAD=true
LINKER_AARCH64=/opt/cray/pe/cce/8.7.3/binutils/cross⌋
↪  /x86_64-aarch64/aarch64-linux-gnu/bin/ld
PE_PETSC_DEFAULT_GENCOMPS_CRAY_sandybridge=86
PE_TPSL_DEFAULT_GENCOMPILERS_GNU_x86_skylake=7.1 6.1
SLURM_PTY_WIN_ROW=53
SLURM_JOB_QOS=daint_debug
SSH_CLIENT=148.187.1.6 44730 22
CRAYPE_DIR=/opt/cray/pe/craype/2.5.15
CRAY_UGNI_POST_LINK_OPTS=-L/opt/cray/ugni/6.0.14.0-6⌋
↪  .0.7.0_23.1__gea11d3d.ari/lib64
CRAY_XPMEM_POST_LINK_OPTS=-L/opt/cray/xpmem/2.2.15-6⌋
↪  .0.7.1_5.10__g7549d06.ari/lib64
```

```
FORTRAN_SYSTEM_MODULE_NAMES=ftn_lib_definitions
PE_NETCDF_DEFAULT_VOLATILE_PRGENV=GNU
PE_PARALLEL_NETCDF_DEFAULT_VOLATILE_PRGENV=GNU
PE_PETSC_DEFAULT_GENCOMPS_GNU_haswell=71 53 49
PE_PETSC_DEFAULT_GENCOMPS_INTEL_haswell=160
PE_TPSL_64_DEFAULT_GENCOMPS_INTEL_x86_skylake=160
PE_TPSL_DEFAULT_GENCOMPS_GNU_sandybridge=71 53 49
PE_TPSL_DEFAULT_REQUIRED_PRODUCTS=PE_MPICH:PE_LIBSCI
PE_TRILINOS_DEFAULT_VOLATILE_PRGENV=CRAY GNU INTEL
PE_MPICH_DIR_PGI_DEFAULT64=64
SLURM_CSCS=yes
SLURM_TOPOLOGY_ADDR_PATTERN=switch.switch.node
TMPDIR=/tmp
PERL5LIB=/opt/slurm/17.11.12.cscs//lib/perl5/site_pe⌋
↪   rl/5.18.2/x86_64-linux-thread-multi:/opt/slurm/d⌋
↪   efault/lib/perl5/site_perl/5.18.2/x86_64-linux-t⌋
↪   hread-multi:
PE_FFTW_DEFAULT_VOLATILE_PKGCONFIG_PATH=/opt/cray/pe⌋
↪   /fftw/3.3.6.5/@PE_FFTW_DEFAULT_TARGET@/lib/pkgco⌋
↪   nfig
PE_HDF5_DEFAULT_VOLATILE_PRGENV=GNU
PE_HDF5_PARALLEL_DEFAULT_VOLATILE_PKGCONFIG_PATH=/op⌋
↪   t/cray/pe/hdf5-parallel/1.10.2.0/@PRGENV@/@PE_HD⌋
↪   F5_PARALLEL_DEFAULT_GENCOMPS@/lib/pkgconfig
PE_NETCDF_HDF5PARALLEL_DEFAULT_VOLATILE_PKGCONFIG_PA⌋
↪   TH=/opt/cray/pe/netcdf-hdf5parallel/4.6.1.2/@PRG⌋
↪   ENV@/@PE_NETCDF_HDF5PARALLEL_DEFAULT_GENCOMPS@/l⌋
↪   ib/pkgconfig
PE_PETSC_DEFAULT_GENCOMPS_CRAY_interlagos=86
CRAY_MPICH2_DIR=/opt/cray/pe/mpt/7.7.2/gni/mpich-cra⌋
↪   y/8.6
ALT_LINKER=/apps/daint/UES/xalt/0.7.6/bin/ld
LOCAL_PATH=/users/*REDACTED*/.local
INSTALL_DIR=/users/*REDACTED*/.local
PE_GA_DEFAULT_VOLATILE_PRGENV=GNU
PE_LIBSCI_DEFAULT_GENCOMPS_GNU_x86_64=71 61 51 49
PE_TPSL_64_DEFAULT_GENCOMPILERS_CRAY_interlagos=8.6
PE_TPSL_DEFAULT_GENCOMPS_CRAY_mic_knl=86
PMI_CONTROL_PORT=25805
MORE=-sl
FPATH=:/opt/cray/pe/modules/3.2.10.6/init/sh_funcs/n⌋
↪   o_redirect:/opt/cray/pe/modules/3.2.10.6/init/sh⌋
↪   _funcs/no_redirect
PERFTOOLS_VERSION=7.0.2
PE_LIBSCI_ACC_DEFAULT_GENCOMPS_CRAY_x86_64=85
PE_LIBSCI_ACC_DEFAULT_REQUIRED_PRODUCTS=PE_MPICH:PE_⌋
↪   LIBSCI
PE_MPICH_DEFAULT_GENCOMPILERS_GNU=7.1 5.1 4.9
PE_PKGCONFIG_PRODUCTS=PE_MPICH:PE_LIBSCI
PE_TPSL_DEFAULT_GENCOMPS_INTEL_x86_64=160
PE_MPICH_GENCOMPS_GNU=71 51 49
SLURM_CPU_BIND_VERBOSE=quiet
PE_PAPI_DEFAULT_ACCEL_LIBS_nvidia35=,-lcupti,-lcudar⌋
↪   t,-lcuda
PE_PETSC_DEFAULT_REQUIRED_PRODUCTS=PE_MPICH:PE_LIBSC⌋
↪   I:PE_HDF5_PARALLEL:PE_TPSL
```

```
PE_TPSL_64_DEFAULT_GENCOMPS_CRAY_haswell=86
PE_TPSL_64_DEFAULT_VOLATILE_PKGCONFIG_PATH=/opt/cray⌋
↪   /pe/tpsl/18.06.1/@PRGENV@64/@PE_TPSL_64_DEFAULT_⌋
↪   GENCOMPS@/@PE_TPSL_64_DEFAULT_TARGET@/lib/pkgcon⌋
↪   fig
PE_CRAY_DEFAULT_FIXED_PKGCONFIG_PATH=/opt/cray/pe/pa⌋
↪   rallel-netcdf/1.8.1.3/CRAY/8.6/lib/pkgconfig:/op⌋
↪   t/cray/pe/netcdf-hdf5parallel/4.6.1.2/CRAY/8.6/l⌋
↪   ib/pkgconfig:/opt/cray/pe/netcdf/4.6.1.2/CRAY/8.⌋
↪   6/lib/pkgconfig:/opt/cray/pe/hdf5-parallel/1.10.⌋
↪   2.0/CRAY/8.6/lib/pkgconfig:/opt/cray/pe/hdf5/1.1⌋
↪   0.2.0/CRAY/8.6/lib/pkgconfig:/opt/cray/pe/ga/5.3⌋
↪   .0.8/CRAY/8.6/lib/pkgconfig
PE_TRILINOS_DEFAULT_GENCOMPILERS_CRAY_x86_64=8.6
SSH_TTY=/dev/pts/7
PE_LIBSCI_DEFAULT_OMP_REQUIRES_openmp=_mp
PE_PETSC_DEFAULT_GENCOMPS_CRAY_x86_64=86
PE_TPSL_64_DEFAULT_GENCOMPILERS_CRAY_sandybridge=8.6
cce_already_loaded=0
PE_FORTRAN_PKGCONFIG_LIBS=mpichf90
CRAYPAT_ALPS_COMPONENT=/opt/cray/pe/perftools/7.0.2/⌋
↪   sbin/pat_alps
CRAYPAT_LD_LIBRARY_PATH=/opt/cray/pe/gcc-libs:/opt/c⌋
↪   ray/gcc-libs:/opt/cray/pe/perftools/7.0.2/lib64
CRAY_BINUTILS_ROOT_AARCH64=/opt/cray/pe/cce/8.7.3/bi⌋
↪   nutils/cross/x86_64-aarch64/aarch64-linux-gnu/../
CRAY_BINUTILS_VERSION=/opt/cray/pe/cce/8.7.3
CRAY_PRGENVCRAY=loaded
PE_SMA_DEFAULT_VOLATILE_PKGCONFIG_PATH=/opt/cray/pe/⌋
↪   mpt/7.7.2/gni/sma@PE_SMA_DEFAULT_DIR_DEFAULT64@/⌋
↪   lib64/pkgconfig
ALLINEA_QUEUE_DLL=/opt/cray/pe/mpt/7.7.2/gni/mpich-c⌋
↪   ray/8.6/lib/libtvmpich.so.3.0.1
SLURM_CPU_BIND_LIST=0xFFFFFF
PE_LIBSCI_ACC_DEFAULT_VOLATILE_PRGENV=CRAY GNU
PE_TRILINOS_DEFAULT_GENCOMPS_INTEL_x86_64=160
CRAY_MPICH_BASEDIR=/opt/cray/pe/mpt/7.7.2/gni
ALPS_APP_ID=12773565
USER=*REDACTED*
JRE_HOME=/usr/lib64/jvm/java/jre
PE_HDF5_PARALLEL_DEFAULT_GENCOMPILERS_GNU=7.1 6.1 5.3⌋
↪   4.9
PE_NETCDF_HDF5PARALLEL_DEFAULT_GENCOMPILERS_GNU=7.1⌋
↪   6.1 5.3 4.9
PE_TPSL_64_DEFAULT_GENCOMPS_CRAY_x86_skylake=86
PE_TPSL_64_DEFAULT_GENCOMPS_INTEL_haswell=160
SLURM_NNODES=1
SLURM_LOG_ACTIONS=yes
```

```
LS_COLORS=no=00:fi=00:di=01;34:ln=00;36:pi=40;33:so=⌋
↪   01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=41;33;⌋
↪   01:ex=00;32:*.cmd=00;32:*.exe=01;32:*.com=01;32:⌋
↪   *.bat=01;32:*.btm=01;32:*.dll=01;32:*.tar=00;31:⌋
↪   *.tbz=00;31:*.tgz=00;31:*.rpm=00;31:*.deb=00;31:⌋
↪   *.arj=00;31:*.taz=00;31:*.lzh=00;31:*.lzma=00;31⌋
↪   :*.zip=00;31:*.zoo=00;31:*.z=00;31:*.Z=00;31:*.g⌋
↪   z=00;31:*.bz2=00;31:*.tb2=00;31:*.tz2=00;31:*.tb⌋
↪   z2=00;31:*.xz=00;31:*.avi=01;35:*.bmp=01;35:*.fl⌋
↪   i=01;35:*.gif=01;35:*.jpg=01;35:*.jpeg=01;35:*.m⌋
↪   ng=01;35:*.mov=01;35:*.mpg=01;35:*.pcx=01;35:*.p⌋
↪   bm=01;35:*.pgm=01;35:*.png=01;35:*.ppm=01;35:*.t⌋
↪   ga=01;35:*.tif=01;35:*.xbm=01;35:*.xpm=01;35:*.d⌋
↪   l=01;35:*.gl=01;35:*.wmv=01;35:*.aiff=00;32:*.au⌋
↪   =00;32:*.mid=00;32:*.mp3=00;32:*.ogg=00;32:*.voc⌋
↪   =00;32:*.wav=00;32:
LD_LIBRARY_PATH=/users/*REDACTED*/.local/lib:/users/⌋
↪   *REDACTED*/.local/lib:/opt/cray/pe/papi/5.6.0.2/⌋
↪   lib64:/opt/cray/job/2.2.3-6.0.7.0_44.1__g6c4e934⌋
↪   .ari/lib64
PE_FFTW_DEFAULT_TARGET_interlagos=interlagos
PE_LIBSCI_DEFAULT_VOLATILE_PRGENV=CRAY GNU INTEL
PE_PETSC_DEFAULT_GENCOMPILERS_INTEL_interlagos=16.0
PE_TPSL_64_DEFAULT_GENCOMPILERS_INTEL_mic_knl=16.0
PE_TPSL_DEFAULT_GENCOMPS_CRAY_x86_64=86
PE_TRILINOS_DEFAULT_GENCOMPILERS_GNU_x86_64=71 53 49
PE_TRILINOS_DEFAULT_GENCOMPILERS_INTEL_x86_64=160
SINFO_FORMAT=%9P %5a %8s %.10l %.6c %.6z %.7D %10T %N
CRAY_RCA_POST_LINK_OPTS=-L/opt/cray/rca/2.2.18-6.0.7⌋
↪   .0_33.3__g2aa4f39.ari/lib64
↪   -lrca
PE_LIBSCI_PKGCONFIG_VARIABLES=PE_LIBSCI_OMP_REQUIRES⌋
↪   _@openmp@:PE_SCI_EXT_LIBPATH:PE_SCI_EXT_LIBNAME
PE_PETSC_DEFAULT_VOLATILE_PRGENV=CRAY CRAY64 GNU
↪   GNU64 INTEL INTEL64
PE_PKGCONFIG_LIBS=mpich:AtpSigHandler:cray-rca:libsc⌋
↪   i_mpi:libsci
PE_TPSL_64_DEFAULT_GENCOMPILERS_GNU_sandybridge=7.1
↪   5.3 4.9
PE_TPSL_64_DEFAULT_GENCOMPILERS_INTEL_haswell=16.0
PE_MPICH_FIXED_PRGENV=INTEL
CRAY_IAA_INFO_FILE=/tmp/cray_iaa_info.12773565
XNLSPATH=/usr/share/X11/nls
FTN_X86_64=/opt/cray/pe/cce/8.7.3/cce/x86_64
PE_PETSC_DEFAULT_GENCOMPILERS_CRAY_mic_knl=8.6
PE_PETSC_DEFAULT_GENCOMPILERS_CRAY_x86_64=8.6
PE_PETSC_DEFAULT_GENCOMPILERS_INTEL_skylake=16.0
PE_PETSC_DEFAULT_GENCOMPS_GNU_interlagos=71 53 49
PE_PETSC_DEFAULT_GENCOMPS_GNU_sandybridge=71 53 49
PE_PETSC_DEFAULT_GENCOMPS_INTEL_interlagos=160
PE_PETSC_DEFAULT_GENCOMPS_INTEL_sandybridge=160
PE_TPSL_DEFAULT_GENCOMPS_GNU_haswell=71 53 49
CRAY_CXX_IPA_LIBS=/opt/cray/pe/cce/8.7.3/cce/x86_64/⌋
↪   lib/libcray-c++-rts.a
MPICH_ABORT_ON_ERROR=1
PE_LIBSCI_DEFAULT_GENCOMPS_CRAY_x86_64=86
PE_PAPI_DEFAULT_PKGCONFIG_VARIABLES=PE_PAPI_ACCEL_LI⌋
↪   BS_@accelerator@
PE_PETSC_DEFAULT_GENCOMPILERS_CRAY_haswell=8.6
PE_PETSC_DEFAULT_GENCOMPS_GNU_mic_knl=53
PE_PETSC_DEFAULT_GENCOMPS_INTEL_mic_knl=160
PE_TPSL_64_DEFAULT_GENCOMPILERS_GNU_interlagos=7.1
↪   5.3 4.9
PE_TPSL_64_DEFAULT_GENCOMPS_INTEL_sandybridge=160
MPICH_DIR=/opt/cray/pe/mpt/7.7.2/gni/mpich-cray/8.6
SLURM_STEP_NUM_NODES=1
HOSTTYPE=x86_64
ATP_POST_LINK_OPTS=-Wl,-L/opt/cray/pe/atp/2.1.2/libA⌋
↪   pp/
PE_FFTW_DEFAULT_REQUIRED_PRODUCTS=PE_MPICH
PE_FFTW_DEFAULT_TARGET_sandybridge=sandybridge
PE_HDF5_PARALLEL_DEFAULT_REQUIRED_PRODUCTS=PE_MPICH
PE_NETCDF_HDF5PARALLEL_DEFAULT_REQUIRED_PRODUCTS=PE_⌋
↪   HDF5_PARALLEL
PE_PETSC_DEFAULT_GENCOMPILERS_INTEL_sandybridge=16.0
PE_TPSL_64_DEFAULT_GENCOMPILERS_CRAY_haswell=8.6
PE_MPICH_FORTRAN_PKGCONFIG_LIBS=mpichf90
CPATH=/users/*REDACTED*/.local/include:/users/*REDAC⌋
↪   TED*/.local/include:
SRUN_DEBUG=3
SLURM_JOBID=12773565
TMOUT=259200
PE_PETSC_DEFAULT_GENCOMPILERS_GNU_mic_knl=5.3
RCLOCAL_PRGENV=true
APPS=/apps/daint
FROM_HEADER=
CHPL_CG_CPP_LINES=1
OFFLOAD_INIT=on_start
PE_LIBSCI_DEFAULT_GENCOMPILERS_INTEL_x86_64=16.0
PE_LIBSCI_GENCOMPS_INTEL_x86_64=160
PE_PRODUCT_LIST=CRAYPE_HASWELL:CRAY_RCA:CRAY_ALPS:DV⌋
↪   S:CRAY_XPMEM:CRAY_DMAPP:CRAY_PMI:CRAY_UGNI:CRAY_⌋
↪   UDREG:CRAY_LIBSCI:CRAYPE:CRAY:PERFTOOLS:CRAYPAT
PE_TPSL_DEFAULT_GENCOMPILERS_CRAY_x86_64=8.6
PE_TPSL_DEFAULT_GENCOMPS_GNU_interlagos=71 53 49
SLURM_NTASKS=1
PAGER=less
PE_MPICH_DEFAULT_GENCOMPS_PGI=153
PE_PETSC_DEFAULT_GENCOMPILERS_GNU_x86_64=7.1 5.3 4.9
PE_TPSL_DEFAULT_GENCOMPS_GNU_x86_skylake=71 61
CRAY_MPICH_ROOTDIR=/opt/cray/pe/mpt/7.7.2
SLURM_LAUNCH_NODE_IPADDR=148.187.26.66
ALPS_LLI_STATUS_OFFSET=1
CSHEDIT=emacs
PE_LIBSCI_GENCOMPILERS_GNU_x86_64=7.1 6.1 5.1 4.9
PE_PETSC_DEFAULT_GENCOMPS_GNU_skylake=61
PE_PETSC_DEFAULT_GENCOMPS_INTEL_skylake=160
PE_TPSL_64_DEFAULT_GENCOMPILERS_INTEL_x86_64=16.0
PE_MPICH_GENCOMPILERS_CRAY=8.6
PE_MPICH_MODULE_NAME=cray-mpich
SLURM_STEP_ID=0
ALPS_APP_PE=0
```

```
XDG_CONFIG_DIRS=/etc/xdg
CRAYPAT_ROOT=/opt/cray/pe/perftools/7.0.2
PE_LIBSCI_DEFAULT_GENCOMPILERS_CRAY_x86_64=8.6
PE_LIBSCI_GENCOMPS_CRAY_x86_64=86
PE_MPICH_DEFAULT_VOLATILE_PRGENV=CRAY GNU PGI
PE_MPICH_TARGET_VAR_nvidia20=-lcudart
PE_TPSL_64_DEFAULT_REQUIRED_PRODUCTS=PE_MPICH:PE_LIB
↪  SCI
PE_TPSL_DEFAULT_GENCOMPS_CRAY_haswell=86
PE_TPSL_DEFAULT_GENCOMPS_CRAY_sandybridge=86
MINICOM=-c on
LIBGL_DEBUG=quiet
USERMODULES=acml:alps:apprentice:apprentice2:atp:blc
↪  r:cce:chapel:cray-ccdb:cray-fftw:cray-ga:cray-hd
↪  f5:cray-hdf5-parallel:cray-lgdb:cray-libsci:cray
↪  -libsci_acc:cray-mpich:cray-mpich2:cray-mpich-com
↪  pat:cray-netcdf:cray-netcdf-hdf5parallel:cray-pa
↪  rallel-netcdf:craypat:craype:cray-petsc:cray-pet
↪  sc-complex:craypkg-gen:cray-shmem:cray-snplaunch
↪  er:cray-tpsl:cray-trilinos:cudatoolkit:ddt:fftw:
↪  ga:gcc:hdf5:hdf5-parallel:intel:iobuf:java:lgdb:
↪  libfast:libsci_acc:mpich1:netcdf:netcdf-hdf5para
↪  llel:netcdf-nofsync:netcdf-nofsync-hdf5parallel:
↪  ntk:onesided:papi:parallel-netcdf:pathscale:perf
↪  tools:perftools-lite:petsc:petsc-complex:pgi:pmi
↪  :PrgEnv-cray:PrgEnv-gnu:PrgEnv-intel:PrgEnv-path
↪  scale:PrgEnv-pgi:stat:totalview:tpsl:trilinos:xt
↪  -asyncpe:xt-craypat:xt-lgdb:xt-libsci:xt-mpich2:x
↪  t-mpt:xt-papi:xt-shmem:xt-totalview
CRAY_DMAPP_INCLUDE_OPTS=-I/opt/cray/dmapp/7.1.1-6.0.
↪  7.0_34.3__g5a674e0.ari/include
↪  -I/opt/cray/gni-headers/5.0.12.0-6.0.7.0_24.1__g
↪  3b1768f.ari/include
CRAY_LIBSCI_BASE_DIR=/opt/cray/pe/libsci/18.07.1
CRAY_LIBSCI_DIR=/opt/cray/pe/libsci/18.07.1
DVS_VERSION=0.9.0
NLSPATH=/opt/cray/pe/cce/8.7.3/cce/x86_64/share/nls/
↪  En/%N.cat
PE_LIBSCI_ACC_DEFAULT_VOLATILE_PKGCONFIG_PATH=/opt/c
↪  ray/pe/libsci_acc/18.07.1/@PRGENV@/@PE_LIBSCI_AC
↪  C_DEFAULT_GENCOMPS@/@PE_LIBSCI_ACC_DEFAULT_TARGE
↪  T@/lib/pkgconfig
PE_LIBSCI_PKGCONFIG_LIBS=libsci_mpi:libsci
PE_NETCDF_DEFAULT_GENCOMPS_GNU=
PE_PARALLEL_NETCDF_DEFAULT_GENCOMPS_GNU=51 49
PE_TPSL_64_DEFAULT_GENCOMPS_GNU_mic_knl=71 53
PE_TPSL_64_DEFAULT_GENCOMPS_GNU_x86_64=71 53 49

PATH=/users/*REDACTED*/.local/bin:/users/*REDACTED*/
↪  .local/bin:/apps/daint/UES/xalt/0.7.6/bin:/opt/s
↪  lurm/17.11.12.cscs/bin:/opt/cray/pe/mpt/7.7.2/gn
↪  i/bin:/opt/cray/pe/perftools/7.0.2/bin:/opt/cray
↪  /pe/papi/5.6.0.2/bin:/opt/cray/rca/2.2.18-6.0.7.
↪  0_33.3__g2aa4f39.ari/bin:/opt/cray/alps/6.6.43-6
↪  .0.7.0_26.4__ga796da3.ari/sbin:/opt/cray/job/2.2
↪  .3-6.0.7.0_44.1__g6c4e934.ari/bin:/opt/cray/pe/c
↪  raype/2.5.15/bin:/opt/cray/pe/cce/8.7.3/binutils
↪  /x86_64/x86_64-pc-linux-gnu/bin:/opt/cray/pe/cce
↪  /8.7.3/binutils/cross/x86_64-aarch64/aarch64-lin
↪  ux-gnu/../bin:/opt/cray/pe/cce/8.7.3/utils/x86_6
↪  4/bin:/opt/cray/pe/modules/3.2.10.6/bin:/opt/slu
↪  rm/default/bin:/apps/daint/system/bin:/apps/comm
↪  on/system/bin:/usr/local/bin:/usr/bin:/bin:/usr/
↪  bin/X11:/usr/lib/mit/bin:/usr/lib/mit/sbin:/opt/
↪  cray/pe/bin
MAIL=/var/mail/*REDACTED*
MODULE_VERSION=3.2.10.6
PAT_REPORT_PRUNE_NAME=_cray$mt_execute_,_cray$mt_sta
↪  rt_,__cray_hwpc_,f_cray_hwpc_,cstart,__pat_,pat_
↪  region_,PAT_,OMP.slave_loop,slave_entry,_new_sla
↪  ve_entry,_thread_pool_slave_entry,THREAD_POOL_jo
↪  in,__libc_start_main,_start,__start,start_thread
↪  ,__wrap_,UPC_ADIO_,_upc_,upc_,__caf_,__pgas_,sys
↪  call,__device_stub
PE_HDF5_DEFAULT_VOLATILE_PKGCONFIG_PATH=/opt/cray/pe
↪  /hdf5/1.10.2.0/@PRGENV@/@PE_HDF5_DEFAULT_GENCOMP
↪  S@/lib/pkgconfig
PE_PKGCONFIG_DEFAULT_PRODUCTS=PE_TRILINOS:PE_TPSL_64
↪  :PE_TPSL:PE_PETSC:PE_PARALLEL_NETCDF:PE_NETCDF_H
↪  DF5PARALLEL:PE_NETCDF:PE_MPICH:PE_LIBSCI_ACC:PE_
↪  LIBSCI:PE_HDF5_PARALLEL:PE_HDF5:PE_GA:PE_FFTW
PE_TPSL_DEFAULT_GENCOMPILERS_GNU_x86_64=7.1 5.3 4.9
PE_TPSL_DEFAULT_GENCOMPS_CRAY_interlagos=86
PE_MPICH_GENCOMPILERS_GNU=7.1 5.1 4.9
SLURM_TASKS_PER_NODE=1
SLURM_STEP_LAUNCHER_PORT=42848
CPU=x86_64
CSCS_CUSTOM_ENV=true
XTPE_NETWORK_TARGET=aries
ATP_IGNORE_SIGTERM=1
PE_FFTW_DEFAULT_TARGET_abudhabi=abudhabi
PE_MPICH_DEFAULT_DIR_PGI_DEFAULT64=64
PE_NETCDF_DEFAULT_GENCOMPILERS_GNU=7.1 6.1 5.3 4.9
PE_PARALLEL_NETCDF_DEFAULT_GENCOMPILERS_GNU=5.1 4.9
PE_PETSC_DEFAULT_GENCOMPS_CRAY_mic_knl=86
PE_TPSL_64_DEFAULT_GENCOMPILERS_GNU_x86_skylake=7.1
↪  6.1
PE_TPSL_DEFAULT_GENCOMPILERS_GNU_haswell=7.1 5.3 4.9
SLURM_WORKING_CLUSTER=daint:daintsl01:6817:8192
PMI_CRAY_NO_SMP_ORDER=0
SSH_SENDS_LOCALE=yes
JAVA_BINDIR=/usr/lib64/jvm/java/bin
SQUEUE_SORT=-t,e,S
```

```
CRAY_CCE_SHARE=/opt/cray/pe/cce/8.7.3/cce/x86_64/sha
↪   re
CRAY_CXX_IPA_LIBS_AARCH64=/opt/cray/pe/cce/8.7.3/cce
↪   /aarch64/lib/libcray-c++-rts.a
PE_HDF5_PARALLEL_DEFAULT_FIXED_PRGENV=CRAY PGI INTEL
PE_HDF5_PARALLEL_DEFAULT_GENCOMPS_GNU=
PE_NETCDF_HDF5PARALLEL_DEFAULT_FIXED_PRGENV=CRAY PGI
↪   INTEL
PE_NETCDF_HDF5PARALLEL_DEFAULT_GENCOMPS_GNU=
PE_SMA_DEFAULT_DIR_CRAY_DEFAULT64=64
PE_TPSL_64_DEFAULT_GENCOMPILERS_CRAY_x86_skylake=8.6
SLURM_JOB_ID=12773565
PMI_NO_FORK=1
CRAY_UDREG_POST_LINK_OPTS=-L/opt/cray/udreg/2.3.2-6.
↪   0.7.0_33.18__g5196236.ari/lib64
PE_TPSL_64_DEFAULT_GENCOMPS_CRAY_sandybridge=86
PE_TPSL_64_DEFAULT_VOLATILE_PRGENV=CRAY CRAY64 GNU
↪   GNU64 INTEL INTEL64
PE_TPSL_DEFAULT_GENCOMPILERS_CRAY_mic_knl=8.6
PE_TPSL_DEFAULT_GENCOMPS_INTEL_interlagos=160
LD_RUN_PATH=/users/*REDACTED*/.local/lib:/users/*RED
↪   ACTED*/.local/lib:/users/*REDACTED*/.local/lib:/
↪   opt/cray/pe/papi/5.6.0.2/lib64:/opt/cray/job/2.2
↪   .3-6.0.7.0_44.1__g6c4e934.ari/lib64
SLURM_STEP_GPUS=0
PWD=/users/*REDACTED*
INPUTRC=/users/*REDACTED*/.inputrc
CRAYPE_VERSION=2.5.15
CRAY_ALPS_POST_LINK_OPTS=-L/opt/cray/alps/6.6.43-6.0
↪   .7.0_26.4__ga796da3.ari/lib64
PE_TPSL_DEFAULT_GENCOMPS_GNU_mic_knl=71 53
PE_MPICH_VOLATILE_PRGENV=CRAY GNU PGI
SLURM_STEPID=0
SLURM_JOB_USER=*REDACTED*
JAVA_HOME=/usr/lib64/jvm/java
TARGETMODULES=craype-abudhabi:craype-abudhabi-cu:cra
↪   ype-accel-host:craype-accel-nvidia20:craype-acce
↪   l-nvidia30:craype-accel-nvidia35:craype-barcelon
↪   a:craype-broadwell:craype-haswell:craype-hugepag
↪   es128K:craype-hugepages128M:craype-hugepages16M:
↪   craype-hugepages256M:craype-hugepages2M:craype-h
↪   ugepages32M:craype-hugepages4M:craype-hugepages5
↪   12K:craype-hugepages512M:craype-hugepages64M:cra
↪   ype-hugepages8M:craype-intel-knc:craype-interlag
↪   os:craype-interlagos-cu:craype-istanbul:craype-i
↪   vybridge:craype-mc12:craype-mc8:craype-mic-knl:c
↪   raype-network-aries:craype-network-gemini:craype
↪   -network-infiniband:craype-network-none:craype-ne
↪   twork-seastar:craype-sandybridge:craype-shanghai
↪   :craype-target-compute_node:craype-target-local_
↪   host:craype-target-native:craype-xeon:xtpe-barce
↪   lona:xtpe-interlagos:xtpe-interlagos-cu:xtpe-ist
↪   anbul:xtpe-mc12:xtpe-mc8:xtpe-network-gemini:xtp
↪   e-network-seastar:xtpe-shanghai:xtpe-target-nati
↪   ve:xtpe-xeon
_LMFILES_=/opt/cray/pe/modulefiles/modules/3.2.10.6:
↪   /opt/cray/pe/modulefiles/cce/8.7.3:/opt/cray/pe/
↪   craype/2.5.15/modulefiles/craype-network-aries:/
↪   opt/cray/pe/modulefiles/craype/2.5.15:/opt/cray/
↪   pe/modulefiles/cray-libsci/18.07.1:/opt/cray/ari
↪   /modulefiles/udreg/2.3.2-6.0.7.0_33.18__g5196236
↪   .ari:/opt/cray/ari/modulefiles/ugni/6.0.14.0-6.0
↪   .7.0_23.1__gea11d3d.ari:/opt/cray/pe/modulefiles
↪   /pmi/5.0.14:/opt/cray/ari/modulefiles/dmapp/7.1.
↪   1-6.0.7.0_34.3__g5a674e0.ari:/opt/cray/ari/modul
↪   efiles/gni-headers/5.0.12.0-6.0.7.0_24.1__g3b176
↪   8f.ari:/opt/cray/ari/modulefiles/xpmem/2.2.15-6.
↪   0.7.1_5.10__g7549d06.ari:/opt/cray/ari/modulefil
↪   es/job/2.2.3-6.0.7.0_44.1__g6c4e934.ari:/opt/cra
↪   y/ari/modulefiles/dvs/2.7_2.2.113-6.0.7.1_7.6__g
↪   1bbc03e:/opt/cray/ari/modulefiles/alps/6.6.43-6.
↪   0.7.0_26.4__ga796da3.ari:/opt/cray/ari/modulefil
↪   es/rca/2.2.18-6.0.7.0_33.3__g2aa4f39.ari:/opt/cr
↪   ay/pe/modulefiles/atp/2.1.2:/opt/cray/pe/modulef
↪   iles/perftools-base/7.0.2:/opt/cray/pe/modulefil
↪   es/PrgEnv-cray/6.0.4:/opt/cray/pe/modulefiles/cr
↪   ay-mpich/7.7.2:/opt/modulefiles/slurm/17.11.12.c
↪   scs-1:/opt/cray/pe/craype/2.5.15/modulefiles/cra
↪   ype-haswell:/apps/daint/UES/easybuild/modulefile
↪   s/xalt/daint-2016.11:/opt/modulefiles/Base-opts/
↪   2.4.135-6.0.7.0_38.1__g718f891.ari
INCLUDE_PATH_X86_64=/opt/cray/pe/cce/8.7.3/cce/x86_6
↪   4/include/craylibs
PE_LIBSCI_DEFAULT_OMP_REQUIRES=
PE_MPICH_DEFAULT_GENCOMPS_CRAY=86
PE_PETSC_DEFAULT_GENCOMPILERS_GNU_sandybridge=7.1
↪   5.3 4.9
PE_TPSL_DEFAULT_GENCOMPILERS_INTEL_haswell=16.0
XALT_TRANSMISSION_STYLE=directdb
SLURM_SRUN_COMM_HOST=148.187.26.66
CUDA_VISIBLE_DEVICES=0
PE_LIBSCI_ACC_DEFAULT_NV_SUFFIX_nvidia20=nv20
PE_LIBSCI_MODULE_NAME=cray-libsci/18.07.1
PE_PETSC_DEFAULT_GENCOMPILERS_CRAY_skylake=8.6
PE_TPSL_DEFAULT_GENCOMPILERS_CRAY_interlagos=8.6
PE_TPSL_DEFAULT_GENCOMPILERS_GNU_mic_knl=7.1 5.3
SLURM_CPU_BIND_TYPE=mask_cpu:
LANG=en_US.UTF-8
PE_TPSL_64_DEFAULT_GENCOMPS_GNU_x86_skylake=71 61
PE_INTEL_FIXED_PKGCONFIG_PATH=/opt/cray/pe/mpt/7.7.2
↪   /gni/mpich-intel/16.0/lib/pkgconfig
PYTHONSTARTUP=/etc/pythonstart
MODULEPATH=/apps/daint/modulefiles:/apps/daint/syste
↪   m/modulefiles:/apps/daint/UES/easybuild/modulefi
↪   les:/apps/common/UES/modulefiles:/apps/common/sy
↪   stem/modulefiles:/opt/cray/pe/perftools/7.0.2/mo
↪   dulefiles:/opt/cray/pe/craype/2.5.15/modulefiles
↪   :/opt/cray/pe/modulefiles:/opt/cray/modulefiles:
↪   /opt/modulefiles:/opt/cray/ari/modulefiles:/opt/
↪   cray/pe/ari/modulefiles
PE_LIBSCI_GENCOMPILERS_CRAY_x86_64=8.6
```

PE_MPICH_NV_LIBS_nvidia20=-lcudart
PE_MPICH_VOLATILE_PKGCONFIG_PATH=/opt/cray/pe/mpt/7.⌋
↪ 7.2/gni/mpich-@PRGENV@@PE_MPICH_DIR_DEFAULT64@/@⌋
↪ PE_MPICH_GENCOMPS@/lib/pkgconfig
SLURM_UMASK=0022
SLURM_PTY_WIN_COL=181
SDK_HOME=/usr/lib64/jvm/java
TZ=Europe/Zurich
LOADEDMODULES=modules/3.2.10.6:cce/8.7.3:craype-netw⌋
↪ ork-aries:craype/2.5.15:cray-libsci/18.07.1:udre⌋
↪ g/2.3.2-6.0.7.0_33.18__g5196236.ari:ugni/6.0.14.⌋
↪ 0-6.0.7.0_23.1__gea11d3d.ari:pmi/5.0.14:dmapp/7.⌋
↪ 1.1-6.0.7.0_34.3__g5a674e0.ari:gni-headers/5.0.1⌋
↪ 2.0-6.0.7.0_24.1__g3b1768f.ari:xpmem/2.2.15-6.0.⌋
↪ 7.1_5.10__g7549d06.ari:job/2.2.3-6.0.7.0_44.1__g⌋
↪ 6c4e934.ari:dvs/2.7_2.2.113-6.0.7.1_7.6__g1bbc03⌋
↪ e:alps/6.6.43-6.0.7.0_26.4__ga796da3.ari:rca/2.2⌋
↪ .18-6.0.7.0_33.3__g2aa4f39.ari:atp/2.1.2:perftoo⌋
↪ ls-base/7.0.2:PrgEnv-cray/6.0.4:cray-mpich/7.7.2⌋
↪ :slurm/17.11.12.cscs-1:craype-haswell:xalt/daint⌋
↪ -2016.11:Base-opts/2.4.135-6.0.7.0_38.1__g718f891⌋
↪ .ari
SHMEM_ABORT_ON_ERROR=1
SLURM_JOB_UID=23992
CRAY_BINUTILS_ROOT_X86_64=/opt/cray/pe/cce/8.7.3/bin⌋
↪ utils/x86_64/x86_64-pc-linux-gnu/../
CRAY_DMAPP_POST_LINK_OPTS=-L/opt/cray/dmapp/7.1.1-6.⌋
↪ 0.7.0_34.3__g5a674e0.ari/lib64
PE_FFTW_DEFAULT_TARGET_ivybridge=ivybridge
PE_FFTW_DEFAULT_TARGET_share=share
PE_FFTW_DEFAULT_TARGET_x86_skylake=x86_skylake
PE_PKG_CONFIG_PATH=/opt/cray/pe/cti/1.0.7/lib/pkgcon⌋
↪ fig:/opt/cray/pe/cti/1.0.6/lib/pkgconfig:/opt/cr⌋
↪ ay/pe/cti/1.0.4/lib/pkgconfig
PE_TPSL_64_DEFAULT_GENCOMPS_GNU_interlagos=71 53 49
PE_TPSL_DEFAULT_GENCOMPILERS_INTEL_mic_knl=16.0
SLURM_NODEID=0
CRAY_RCA_INCLUDE_OPTS=-I/opt/cray/rca/2.2.18-6.0.7.0⌋
↪ _33.3__g2aa4f39.ari/include
↪ -I/opt/cray/krca/2.2.4-6.0.7.1_5.27__g8505b97.ar⌋
↪ i/include
↪ -I/opt/cray-hss-devel/8.0.0/include
PAT_BUILD_PAPI_BASEDIR=/opt/cray/pe/papi/5.6.0.2
PE_LIBSCI_OMP_REQUIRES_openmp=_mp
PE_PETSC_DEFAULT_GENCOMPILERS_GNU_skylake=6.1
PE_TPSL_DEFAULT_GENCOMPILERS_CRAY_x86_skylake=8.6
SLURM_SUBMIT_DIR=/users/*REDACTED*
SLURM_STEP_RESV_PORTS=25805
PE_TPSL_64_DEFAULT_GENCOMPS_CRAY_mic_knl=86
PE_TPSL_DEFAULT_GENCOMPILERS_INTEL_x86_64=16.0
CRAY_MPICH_DIR=/opt/cray/pe/mpt/7.7.2/gni/mpich-cray⌋
↪ /8.6
PE_MPICH_CXX_PKGCONFIG_LIBS=mpichcxx
SLURM_NPROCS=1
SLURM_TASK_PID=17392

SQUEUE_FORMAT=%.8i %.8u %.7a %.14j %.3t %9r %19S
↪ %.10M %.10L %.5D %.4C
CRAY_BINUTILS_BIN_AARCH64=/opt/cray/pe/cce/8.7.3/bin⌋
↪ utils/cross/x86_64-aarch64/aarch64-linux-gnu/bin
PE_LIBSCI_ACC_DEFAULT_GENCOMPILERS_GNU_x86_64=4.9
PE_LIBSCI_DEFAULT_GENCOMPS_INTEL_x86_64=160
PE_MPICH_PKGCONFIG_VARIABLES=PE_MPICH_NV_LIBS_@accel⌋
↪ erator@:PE_MPICH_ALTERNATE_LIBS_@multithreaded@:⌋
↪ PE_MPICH_ALTERNATE_LIBS_@dpm@
SLURM_DISTRIBUTION=cyclic
SLURM_CPUS_ON_NODE=24
APP2_STATE=7.0.2
CRAY_CC_VERSION=8.7.3
CRAY_PMI_POST_LINK_OPTS=-L/opt/cray/pe/pmi/5.0.14/li⌋
↪ b64
PE_HDF5_DEFAULT_FIXED_PRGENV=CRAY PGI INTEL
PE_TPSL_64_DEFAULT_GENCOMPILERS_CRAY_mic_knl=8.6
PE_TPSL_DEFAULT_GENCOMPILERS_INTEL_x86_skylake=16.0
PE_TPSL_DEFAULT_VOLATILE_PKGCONFIG_PATH=/opt/cray/pe⌋
↪ /tpsl/18.06.1/@PRGENV@/@PE_TPSL_DEFAULT_GENCOMPS⌋
↪ @/@PE_TPSL_DEFAULT_TARGET@/lib/pkgconfig
CRAY_MPICH2_VER=7.7.2
PE_MPICH_PKGCONFIG_LIBS=mpich
SLURM_PROCID=0
GPG_TTY=/dev/pts/0
PE_GA_DEFAULT_GENCOMPILERS_GNU=5.3 4.9
PE_LIBSCI_ACC_DEFAULT_GENCOMPS_GNU_x86_64=49
PE_LIBSCI_VOLATILE_PKGCONFIG_PATH=/opt/cray/pe/libsc⌋
↪ i/18.07.1/@PRGENV@/@PE_LIBSCI_GENCOMPS@/@PE_LIBS⌋
↪ CI_TARGET@/lib/pkgconfig
PE_MPICH_ALTERNATE_LIBS_multithreaded=_mt
PE_NETCDF_DEFAULT_FIXED_PRGENV=CRAY PGI INTEL
PE_PARALLEL_NETCDF_DEFAULT_FIXED_PRGENV=CRAY PGI
↪ INTEL
SLURM_JOB_NODELIST=nid03508
HOME=/users/*REDACTED*
SHLVL=4
JDK_HOME=/usr/lib64/jvm/java
QT_SYSTEM_DIR=/usr/share/desktop-data
CRAY_LIBSCI_VERSION=18.07.1
PE_HDF5_PARALLEL_DEFAULT_VOLATILE_PRGENV=GNU
PE_MPICH_TARGET_VAR_nvidia35=-lcudart
PE_NETCDF_HDF5PARALLEL_DEFAULT_VOLATILE_PRGENV=GNU
PE_PKGCONFIG_PRODUCTS_DEFAULT=PE_PAPI
PE_TPSL_64_DEFAULT_GENCOMPS_GNU_haswell=71 53 49
SLURM_PTY_PORT=39693
OSTYPE=linux
LESS_ADVANCED_PREPROCESSOR=no
PE_TPSL_DEFAULT_GENCOMPILERS_INTEL_interlagos=16.0
SLURM_LOCALID=0
LINKER_X86_64=/opt/cray/pe/cce/8.7.3/binutils/x86_64⌋
↪ /x86_64-pc-linux-gnu/bin/ld
PE_LIBSCI_ACC_DEFAULT_NV_SUFFIX_nvidia60=nv60

```
PE_MPICH_DEFAULT_VOLATILE_PKGCONFIG_PATH=/opt/cray/p↵
↪  e/mpt/7.7.2/gni/mpich-@PRGENV@@PE_MPICH_DEFAULT_↵
↪  DIR_DEFAULT64@/@PE_MPICH_DEFAULT_GENCOMPS@/lib/p↵
↪  kgconfig
PE_PETSC_DEFAULT_GENCOMPILERS_CRAY_interlagos=8.6
PE_TPSL_DEFAULT_VOLATILE_PRGENV=CRAY CRAY64 GNU GNU64
↪  INTEL INTEL64
XCURSOR_THEME=DMZ
LS_OPTIONS=-N --color=none -T 0
CRAY_PMI_INCLUDE_OPTS=-I/opt/cray/pe/pmi/5.0.14/incl↵
↪  ude
PE_TPSL_64_DEFAULT_GENCOMPS_CRAY_interlagos=86
PE_TPSL_DEFAULT_GENCOMPS_INTEL_sandybridge=160
PROCESSOR_COUNT=26
SLURM_CLUSTER_NAME=daint
SLURM_JOB_CPUS_PER_NODE=24
SLURM_JOB_GID=31496
WINDOWMANAGER=
PRGENVMODULES=PrgEnv-cray:PrgEnv-gnu:PrgEnv-intel:Pr↵
↪  gEnv-pathscale:PrgEnv-pgi
CRAYPE_NETWORK_TARGET=aries
ATP_MRNET_COMM_PATH=/opt/cray/pe/atp/2.1.2/libexec/a↵
↪  tp_mrnet_commnode_wrapper
CRAYLMD_LICENSE_FILE=/opt/cray/pe/cce/cce.lic
PE_TPSL_DEFAULT_GENCOMPILERS_CRAY_haswell=8.6
PKG_CONFIG_PATH_DEFAULT=/opt/cray/pe/papi/5.6.0.2/li↵
↪  b64/pkgconfig
PE_MPICH_DIR_CRAY_DEFAULT64=64
SLURM_SUBMIT_HOST=daint103
SLURM_GTIDS=0
PE_LEVEL=8.7
PE_PETSC_DEFAULT_GENCOMPILERS_GNU_haswell=7.1 5.3 4.9
PE_TPSL_64_DEFAULT_GENCOMPILERS_GNU_mic_knl=7.1 5.3
PE_TPSL_DEFAULT_GENCOMPILERS_GNU_interlagos=7.1 5.3
↪  4.9
PE_TPSL_DEFAULT_GENCOMPILERS_INTEL_sandybridge=16.0
SLURM_JOB_PARTITION=debug
LOGNAME=*REDACTED*
MACHTYPE=x86_64-suse-linux
LESS=-M -I -R
G_FILENAME_ENCODING=@locale,UTF-8,ISO-8859-15,CP1252
CRAYLIBS_AARCH64=/opt/cray/pe/cce/8.7.3/cce/aarch64/↵
↪  lib
CRAYLIBS_X86_64=/opt/cray/pe/cce/8.7.3/cce/x86_64/lib
CRAY_GNI_HEADERS_INCLUDE_OPTS=-I/opt/cray/gni-header↵
↪  s/5.0.12.0-6.0.7.0_24.1__g3b1768f.ari/include
CRAY_LIBSCI_PREFIX_DIR=/opt/cray/pe/libsci/18.07.1/C↵
↪  RAY/8.6/x86_64
PE_HDF5_DEFAULT_GENCOMPS_GNU=
PE_MPICH_NV_LIBS=
PE_NETCDF_DEFAULT_REQUIRED_PRODUCTS=PE_HDF5
PE_TPSL_64_DEFAULT_GENCOMPILERS_GNU_haswell=7.1 5.3
↪  4.9
PE_TPSL_64_DEFAULT_GENCOMPILERS_INTEL_sandybridge=16↵
↪  .0
PE_TPSL_DEFAULT_GENCOMPS_GNU_x86_64=71 53 49
PE_TRILINOS_DEFAULT_REQUIRED_PRODUCTS=PE_MPICH:PE_HD↵
↪  F5_PARALLEL:PE_NETCDF_HDF5PARALLEL:PE_LIBSCI:PE_↵
↪  TPSL
PYTHONPATH=/apps/daint/UES/xalt/0.7.6/site:/apps/dai↵
↪  nt/UES/xalt/0.7.6/libexec
CVS_RSH=ssh
DMAPP_ABORT_ON_ERROR=1
PE_LIBSCI_OMP_REQUIRES=
PE_MPICH_DEFAULT_GENCOMPILERS_CRAY=8.6
PE_TRILINOS_DEFAULT_GENCOMPS_GNU_x86_64=71 53 49
PE_MPICH_GENCOMPS_CRAY=86
SLURM_STEP_NUM_TASKS=1
SSH_CONNECTION=148.187.1.6 44730 148.187.26.66 22
XDG_DATA_DIRS=/usr/share
TOOLMODULES=apprentice:apprentice2:atp:chapel:cray-l↵
↪  gdb:craypat:craypkg-gen:cray-snplauncher:ddt:gdb↵
↪  :iobuf:papi:perftools:perftools-lite:stat:totalv↵
↪  iew:xt-craypat:xt-lgdb:xt-papi:xt-totalview
DVS_INCLUDE_OPTS=-I/opt/cray/dvs/2.7_2.2.113-6.0.7.1↵
↪  _7.6__g1bbc03e/include
PE_LIBSCI_ACC_DEFAULT_GENCOMPILERS_CRAY_x86_64=8.5
PE_LIBSCI_ACC_DEFAULT_NV_SUFFIX_nvidia35=nv35
PE_LIBSCI_DEFAULT_REQUIRED_PRODUCTS=PE_MPICH
PE_MPICH_DEFAULT_FIXED_PRGENV=INTEL
PE_MPICH_DEFAULT_GENCOMPS_GNU=71 51 49
PE_TPSL_64_DEFAULT_GENCOMPILERS_INTEL_interlagos=16.0
PE_TPSL_DEFAULT_GENCOMPILERS_CRAY_sandybridge=8.6
SLURM_JOB_ACCOUNT=d69
GPU_DEVICE_ORDINAL=0
MODULESHOME=/opt/cray/pe/modules/3.2.10.6
PE_GA_DEFAULT_FIXED_PRGENV=CRAY PGI INTEL
PE_LIBSCI_DEFAULT_VOLATILE_PKGCONFIG_PATH=/opt/cray/↵
↪  pe/libsci/18.07.1/@PRGENV@/@PE_LIBSCI_DEFAULT_GE↵
↪  NCOMPS@/@PE_LIBSCI_DEFAULT_TARGET@/lib/pkgconfig
PE_TPSL_DEFAULT_GENCOMPILERS_GNU_sandybridge=7.1 5.3
↪  4.9
SLURM_JOB_NUM_NODES=1
LESSOPEN=lessopen.sh %s
PKG_CONFIG_PATH=/opt/slurm/17.11.12.cscs/lib64/pkgco↵
↪  nfig:/opt/cray/rca/2.2.18-6.0.7.0_33.3__g2aa4f39↵
↪  .ari/lib64/pkgconfig:/opt/cray/alps/6.6.43-6.0.7↵
↪  .0_26.4__ga796da3.ari/lib64/pkgconfig:/opt/cray/↵
↪  xpmem/2.2.15-6.0.7.1_5.10__g7549d06.ari/lib64/pk↵
↪  gconfig:/opt/cray/gni-headers/5.0.12.0-6.0.7.0_2↵
↪  4.1__g3b1768f.ari/lib64/pkgconfig:/opt/cray/dmap↵
↪  p/7.1.1-6.0.7.0_34.3__g5a674e0.ari/lib64/pkgconf↵
↪  ig:/opt/cray/pe/pmi/5.0.14/lib64/pkgconfig:/opt/↵
↪  cray/ugni/6.0.14.0-6.0.7.0_23.1__gea11d3d.ari/li↵
↪  b64/pkgconfig:/opt/cray/udreg/2.3.2-6.0.7.0_33.1↵
↪  8__g5196236.ari/lib64/pkgconfig:/opt/cray/pe/cra↵
↪  ype/2.5.15/pkg-config:/opt/cray/pe/iobuf/2.0.8/l↵
↪  ib/pkgconfig:/opt/slurm/default/lib64/pkgconfig:↵
↪  /opt/cray/pe/atp/2.1.2/lib/pkgconfig
SLURM_TIME_FORMAT=relative
```

```
CRAY_CXX_IPA_LIBS_X86_64=/opt/cray/pe/cce/8.7.3/cce/⌋
↪    x86_64/lib/libcray-c++-rts.a
PE_MPICH_NV_LIBS_nvidia35=-lcudart
PE_PETSC_DEFAULT_VOLATILE_PKGCONFIG_PATH=/opt/cray/p⌋
↪    e/petsc/3.8.4.0/complex/@PRGENV@/@PE_PETSC_DEFAU⌋
↪    LT_GENCOMPS@/@PE_PETSC_DEFAULT_TARGET@/lib/pkgco⌋
↪    nfig
PELOCAL_PRGENV=true
CRAYPAT_OPTS_EXECUTABLE=sbin/pat-opts
CRAY_BINUTILS_BIN_X86_64=/opt/cray/pe/cce/8.7.3/binu⌋
↪    tils/x86_64/bin
INCLUDE_PATH_AARCH64=/opt/cray/pe/cce/8.7.3/cce/aarc⌋
↪    h64/include/craylibs
LIBSCI_BASE_DIR=/opt/cray/pe/libsci/18.07.1
PE_TPSL_64_DEFAULT_GENCOMPS_INTEL_x86_64=160
SLURM_STEP_TASKS_PER_NODE=1
CRAY_NUM_COOKIES=2
LIBSCI_VERSION=18.07.1
PE_LIBSCI_DEFAULT_PKGCONFIG_VARIABLES=PE_LIBSCI_DEFA⌋
↪    ULT_OMP_REQUIRES_@openmp@:PE_SCI_EXT_LIBPATH:PE_⌋
↪    SCI_EXT_LIBNAME
PE_MPICH_NV_LIBS_nvidia60=-lcudart
PE_TPSL_64_DEFAULT_GENCOMPS_GNU_sandybridge=71 53 49
PE_TPSL_DEFAULT_GENCOMPS_INTEL_mic_knl=160
ACLOCAL_PATH=/users/*REDACTED*/.local/share/aclocal:⌋
↪    /users/*REDACTED*/.local/share/aclocal:
SLURM_STEP_NODELIST=nid03508
CRAY_COOKIES=2850160640,2850226176
XDG_RUNTIME_DIR=/run/user/23992
CRAY_PRE_COMPILE_OPTS=-hnetwork=aries
CRAY_ALPS_INCLUDE_OPTS=-I/opt/cray/alps/6.6.43-6.0.7⌋
↪    .0_26.4__ga796da3.ari/include
PE_FFTW_DEFAULT_TARGET_broadwell=broadwell
PE_LIBSCI_GENCOMPILERS_INTEL_x86_64=16.0
PE_PGI_DEFAULT_FIXED_PKGCONFIG_PATH=/opt/cray/pe/par⌋
↪    allel-netcdf/1.8.1.3/PGI/15.3/lib/pkgconfig:/opt⌋
↪    /cray/pe/netcdf-hdf5parallel/4.6.1.2/PGI/17.10/l⌋
↪    ib/pkgconfig:/opt/cray/pe/netcdf/4.6.1.2/PGI/17.⌋
↪    10/lib/pkgconfig:/opt/cray/pe/hdf5-parallel/1.10⌋
↪    .2.0/PGI/17.10/lib/pkgconfig:/opt/cray/pe/hdf5/1⌋
↪    .10.2.0/PGI/17.10/lib/pkgconfig:/opt/cray/pe/ga/⌋
↪    5.3.0.8/PGI/17.10/lib/pkgconfig
PE_TPSL_64_DEFAULT_GENCOMPILERS_GNU_x86_64=7.1 5.3⌋
↪    4.9
CRAY_CPU_TARGET=haswell
CRAY_UGNI_INCLUDE_OPTS=-I/opt/cray/ugni/6.0.14.0-6.0⌋
↪    .7.0_23.1__gea11d3d.ari/include
CRAY_XPMEM_INCLUDE_OPTS=-I/opt/cray/xpmem/2.2.15-6.0⌋
↪    .7.1_5.10__g7549d06.ari/include
PE_LIBSCI_REQUIRED_PRODUCTS=PE_MPICH
PE_MPICH_DEFAULT_GENCOMPILERS_PGI=15.3
PE_PAPI_DEFAULT_ACCELL_FAMILY_LIBS=
PE_TPSL_64_DEFAULT_GENCOMPS_CRAY_x86_64=86
craype_already_loaded=0
PE_MPICH_GENCOMPS_PGI=153
CUDA_CACHE_PATH=/scratch/snx3000/*REDACTED*/.nv/Comp⌋
↪    uteCache
PE_LIBSCI_DEFAULT_GENCOMPILERS_GNU_x86_64=7.1 6.1 5.1⌋
↪    4.9
PE_LIBSCI_GENCOMPS_GNU_x86_64=71 61 51 49
PE_TPSL_DEFAULT_GENCOMPS_INTEL_haswell=160
SLURM_CPU_BIND=quiet,mask_cpu:0xFFFFFF
LESSCLOSE=lessclose.sh %s %s
ATP_HOME=/opt/cray/pe/atp/2.1.2
PE_FFTW_DEFAULT_TARGET_x86_64=x86_64
PE_PETSC_DEFAULT_GENCOMPILERS_INTEL_x86_64=16.0
SCRATCH=/scratch/snx3000/*REDACTED*
G_BROKEN_FILENAMES=1
CC_X86_64=/opt/cray/pe/cce/8.7.3/cce/x86_64
CRAY_LD_LIBRARY_PATH=/opt/cray/pe/mpt/7.7.2/gni/mpic⌋
↪    h-cray/8.6/lib:/opt/cray/pe/perftools/7.0.2/lib6⌋
↪    4:/opt/cray/rca/2.2.18-6.0.7.0_33.3__g2aa4f39.ar⌋
↪    i/lib64:/opt/cray/alps/6.6.43-6.0.7.0_26.4__ga79⌋
↪    6da3.ari/lib64:/opt/cray/xpmem/2.2.15-6.0.7.1_5.⌋
↪    10__g7549d06.ari/lib64:/opt/cray/dmapp/7.1.1-6.0⌋
↪    .7.0_34.3__g5a674e0.ari/lib64:/opt/cray/pe/pmi/5⌋
↪    .0.14/lib64:/opt/cray/ugni/6.0.14.0-6.0.7.0_23.1⌋
↪    __gea11d3d.ari/lib64:/opt/cray/udreg/2.3.2-6.0.7⌋
↪    .0_33.18__g5196236.ari/lib64:/opt/cray/pe/libsci⌋
↪    /18.07.1/CRAY/8.6/x86_64/lib:/opt/cray/pe/cce/8.⌋
↪    7.3/cce/x86_64/lib
PE_FFTW_DEFAULT_TARGET_haswell=haswell
PE_GA_DEFAULT_GENCOMPS_GNU=53 49
PE_GA_DEFAULT_VOLATILE_PKGCONFIG_PATH=/opt/cray/pe/g⌋
↪    a/5.3.0.8/@PRGENV@/@PE_GA_DEFAULT_GENCOMPS@/lib/⌋
↪    pkgconfig
PE_INTEL_DEFAULT_FIXED_PKGCONFIG_PATH=/opt/cray/pe/p⌋
↪    arallel-netcdf/1.8.1.3/INTEL/16.0/lib/pkgconfig:⌋
↪    /opt/cray/pe/netcdf-hdf5parallel/4.6.1.2/INTEL/1⌋
↪    6.0/lib/pkgconfig:/opt/cray/pe/netcdf/4.6.1.2/IN⌋
↪    TEL/16.0/lib/pkgconfig:/opt/cray/pe/mpt/7.7.2/gn⌋
↪    i/mpich-intel/16.0/lib/pkgconfig:/opt/cray/pe/hd⌋
↪    f5-parallel/1.10.2.0/INTEL/16.0/lib/pkgconfig:/o⌋
↪    pt/cray/pe/hdf5/1.10.2.0/INTEL/16.0/lib/pkgconfi⌋
↪    g:/opt/cray/pe/ga/5.3.0.8/INTEL/18.0/lib/pkgconf⌋
↪    ig
PE_PAPI_DEFAULT_ACCEL_LIBS=
PE_PETSC_DEFAULT_GENCOMPILERS_GNU_interlagos=7.1 5.3⌋
↪    4.9
PE_PETSC_DEFAULT_GENCOMPILERS_INTEL_haswell=16.0
PE_SMA_DEFAULT_DIR_PGI_DEFAULT64=64
PE_TPSL_64_DEFAULT_GENCOMPILERS_INTEL_x86_skylake=16⌋
↪    .0
COLORTERM=1
JAVA_ROOT=/usr/lib64/jvm/java
PE_MPICH_DEFAULT_DIR_CRAY_DEFAULT64=64
PE_PETSC_DEFAULT_GENCOMPS_CRAY_haswell=86
PE_PETSC_DEFAULT_GENCOMPS_GNU_x86_64=71 53 49
PE_PETSC_DEFAULT_GENCOMPS_INTEL_x86_64=160
```

```
BASH_FUNC_module%%=() {  eval
↪ `/opt/cray/pe/modules/3.2.10.6/bin/modulecmd
↪ bash $*`
}
_=/usr/bin/env
+ inxi -F -c0
./collect_environment.sh: line 8: inxi: command not
↪ found
+ lsblk -a
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
loop0   7:0    0 18.5M  0 loop
↪ /var/opt/cray/imps-distribution/squash/mounts/p0
loop1   7:1    0  128K  0 loop /var/opt/cray/imps-di⌋
↪ stribution/squash/mounts/global
loop2   7:2    0  2.5G  1 loop /.rootfs_lower_ro
loop3   7:3    0  1.6G  1 loop
↪ /var/opt/cray/imps-image-binding/diags/squash_mo⌋
↪ unts/squashfs_1z3Y3h_mount_point
loop4   7:4    0        1 loop
loop5   7:5    0        1 loop
loop6   7:6    0        1 loop
loop7   7:7    0        1 loop
loop8   7:8    0        1 loop
loop9   7:9    0        1 loop
loop10  7:10   0        1 loop
loop11  7:11   0        1 loop
loop12  7:12   0        1 loop
loop13  7:13   0        1 loop
loop14  7:14   0        1 loop
loop15  7:15   0        1 loop
loop16  7:16   0        1 loop
loop17  7:17   0        1 loop
loop18  7:18   0        1 loop
loop19  7:19   0        1 loop
loop20  7:20   0        1 loop
loop21  7:21   0        1 loop
loop22  7:22   0        1 loop
loop23  7:23   0        1 loop
loop24  7:24   0        1 loop
loop25  7:25   0        1 loop
loop26  7:26   0        1 loop
loop27  7:27   0        1 loop
+ lsscsi -s
+ module list
++ /opt/cray/pe/modules/3.2.10.6/bin/modulecmd bash
↪ list
Currently Loaded Modulefiles:
  1) modules/3.2.10.6
  2) cce/8.7.3
  3) craype-network-aries
  4) craype/2.5.15
  5) cray-libsci/18.07.1
  6) udreg/2.3.2-6.0.7.0_33.18__g5196236.ari
  7) ugni/6.0.14.0-6.0.7.0_23.1__gea11d3d.ari
  8) pmi/5.0.14
  9) dmapp/7.1.1-6.0.7.0_34.3__g5a674e0.ari
 10) gni-headers/5.0.12.0-6.0.7.0_24.1__g3b1768f.ari
 11) xpmem/2.2.15-6.0.7.1_5.10__g7549d06.ari
 12) job/2.2.3-6.0.7.0_44.1__g6c4e934.ari
 13) dvs/2.7_2.2.113-6.0.7.1_7.6__g1bbc03e
 14) alps/6.6.43-6.0.7.0_26.4__ga796da3.ari
 15) rca/2.2.18-6.0.7.0_33.3__g2aa4f39.ari
 16) atp/2.1.2
 17) perftools-base/7.0.2
 18) PrgEnv-cray/6.0.4
 19) cray-mpich/7.7.2
 20) slurm/17.11.12.cscs-1
 21) craype-haswell
 22) xalt/daint-2016.11
 23) Base-opts/2.4.135-6.0.7.0_38.1__g718f891.ari
+ eval
+ nvidia-smi
Fri Apr  5 02:01:25 2019
+--------------------------------------------------⌋
↪ ------------------------+
| NVIDIA-SMI 396.44              Driver Version:
↪ 396.44                  |
|------------------------------+------------------⌋
↪ ---+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A
↪ | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|        Memory-Usage
↪ | GPU-Util  Compute M. |
|==============================+==================⌋
↪ ===+======================|
|   0  Tesla P100-PCIE...  On   | 00000000:02:00.0 Off
↪ |                    0 |
| N/A   30C    P0    28W / 250W |      0MiB / 16280MiB
↪ |      0%   E. Process |
+------------------------------+------------------⌋
↪ ---+----------------------+

+--------------------------------------------------⌋
↪ ------------------------+
| Processes:
↪           GPU Memory |
|  GPU       PID   Type   Process name
↪             Usage      |
|==================================================⌋
↪ ========================|
|  No running processes found
↪                      |
+--------------------------------------------------⌋
↪ ------------------------+
+ cat
+ lshw -short -quiet -sanitize
./collect_environment.sh: line 13: lshw: command not
↪ found
+ lspci
./collect_environment.sh: line 13: lspci: command not
↪ found
```

```
Output from node-level machine 1: geev:
LC_PAPER=de_DE.UTF-8
XDG_SESSION_ID=6021
LC_ADDRESS=de_DE.UTF-8
HOSTNAME=geev
LC_MONETARY=de_DE.UTF-8
TERM=screen-256color
SHELL=/bin/zsh
HISTSIZE=5000
SSH_CLIENT=10.3.3.8 33552 22
LC_NUMERIC=de_DE.UTF-8
QTDIR=/usr/lib64/qt-3.3
QTINC=/usr/lib64/qt-3.3/include
SSH_TTY=/dev/pts/0
ZSH=/home/USER/oh-my-zsh
LC_ALL=en_US.UTF-8
TMP_PROMPT=[%{$fg[$NCOLOR]%}%B%n%b%{$reset_color%}:%⌋
↪  {$fg[red]%}%30<...<%~%<<%{$reset_color%}]%(!.#.$)
GIT_EDITOR=vim
QT_GRAPHICSSYSTEM_CHECKED=1
HISTFILESIZE=10000
USER=USER
LS_COLORS=no=00:fi=00:di=01;34:ln=00;36:pi=40;33:so=⌋
↪  01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=41;33;⌋
↪  01:ex=00;32:*.cmd=00;32:*.exe=01;32:*.com=01;32:⌋
↪  *.bat=01;32:*.btm=01;32:*.dll=01;32:*.tar=00;31:⌋
↪  *.tbz=00;31:*.tgz=00;31:*.rpm=00;31:*.deb=00;31:⌋
↪  *.arj=00;31:*.taz=00;31:*.lzh=00;31:*.lzma=00;31⌋
↪  :*.zip=00;31:*.zoo=00;31:*.z=00;31:*.Z=00;31:*.g⌋
↪  z=00;31:*.bz2=00;31:*.tb2=00;31:*.tz2=00;31:*.tb⌋
↪  z2=00;31:*.avi=01;35:*.bmp=01;35:*.fli=01;35:*.g⌋
↪  if=01;35:*.jpg=01;35:*.jpeg=01;35:*.mng=01;35:*.⌋
↪  mov=01;35:*.mpg=01;35:*.pcx=01;35:*.pbm=01;35:*.⌋
↪  pgm=01;35:*.png=01;35:*.ppm=01;35:*.tga=01;35:*.⌋
↪  tif=01;35:*.xbm=01;35:*.xpm=01;35:*.dl=01;35:*.g⌋
↪  l=01;35:*.wmv=01;35:*.aiff=00;32:*.au=00;32:*.mi⌋
↪  d=00;32:*.mp3=00;32:*.ogg=00;32:*.voc=00;32:*.wa⌋
↪  v=00;32:
LC_TELEPHONE=de_DE.UTF-8
FZF_DEFAULT_OPTS=--reverse --border
PAGER=less
LSCOLORS=Gxfxcxdxbxegedabagacad
MAIL=/var/spool/mail/USER
PATH=/usr/lib64/qt-3.3/bin:/usr/local/bin:/usr/bin:/⌋
↪  home/USER/bin:/usr/local/sbin:/usr/sbin:/home/US⌋
↪  ER/.fzf/bin
FZF_COMPLETION_TRIGGER=]]
_=/usr/bin/env
LC_IDENTIFICATION=de_DE.UTF-8
PWD=/home/USER
EDITOR=vim
LANG=en_US.UTF-8
MODULEPATH=/usr/share/Modules/modulefiles:/etc/modul⌋
↪  efiles:/opt/boost/modules:/opt/modules:/opt/eb/m⌋
↪  odules/all:/opt/hpx/modules
```

```
KEYTIMEOUT=1
LOADEDMODULES=
LC_MEASUREMENT=de_DE.UTF-8
HISTCONTROL=ignoredups
SHLVL=2
HOME=/home/USER
LESS=-R
LOGNAME=USER
QTLIB=/usr/lib64/qt-3.3/lib
SSH_CONNECTION=10.3.3.8 33552 10.3.3.22 22
LC_CTYPE=en_US.UTF-8
MODULESHOME=/usr/share/Modules
LESSOPEN=||/usr/bin/lesspipe.sh %s
XDG_RUNTIME_DIR=/run/user/3095
LC_TIME=de_DE.UTF-8
LC_NAME=de_DE.UTF-8
LSB Version:      :core-4.1-amd64:core-4.1-noarch:⌋
↪  cxx-4.1-amd64:cxx-4.1-noarch:desktop-4.1-amd64:d⌋
↪  esktop-4.1-noarch:languages-4.1-amd64:languages-⌋
↪  4.1-noarch:printing-4.1-amd64:printing-4.1-noarch
Distributor ID:    CentOS
Description:       CentOS Linux release 7.6.1810
↪  (Core)
Release:       7.6.1810
Codename:      Core
Linux geev 3.10.0-957.10.1.el7.x86_64 #1 SMP Mon Mar
↪  18 15:06:45 UTC 2019 x86_64 x86_64 x86_64
↪  GNU/Linux
Architecture:      x86_64
CPU op-mode(s):    32-bit, 64-bit
Byte Order:        Little Endian
CPU(s):            20
On-line CPU(s) list: 0-19
Thread(s) per core: 1
Core(s) per socket: 10
Socket(s):         2
NUMA node(s):      2
Vendor ID:         GenuineIntel
CPU family:        6
Model:             63
Model name:        Intel(R) Xeon(R) CPU E5-2660 v3
↪  @ 2.60GHz
Stepping:          2
CPU MHz:           1200.183
CPU max MHz:       3300.0000
CPU min MHz:       1200.0000
BogoMIPS:          5199.81
Virtualization:    VT-x
L1d cache:         32K
L1i cache:         32K
L2 cache:          256K
L3 cache:          25600K
NUMA node0 CPU(s): 0,2,4,6,8,10,12,14,16,18
NUMA node1 CPU(s): 1,3,5,7,9,11,13,15,17,19
```

```
Flags:                    fpu vme de pse tsc msr pae mce
↪  cx8 apic sep mtrr pge mca cmov pat pse36 clflush
↪  dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx
↪  pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs
↪  bts rep_good nopl xtopology nonstop_tsc
↪  aperfmperf eagerfpu pni pclmulqdq dtes64 monitor
↪  ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr
↪  pdcm pcid dca sse4_1 sse4_2 x2apic movbe popcnt
↪  tsc_deadline_timer aes xsave avx f16c rdrand
↪  lahf_lm abm epb ssbd ibrs ibpb stibp tpr_shadow
↪  vnmi flexpriority ept vpid fsgsbase tsc_adjust
↪  bmi1 avx2 smep bmi2 erms invpcid cqm xsaveopt
↪  cqm_llc cqm_occup_llc dtherm ida arat pln pts
↪  spec_ctrl intel_stibp flush_l1d
MemTotal:       264033324 kB
MemFree:        181286056 kB
MemAvailable:   259627632 kB
Buffers:             2120 kB
Cached:          76094992 kB
SwapCached:             0 kB
Active:          41680532 kB
Inactive:        34474532 kB
Active(anon):       66412 kB
Inactive(anon):     26476 kB
Active(file):    41614120 kB
Inactive(file):  34448056 kB
Unevictable:            0 kB
Mlocked:                0 kB
SwapTotal:        4194300 kB
SwapFree:         4194300 kB
Dirty:                  0 kB
Writeback:              0 kB
AnonPages:          57688 kB
Mapped:             78544 kB
Shmem:              34936 kB
Slab:             3427184 kB
SReclaimable:     3163536 kB
SUnreclaim:        263648 kB
KernelStack:         8208 kB
PageTables:          8968 kB
NFS_Unstable:           0 kB
Bounce:                 0 kB
WritebackTmp:           0 kB
CommitLimit:    136210960 kB
Committed_AS:      298832 kB
VmallocTotal:   34359738367 kB
VmallocUsed:       780616 kB
VmallocChunk:   34224715772 kB
HardwareCorrupted:      0 kB
AnonHugePages:       4096 kB
CmaTotal:               0 kB
CmaFree:                0 kB
HugePages_Total:        0
HugePages_Free:         0
HugePages_Rsvd:         0
HugePages_Surp:         0
Hugepagesize:        2048 kB

DirectMap4k:      3781364 kB
DirectMap2M:    193255424 kB
DirectMap1G:     73400320 kB
NAME            MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda               8:0    0 222.5G  0 disk
├─sda1            8:1    0   512M  0 part /boot
└─sda2            8:2    0   222G  0 part
  ├─rostam-root 253:0    0 196.2G  0 lvm  /
  └─rostam-swap 253:1    0     4G  0 lvm  [SWAP]
sr0              11:0    1  1024M  0 rom
[0:2:0:0]    disk    DELL    PERC H730 Mini  4.29
↪  /dev/sda    238GB
[10:0:0:0]   cd/dvd  HL-DT-ST DVD+-RW GTA0N    A3B0
↪  /dev/sr0       -
Wed Apr 10 11:56:18 2019
+-------------------------------------------------┐
↪  ------------------------+
| NVIDIA-SMI 418.40.04    Driver Version: 418.40.04
↪  CUDA Version: 10.1      |
|------------------------------+------------------┐
↪  ---+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A
↪  | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|        Memory-Usage
↪  | GPU-Util  Compute M. |
|==============================+==================┐
↪  ===+======================|
|   0  Tesla V100-PCIE...  Off  | 00000000:03:00.0 Off
↪  |                    0 |
| N/A   30C    P0    34W / 250W |      0MiB / 32480MiB
↪  |      0%      Default |
+------------------------------+------------------┐
↪  ---+----------------------+
|   1  Tesla V100-PCIE...  Off  | 00000000:82:00.0 Off
↪  |                    0 |
| N/A   31C    P0    35W / 250W |      0MiB / 32480MiB
↪  |      0%      Default |
+------------------------------+------------------┐
↪  ---+----------------------+

+-------------------------------------------------┐
↪  ------------------------+
| Processes:
↪           GPU Memory |
| GPU       PID   Type   Process name
↪              Usage       |
|=================================================┐
↪  =========================|
|  No running processes found
↪                     |
+-------------------------------------------------┐
↪  ------------------------+
00:00.0 Host bridge: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 DMI2 (rev 02)
```

00:01.0 PCI bridge: Intel Corporation Xeon E7 v3/Xeon
↪ E5 v3/Core i7 PCI Express Root Port 1 (rev 02)
00:02.0 PCI bridge: Intel Corporation Xeon E7 v3/Xeon
↪ E5 v3/Core i7 PCI Express Root Port 2 (rev 02)
00:03.0 PCI bridge: Intel Corporation Xeon E7 v3/Xeon
↪ E5 v3/Core i7 PCI Express Root Port 3 (rev 02)
00:03.2 PCI bridge: Intel Corporation Xeon E7 v3/Xeon
↪ E5 v3/Core i7 PCI Express Root Port 3 (rev 02)
00:05.0 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Address Map, VTd_Misc,
↪ System Management (rev 02)
00:05.1 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Hot Plug (rev 02)
00:05.2 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 RAS, Control Status and
↪ Global Errors (rev 02)
00:05.4 PIC: Intel Corporation Xeon E7 v3/Xeon E5
↪ v3/Core i7 I/O APIC (rev 02)
00:11.0 Unassigned class [ff00]: Intel Corporation
↪ C610/X99 series chipset SPSR (rev 05)
00:11.4 SATA controller: Intel Corporation C610/X99
↪ series chipset sSATA Controller [AHCI mode] (rev
↪ 05)
00:16.0 Communication controller: Intel Corporation
↪ C610/X99 series chipset MEI Controller #1 (rev 05)
00:16.1 Communication controller: Intel Corporation
↪ C610/X99 series chipset MEI Controller #2 (rev 05)
00:1a.0 USB controller: Intel Corporation C610/X99
↪ series chipset USB Enhanced Host Controller #2
↪ (rev 05)
00:1c.0 PCI bridge: Intel Corporation C610/X99 series
↪ chipset PCI Express Root Port #1 (rev d5)
00:1c.7 PCI bridge: Intel Corporation C610/X99 series
↪ chipset PCI Express Root Port #8 (rev d5)
00:1d.0 USB controller: Intel Corporation C610/X99
↪ series chipset USB Enhanced Host Controller #1
↪ (rev 05)
00:1f.0 ISA bridge: Intel Corporation C610/X99 series
↪ chipset LPC Controller (rev 05)
00:1f.2 SATA controller: Intel Corporation C610/X99
↪ series chipset 6-Port SATA Controller [AHCI mode]
↪ (rev 05)
01:00.0 Ethernet controller: Broadcom Inc. and
↪ subsidiaries NetXtreme II BCM57800 1/10 Gigabit
↪ Ethernet (rev 10)
01:00.1 Ethernet controller: Broadcom Inc. and
↪ subsidiaries NetXtreme II BCM57800 1/10 Gigabit
↪ Ethernet (rev 10)
01:00.2 Ethernet controller: Broadcom Inc. and
↪ subsidiaries NetXtreme II BCM57800 1/10 Gigabit
↪ Ethernet (rev 10)
01:00.3 Ethernet controller: Broadcom Inc. and
↪ subsidiaries NetXtreme II BCM57800 1/10 Gigabit
↪ Ethernet (rev 10)

02:00.0 RAID bus controller: LSI Logic / Symbios Logic
↪ MegaRAID SAS-3 3108 [Invader] (rev 02)
03:00.0 3D controller: NVIDIA Corporation GV100GL
↪ [Tesla V100 PCIe 32GB] (rev a1)
06:00.0 PCI bridge: Renesas Technology Corp. SH7758
↪ PCIe Switch [PS]
07:00.0 PCI bridge: Renesas Technology Corp. SH7758
↪ PCIe Switch [PS]
08:00.0 PCI bridge: Renesas Technology Corp. SH7758
↪ PCIe-PCI Bridge [PPB]
09:00.0 VGA compatible controller: Matrox Electronics
↪ Systems Ltd. G200eR2 (rev 01)
7f:08.0 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 QPI Link 0 (rev 02)
7f:08.2 Performance counters: Intel Corporation Xeon
↪ E7 v3/Xeon E5 v3/Core i7 QPI Link 0 (rev 02)
7f:08.3 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 QPI Link 0 (rev 02)
7f:09.0 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 QPI Link 1 (rev 02)
7f:09.2 Performance counters: Intel Corporation Xeon
↪ E7 v3/Xeon E5 v3/Core i7 QPI Link 1 (rev 02)
7f:09.3 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 QPI Link 1 (rev 02)
7f:0b.0 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 R3 QPI Link 0 & 1 Monitoring
↪ (rev 02)
7f:0b.1 Performance counters: Intel Corporation Xeon
↪ E7 v3/Xeon E5 v3/Core i7 R3 QPI Link 0 & 1
↪ Monitoring (rev 02)
7f:0b.2 Performance counters: Intel Corporation Xeon
↪ E7 v3/Xeon E5 v3/Core i7 R3 QPI Link 0 & 1
↪ Monitoring (rev 02)
7f:0c.0 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Unicast Registers (rev 02)
7f:0c.1 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Unicast Registers (rev 02)
7f:0c.2 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Unicast Registers (rev 02)
7f:0c.3 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Unicast Registers (rev 02)
7f:0c.4 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Unicast Registers (rev 02)
7f:0c.5 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Unicast Registers (rev 02)
7f:0c.6 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Unicast Registers (rev 02)
7f:0c.7 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Unicast Registers (rev 02)
7f:0d.0 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Unicast Registers (rev 02)
7f:0d.1 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Unicast Registers (rev 02)

```
7f:0f.0 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Buffered Ring Agent (rev 02)
7f:0f.1 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Buffered Ring Agent (rev 02)
7f:0f.2 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Buffered Ring Agent (rev 02)
7f:0f.3 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Buffered Ring Agent (rev 02)
7f:0f.4 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 System Address Decoder &
↪  Broadcast Registers (rev 02)
7f:0f.5 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 System Address Decoder &
↪  Broadcast Registers (rev 02)
7f:0f.6 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 System Address Decoder &
↪  Broadcast Registers (rev 02)
7f:10.0 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 PCIe Ring Interface (rev 02)
7f:10.1 Performance counters: Intel Corporation Xeon
↪  E7 v3/Xeon E5 v3/Core i7 PCIe Ring Interface (rev
↪  02)
7f:10.5 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Scratchpad & Semaphore
↪  Registers (rev 02)
7f:10.6 Performance counters: Intel Corporation Xeon
↪  E7 v3/Xeon E5 v3/Core i7 Scratchpad & Semaphore
↪  Registers (rev 02)
7f:10.7 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Scratchpad & Semaphore
↪  Registers (rev 02)
7f:12.0 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Home Agent 0 (rev 02)
7f:12.1 Performance counters: Intel Corporation Xeon
↪  E7 v3/Xeon E5 v3/Core i7 Home Agent 0 (rev 02)
7f:12.2 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Home Agent 0 Debug (rev 02)
7f:12.4 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Home Agent 1 (rev 02)
7f:12.5 Performance counters: Intel Corporation Xeon
↪  E7 v3/Xeon E5 v3/Core i7 Home Agent 1 (rev 02)
7f:12.6 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Home Agent 1 Debug (rev 02)
7f:13.0 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Integrated Memory Controller
↪  0 Target Address, Thermal & RAS Registers (rev 02)
7f:13.1 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Integrated Memory Controller
↪  0 Target Address, Thermal & RAS Registers (rev 02)
7f:13.2 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Integrated Memory Controller
↪  0 Channel Target Address Decoder (rev 02)
7f:13.3 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Integrated Memory Controller
↪  0 Channel Target Address Decoder (rev 02)
7f:13.6 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 DDRIO Channel 0/1 Broadcast
↪  (rev 02)
7f:13.7 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 DDRIO Global Broadcast (rev
↪  02)
7f:14.0 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Integrated Memory
↪  Controller 0 Channel 0 Thermal Control (rev 02)
7f:14.1 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Integrated Memory
↪  Controller 0 Channel 1 Thermal Control (rev 02)
7f:14.2 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Integrated Memory
↪  Controller 0 Channel 0 ERROR Registers (rev 02)
7f:14.3 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Integrated Memory
↪  Controller 0 Channel 1 ERROR Registers (rev 02)
7f:14.4 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 DDRIO (VMSE) 0 & 1 (rev 02)
7f:14.5 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 DDRIO (VMSE) 0 & 1 (rev 02)
7f:14.6 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 DDRIO (VMSE) 0 & 1 (rev 02)
7f:14.7 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 DDRIO (VMSE) 0 & 1 (rev 02)
7f:16.0 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Integrated Memory Controller
↪  1 Target Address, Thermal & RAS Registers (rev 02)
7f:16.1 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Integrated Memory Controller
↪  1 Target Address, Thermal & RAS Registers (rev 02)
7f:16.2 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Integrated Memory Controller
↪  1 Channel Target Address Decoder (rev 02)
7f:16.3 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Integrated Memory Controller
↪  1 Channel Target Address Decoder (rev 02)
7f:16.6 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 DDRIO Channel 2/3 Broadcast
↪  (rev 02)
7f:16.7 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 DDRIO Global Broadcast (rev
↪  02)
7f:17.0 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Integrated Memory
↪  Controller 1 Channel 0 Thermal Control (rev 02)
7f:17.1 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Integrated Memory
↪  Controller 1 Channel 1 Thermal Control (rev 02)
7f:17.2 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Integrated Memory
↪  Controller 1 Channel 0 ERROR Registers (rev 02)
7f:17.3 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Integrated Memory
↪  Controller 1 Channel 1 ERROR Registers (rev 02)
```

```
7f:17.4 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 DDRIO (VMSE) 2 & 3 (rev 02)
7f:17.5 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 DDRIO (VMSE) 2 & 3 (rev 02)
7f:17.6 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 DDRIO (VMSE) 2 & 3 (rev 02)
7f:17.7 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 DDRIO (VMSE) 2 & 3 (rev 02)
7f:1e.0 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Power Control Unit (rev 02)
7f:1e.1 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Power Control Unit (rev 02)
7f:1e.2 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Power Control Unit (rev 02)
7f:1e.3 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Power Control Unit (rev 02)
7f:1e.4 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Power Control Unit (rev 02)
7f:1f.0 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 VCU (rev 02)
7f:1f.2 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 VCU (rev 02)
80:01.0 PCI bridge: Intel Corporation Xeon E7 v3/Xeon
↪  E5 v3/Core i7 PCI Express Root Port 1 (rev 02)
80:02.0 PCI bridge: Intel Corporation Xeon E7 v3/Xeon
↪  E5 v3/Core i7 PCI Express Root Port 2 (rev 02)
80:03.0 PCI bridge: Intel Corporation Xeon E7 v3/Xeon
↪  E5 v3/Core i7 PCI Express Root Port 3 (rev 02)
80:03.2 PCI bridge: Intel Corporation Xeon E7 v3/Xeon
↪  E5 v3/Core i7 PCI Express Root Port 3 (rev 02)
80:05.0 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Address Map, VTd_Misc,
↪  System Management (rev 02)
80:05.1 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Hot Plug (rev 02)
80:05.2 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 RAS, Control Status and
↪  Global Errors (rev 02)
80:05.4 PIC: Intel Corporation Xeon E7 v3/Xeon E5
↪  v3/Core i7 I/O APIC (rev 02)
81:00.0 Network controller: Mellanox Technologies
↪  MT27500 Family [ConnectX-3]
82:00.0 3D controller: NVIDIA Corporation GV100GL
↪  [Tesla V100 PCIe 32GB] (rev a1)
ff:08.0 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 QPI Link 0 (rev 02)
ff:08.2 Performance counters: Intel Corporation Xeon
↪  E7 v3/Xeon E5 v3/Core i7 QPI Link 0 (rev 02)
ff:08.3 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 QPI Link 0 (rev 02)
ff:09.0 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 QPI Link 1 (rev 02)
ff:09.2 Performance counters: Intel Corporation Xeon
↪  E7 v3/Xeon E5 v3/Core i7 QPI Link 1 (rev 02)
ff:09.3 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 QPI Link 1 (rev 02)
ff:0b.0 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 R3 QPI Link 0 & 1 Monitoring
↪  (rev 02)
ff:0b.1 Performance counters: Intel Corporation Xeon
↪  E7 v3/Xeon E5 v3/Core i7 R3 QPI Link 0 & 1
↪  Monitoring (rev 02)
ff:0b.2 Performance counters: Intel Corporation Xeon
↪  E7 v3/Xeon E5 v3/Core i7 R3 QPI Link 0 & 1
↪  Monitoring (rev 02)
ff:0c.0 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Unicast Registers (rev 02)
ff:0c.1 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Unicast Registers (rev 02)
ff:0c.2 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Unicast Registers (rev 02)
ff:0c.3 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Unicast Registers (rev 02)
ff:0c.4 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Unicast Registers (rev 02)
ff:0c.5 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Unicast Registers (rev 02)
ff:0c.6 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Unicast Registers (rev 02)
ff:0c.7 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Unicast Registers (rev 02)
ff:0d.0 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Unicast Registers (rev 02)
ff:0d.1 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Unicast Registers (rev 02)
ff:0f.0 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Buffered Ring Agent (rev 02)
ff:0f.1 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Buffered Ring Agent (rev 02)
ff:0f.2 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Buffered Ring Agent (rev 02)
ff:0f.3 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 Buffered Ring Agent (rev 02)
ff:0f.4 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 System Address Decoder &
↪  Broadcast Registers (rev 02)
ff:0f.5 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 System Address Decoder &
↪  Broadcast Registers (rev 02)
ff:0f.6 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 System Address Decoder &
↪  Broadcast Registers (rev 02)
ff:10.0 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 PCIe Ring Interface (rev 02)
ff:10.1 Performance counters: Intel Corporation Xeon
↪  E7 v3/Xeon E5 v3/Core i7 PCIe Ring Interface (rev
↪  02)
```

```
ff:10.5 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Scratchpad & Semaphore
↪ Registers (rev 02)
ff:10.6 Performance counters: Intel Corporation Xeon
↪ E7 v3/Xeon E5 v3/Core i7 Scratchpad & Semaphore
↪ Registers (rev 02)
ff:10.7 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Scratchpad & Semaphore
↪ Registers (rev 02)
ff:12.0 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Home Agent 0 (rev 02)
ff:12.1 Performance counters: Intel Corporation Xeon
↪ E7 v3/Xeon E5 v3/Core i7 Home Agent 0 (rev 02)
ff:12.2 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Home Agent 0 Debug (rev 02)
ff:12.4 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Home Agent 1 (rev 02)
ff:12.5 Performance counters: Intel Corporation Xeon
↪ E7 v3/Xeon E5 v3/Core i7 Home Agent 1 (rev 02)
ff:12.6 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Home Agent 1 Debug (rev 02)
ff:13.0 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Integrated Memory Controller
↪ 0 Target Address, Thermal & RAS Registers (rev 02)
ff:13.1 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Integrated Memory Controller
↪ 0 Target Address, Thermal & RAS Registers (rev 02)
ff:13.2 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Integrated Memory Controller
↪ 0 Channel Target Address Decoder (rev 02)
ff:13.3 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Integrated Memory Controller
↪ 0 Channel Target Address Decoder (rev 02)
ff:13.6 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 DDRIO Channel 0/1 Broadcast
↪ (rev 02)
ff:13.7 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 DDRIO Global Broadcast (rev
↪ 02)
ff:14.0 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Integrated Memory
↪ Controller 0 Channel 0 Thermal Control (rev 02)
ff:14.1 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Integrated Memory
↪ Controller 0 Channel 1 Thermal Control (rev 02)
ff:14.2 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Integrated Memory
↪ Controller 0 Channel 0 ERROR Registers (rev 02)
ff:14.3 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Integrated Memory
↪ Controller 0 Channel 1 ERROR Registers (rev 02)
ff:14.4 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 DDRIO (VMSE) 0 & 1 (rev 02)
ff:14.5 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 DDRIO (VMSE) 0 & 1 (rev 02)
ff:14.6 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 DDRIO (VMSE) 0 & 1 (rev 02)
ff:14.7 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 DDRIO (VMSE) 0 & 1 (rev 02)
ff:16.0 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Integrated Memory Controller
↪ 1 Target Address, Thermal & RAS Registers (rev 02)
ff:16.1 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Integrated Memory Controller
↪ 1 Target Address, Thermal & RAS Registers (rev 02)
ff:16.2 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Integrated Memory Controller
↪ 1 Channel Target Address Decoder (rev 02)
ff:16.3 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Integrated Memory Controller
↪ 1 Channel Target Address Decoder (rev 02)
ff:16.6 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 DDRIO Channel 2/3 Broadcast
↪ (rev 02)
ff:16.7 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 DDRIO Global Broadcast (rev
↪ 02)
ff:17.0 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Integrated Memory
↪ Controller 1 Channel 0 Thermal Control (rev 02)
ff:17.1 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Integrated Memory
↪ Controller 1 Channel 1 Thermal Control (rev 02)
ff:17.2 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Integrated Memory
↪ Controller 1 Channel 0 ERROR Registers (rev 02)
ff:17.3 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Integrated Memory
↪ Controller 1 Channel 1 ERROR Registers (rev 02)
ff:17.4 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 DDRIO (VMSE) 2 & 3 (rev 02)
ff:17.5 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 DDRIO (VMSE) 2 & 3 (rev 02)
ff:17.6 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 DDRIO (VMSE) 2 & 3 (rev 02)
ff:17.7 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 DDRIO (VMSE) 2 & 3 (rev 02)
ff:1e.0 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Power Control Unit (rev 02)
ff:1e.1 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Power Control Unit (rev 02)
ff:1e.2 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Power Control Unit (rev 02)
ff:1e.3 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Power Control Unit (rev 02)
ff:1e.4 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 Power Control Unit (rev 02)
ff:1f.0 System peripheral: Intel Corporation Xeon E7
↪ v3/Xeon E5 v3/Core i7 VCU (rev 02)
```

```
ff:1f.2 System peripheral: Intel Corporation Xeon E7
↪  v3/Xeon E5 v3/Core i7 VCU (rev 02)

Output from node-level machine 2: argon-knl:
LESSOPEN=| /usr/bin/lesspipe %s
SLURM_STEP_NODELIST=argon-knl
MAIL=/var/mail/USER
SSH_CLIENT=129.69.216.249 51664 22
USER=USER
SLURM_JOBID=13966
SLURM_JOB_USER=USER
LC_TIME=de_DE.UTF-8
GIT_EDITOR=emacsclient -nw
SLURM_PTY_PORT=39606
SRUN_DEBUG=3
FZF_DEFAULT_OPTS=--reverse --border
SLURM_SRUN_COMM_PORT=38551
LD_LIBRARY_PATH=/scratch/USER/supertiger/build/hdf5/
↪  lib:/home/USER/Work/SGpp/lshknn/build:/home/USER
↪  /Work/AutoTuneTMP/boost_install/lib:/home/USER/W
↪  ork/SGpp/lib/sgpp:/media/g/Volume/Programmieren/
↪  IPVS-Hiwi/SGpp/lib/sgpp:/home/USER/Work/SGpp/lsh
↪  knn/build:/home/USER/Work/AutoTuneTMP/boost_inst
↪  all/lib:/home/USER/Work/SGpp/lib/sgpp:/media/g/V
↪  olume/Programmieren/IPVS-Hiwi/SGpp/lib/sgpp:
SHLVL=2
SLURM_JOB_NUM_NODES=1
SLURM_TASKS_PER_NODE=1
HOME=/home/USER
LESS=-R
OLDPWD=/scratch/USER/supertiger
SLURM_TOPOLOGY_ADDR_PATTERN=node
SSH_TTY=/dev/pts/2
ZSH=/home/USER/oh-my-zsh
LSCOLORS=Gxfxcxdxbxegedabagacad
SLURM_PRIO_PROCESS=0
PAGER=less
LC_MONETARY=de_DE.UTF-8
LC_CTYPE=en_US.UTF-8
SLURM_JOB_NAME=bash
SLURM_JOB_CPUS_PER_NODE=256
SLURM_CPUS_ON_NODE=256
SLURM_PROCID=0
TMPDIR=/tmp
SLURM_STEP_LAUNCHER_PORT=38551
CUDA_VISIBLE_DEVICES=NoDevFiles
FZF_TMUX_HEIGHT=50%
LOGNAME=USER
SLURM_SUBMIT_HOST=argon-fs
_=/scratch/USER/supertiger/src/octotiger/./collect-e
↪  nviroment.sh
TMP_PROMPT=[%{$fg[$NCOLOR]%}%B%n%b%{$reset_color%}:%
↪  {$fg[red]%}%30<...<%~%<<%{$reset_color%}]%(!.#.$)
SLURM_NODELIST=argon-knl
XDG_SESSION_ID=3457
TERM=screen-256color

SLURM_NNODES=1
SLURM_JOB_ID=13966
GPU_DEVICE_ORDINAL=NoDevFiles
SLURMD_NODENAME=argon-knl
SLURM_JOB_NODELIST=argon-knl
PATH=/home/USER/external_packages/cask/bin:/home/USE
↪  R/bin:/usr/local/bin:/home/USER/.node/bin:/home/
↪  USER/packages/cask/bin:/home/USER/Skripte:/home/
↪  USER/external_packages/FlameGraph:/home/USER/.no
↪  de/bin:/home/USER/Skripte:/home/USER/external_pa
↪  ckages/cask/bin:/home/USER/bin:/usr/local/bin:/h
↪  ome/USER/.node/bin:/home/USER/packages/cask/bin:
↪  /home/USER/Skripte:/home/USER/external_packages/
↪  FlameGraph:/home/USER/.node/bin:/home/USER/Skrip
↪  te:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr
↪  /bin:/sbin:/bin:/usr/games:/usr/local/games:/sna
↪  p/bin:/home/USER/.fzf/bin
SLURM_STEPID=0
SLURM_GTIDS=0
LC_ADDRESS=de_DE.UTF-8
XDG_RUNTIME_DIR=/run/user/20239
SLURM_CHECKPOINT_IMAGE_DIR=/var/slurm/checkpoint
FZF_TMUX=1
LC_TELEPHONE=de_DE.UTF-8
SLURM_STEP_NUM_NODES=1
LANG=en_US.UTF-8
SLURM_DISTRIBUTION=cyclic
SLURM_PTY_WIN_ROW=68
LS_COLORS=no=00:fi=00:di=01;34:ln=00;36:pi=40;33:so=
↪  01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=41;33;
↪  01:ex=00;32:*.cmd=00;32:*.exe=01;32:*.com=01;32:
↪  *.bat=01;32:*.btm=01;32:*.dll=01;32:*.tar=00;31:
↪  *.tbz=00;31:*.tgz=00;31:*.rpm=00;31:*.deb=00;31:
↪  *.arj=00;31:*.taz=00;31:*.lzh=00;31:*.lzma=00;31
↪  :*.zip=00;31:*.zoo=00;31:*.z=00;31:*.Z=00;31:*.g
↪  z=00;31:*.bz2=00;31:*.tb2=00;31:*.tz2=00;31:*.tb
↪  z2=00;31:*.avi=01;35:*.bmp=01;35:*.fli=01;35:*.g
↪  if=01;35:*.jpg=01;35:*.jpeg=01;35:*.mng=01;35:*.
↪  mov=01;35:*.mpg=01;35:*.pcx=01;35:*.pbm=01;35:*.
↪  pgm=01;35:*.png=01;35:*.ppm=01;35:*.tga=01;35:*.
↪  tif=01;35:*.xbm=01;35:*.xpm=01;35:*.dl=01;35:*.g
↪  l=01;35:*.wmv=01;35:*.aiff=00;32:*.au=00;32:*.mi
↪  d=00;32:*.mp3=00;32:*.ogg=00;32:*.voc=00;32:*.wa
↪  v=00;32:
SSH_AUTH_SOCK=/tmp/ssh-woeilFazTv/agent.23567
FZF_COMPLETION_TRIGGER=]]
SLURM_JOB_UID=20239
SLURM_CLUSTER_NAME=argon
SHELL=/bin/bash
NODE_PATH=/home/USER/.node/lib/node_modules:/home/US
↪  ER/.node/lib/node_modules:/home/USER/.node/lib/n
↪  ode_modules:/home/USER/.node/lib/node_modules:
SLURM_STEP_TASKS_PER_NODE=1
LC_NAME=de_DE.UTF-8
SLURM_LOCALID=0
LESSCLOSE=/usr/bin/lesspipe %s %s
```

SLURM_LAUNCH_NODE_IPADDR=129.69.213.243
MODULE_VERSION=3.2.9
LC_MEASUREMENT=de_DE.UTF-8
SLURM_JOB_PARTITION=knl
MODULE_VERSION_STACK=3.2.9
LC_IDENTIFICATION=de_DE.UTF-8
SLURM_TASK_PID=94483
SLURM_NTASKS=1
SLURM_TOPOLOGY_ADDR=argon-knl
LC_ALL=en_US.UTF-8
PWD=/scratch/USER/supertiger/src/octotiger
LOADEDMODULES=
SLURM_NPROCS=1
FZF_DEFAULT_COMMAND=ag -l -g ""
KEYTIMEOUT=1
SSH_CONNECTION=129.69.216.249 51664 129.69.213.243 22
XDG_DATA_DIRS=/usr/local/share:/usr/share:/var/lib/s⌋
↪  napd/desktop
LC_NUMERIC=de_DE.UTF-8
PYTHONPATH=/home/USER/Work/SGpp/lib:/media/g/Volume/⌋
↪  Programmieren/IPVS-Hiwi/SGpp/lib:/home/USER/Work⌋
↪  /SGpp/lib:/media/g/Volume/Programmieren/IPVS-Hiw⌋
↪  i/SGpp/lib:
LC_PAPER=de_DE.UTF-8
SLURM_SRUN_COMM_HOST=129.69.213.243
MODULEPATH=/usr/local.nfs/Modules/modulefiles
SLURM_STEP_NUM_TASKS=1
SLURM_STEP_ID=0
EDITOR=vim
SLURM_PTY_WIN_COL=273
SLURM_NODEID=0
SLURM_SUBMIT_DIR=/home/USER
MODULESHOME=/usr/local.nfs/Modules/3.2.9
Distributor ID:        Ubuntu
Description:        Ubuntu 16.04.6 LTS
Release:        16.04
Codename:        xenial
Linux argon-knl 4.4.0-143-generic #169-Ubuntu SMP Thu
↪  Feb 7 07:56:38 UTC 2019 x86_64 x86_64 x86_64
↪  GNU/Linux
Architecture:        x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:        Little Endian
CPU(s):        256
On-line CPU(s) list:  0-255
Thread(s) per core:  4
Core(s) per socket:  64
Socket(s):        1
NUMA node(s):        1
Vendor ID:        GenuineIntel
CPU family:        6
Model:        87
Model name:        Intel(R) Xeon Phi(TM) CPU 7210
↪  @ 1.30GHz
Stepping:        1
CPU MHz:        1027.609

CPU max MHz:        1500.0000
CPU min MHz:        1000.0000
BogoMIPS:        2599.83
L1d cache:        32K
L1i cache:        32K
L2 cache:        1024K
NUMA node0 CPU(s):    0-255
Flags:        fpu vme de pse tsc msr pae mce
↪  cx8 apic sep mtrr pge mca cmov pat pse36 clflush
↪  dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx
↪  pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs
↪  bts rep_good nopl xtopology nonstop_tsc
↪  aperfmperf pni pclmulqdq dtes64 monitor ds_cpl
↪  est tm2 ssse3 fma cx16 xtpr pdcm sse4_1 sse4_2
↪  x2apic movbe popcnt tsc_deadline_timer aes xsave
↪  avx f16c rdrand lahf_lm abm 3dnowprefetch epb
↪  kaiser fsgsbase tsc_adjust bmi1 avx2 smep bmi2
↪  erms avx512f rdseed adx avx512pf avx512er
↪  avx512cd xsaveopt dtherm ida arat pln pts
MemTotal:        98845136 kB
MemFree:        78524404 kB
MemAvailable:    96356720 kB
Buffers:        675324 kB
Cached:        16631328 kB
SwapCached:        0 kB
Active:        16648584 kB
Inactive:        681972 kB
Active(anon):    27152 kB
Inactive(anon):  36252 kB
Active(file):    16621432 kB
Inactive(file):  645720 kB
Unevictable:        3652 kB
Mlocked:        3652 kB
SwapTotal:        4194300 kB
SwapFree:        4194300 kB
Dirty:        4 kB
Writeback:        0 kB
AnonPages:        27664 kB
Mapped:        47888 kB
Shmem:        37052 kB
Slab:        1745276 kB
SReclaimable:    1225968 kB
SUnreclaim:        519308 kB
KernelStack:        35152 kB
PageTables:        3036 kB
NFS_Unstable:        0 kB
Bounce:        0 kB
WritebackTmp:        0 kB
CommitLimit:        53616868 kB
Committed_AS:    228420 kB
VmallocTotal:    34359738367 kB
VmallocUsed:        0 kB
VmallocChunk:        0 kB
HardwareCorrupted:  0 kB
AnonHugePages:    0 kB
CmaTotal:        0 kB
CmaFree:        0 kB

```
HugePages_Total:        0
HugePages_Free:         0
HugePages_Rsvd:         0
HugePages_Surp:         0
Hugepagesize:        2048 kB
DirectMap4k:       531164 kB
DirectMap2M:     15071232 kB
DirectMap1G:     87031808 kB
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda           8:0    0  477G  0 disk
└─sda1        8:1    0  477G  0 part
  ├─vg0-lv01 252:0   0   15G  0 lvm  /
  ├─vg0-lv00 252:1   0    4G  0 lvm  [SWAP]
  ├─vg0-lv02 252:2   0    4G  0 lvm  /var
  ├─vg0-lv03 252:3   0    2G  0 lvm  /tmp
  └─vg0-lv04 252:4   0  100G  0 lvm  /data/scratch
sdb          8:16    0  1.8T  0 disk
└─sdb1       8:17    0  1.8T  0 part
  └─vg1-lv10 252:5   0    1T  0 lvm  /data/scratch2
loop0         7:0    0         0 loop
loop1         7:1    0         0 loop
loop2         7:2    0         0 loop
loop3         7:3    0         0 loop
loop4         7:4    0         0 loop
loop5         7:5    0         0 loop
loop6         7:6    0         0 loop
loop7         7:7    0         0 loop
H/W path        Device Class         Description
==================================================
                       system        Computer
/0                     bus           Motherboard
/0/0                   memory        94GiB System
↪  memory
/0/1                   processor     Intel(R) Xeon
↪  Phi(TM) CPU 7210 @ 1.30GHz
/0/100                 bridge        Intel Corporation
/0/100/0.2             bridge        Intel
↪  Corporation
/0/100/0.2/0    eth0   network       Ethernet
↪  Controller 10G X550T
/0/100/0.2/0.1  eth1   network       Ethernet
↪  Controller 10G X550T
/0/100/5               generic       Intel
↪  Corporation
/0/100/5.2             generic       Intel
↪  Corporation
/0/100/5.4             generic       Intel
↪  Corporation
/0/100/5.6             generic       Intel
↪  Corporation
/0/100/8               generic       Intel
↪  Corporation
/0/100/8.1             generic       Intel
↪  Corporation
```

```
/0/100/8.2             generic       Intel
↪  Corporation
/0/100/11              generic       C610/X99 series
↪  chipset SPSR
/0/100/14              bus           C610/X99 series
↪  chipset USB xHCI Host Controller
/0/100/16              communication C610/X99
↪  series chipset MEI Controller #1
/0/100/16.1            communication C610/X99
↪  series chipset MEI Controller #2
/0/100/1a              bus           C610/X99 series
↪  chipset USB Enhanced Host Controller #2
/0/100/1c              bridge        C610/X99 series
↪  chipset PCI Express Root Port #1
/0/100/1c.3            bridge        C610/X99
↪  series chipset PCI Express Root Port #4
/0/100/1c.3/0          bridge        AST1150
↪  PCI-to-PCI Bridge
/0/100/1c.3/0/0        display       ASPEED
↪  Graphics Family
/0/100/1c.4            bridge        C610/X99
↪  series chipset PCI Express Root Port #5
/0/100/1c.4/0   eth2   network       I350 Gigabit
↪  Network Connection
/0/100/1c.4/0.1 eth3   network       I350 Gigabit
↪  Network Connection
/0/100/1d              bus           C610/X99 series
↪  chipset USB Enhanced Host Controller #1
/0/100/1f              bridge        C610/X99 series
↪  chipset LPC Controller
/0/100/1f.2            storage       C610/X99
↪  series chipset 6-Port SATA Controller [AHCI mode]
/0/100/1f.3            bus           C610/X99 series
↪  chipset SMBus Controller
/0/100/1f.6            generic       C610/X99
↪  series chipset Thermal Subsystem
/0/2                   generic       Intel Corporation
/0/3                   generic       Intel Corporation
/0/4                   generic       Intel Corporation
/0/5                   generic       Intel Corporation
/0/6                   generic       Intel Corporation
/0/8.5                 generic       Intel
↪  Corporation
/0/8.6                 generic       Intel
↪  Corporation
/0/8.7                 generic       Intel
↪  Corporation
/0/7                   generic       Intel Corporation
/0/8                   generic       Intel Corporation
/0/9                   generic       Intel Corporation
/0/a                   generic       Intel Corporation
/0/b                   generic       Intel Corporation
/0/9.5                 generic       Intel
↪  Corporation
```

From Piz Daint to the Stars: Simulation of Stellar Mergers Using High-Level Abstractions

| | | |
|---|---|---|
| /0/9.6 | generic | Intel Corporation |
| /0/9.7 | generic | Intel Corporation |
| /0/c | generic | Intel Corporation |
| /0/d | generic | Intel Corporation |
| /0/f | generic | Intel Corporation |
| /0/a.3 | generic | Intel Corporation |
| /0/a.4 | generic | Intel Corporation |
| /0/a.5 | generic | Intel Corporation |
| /0/a.6 | generic | Intel Corporation |
| /0/a.7 | generic | Intel Corporation |
| /0/10 | generic | Intel Corporation |
| /0/11 | generic | Intel Corporation |
| /0/12 | generic | Intel Corporation |
| /0/b.3 | generic | Intel Corporation |
| /0/b.4 | generic | Intel Corporation |
| /0/b.5 | generic | Intel Corporation |
| /0/b.6 | generic | Intel Corporation |
| /0/b.7 | generic | Intel Corporation |
| /0/14 | generic | Intel Corporation |
| /0/15 | generic | Intel Corporation |
| /0/16 | generic | Intel Corporation |
| /0/c.3 | generic | Intel Corporation |
| /0/c.4 | generic | Intel Corporation |
| /0/c.5 | generic | Intel Corporation |
| /0/e | generic | Intel Corporation |
| /0/e.1 | generic | Intel Corporation |
| /0/e.2 | generic | Intel Corporation |
| /0/e.3 | generic | Intel Corporation |
| /0/e.4 | generic | Intel Corporation |
| /0/e.5 | generic | Intel Corporation |
| /0/e.6 | generic | Intel Corporation |
| /0/e.7 | generic | Intel Corporation |
| /0/18 | generic | Intel Corporation |
| /0/1d | generic | Intel Corporation |
| /0/1e | generic | Intel Corporation |
| /0/f.3 | generic | Intel Corporation |
| /0/f.4 | generic | Intel Corporation |
| /0/f.5 | generic | Intel Corporation |
| /0/f.6 | generic | Intel Corporation |
| /0/f.7 | generic | Intel Corporation |
| /0/20 | generic | Intel Corporation |
| /0/21 | generic | Intel Corporation |
| /0/22 | generic | Intel Corporation |
| /0/10.3 | generic | Intel Corporation |
| /0/10.4 | generic | Intel Corporation |
| /0/23 | generic | Intel Corporation |
| /0/24 | generic | Intel Corporation |
| /0/25 | generic | Intel Corporation |
| /0/26 | generic | Intel Corporation |
| /0/27 | generic | Intel Corporation |
| /0/28 | generic | Intel Corporation |
| /0/11.3 | generic | Intel Corporation |
| /0/11.4 | generic | Intel Corporation |
| /0/29 | generic | Intel Corporation |
| /0/2a | generic | Intel Corporation |
| /0/2b | generic | Intel Corporation |
| /0/2c | generic | Intel Corporation |
| /0/2d | generic | Intel Corporation |
| /0/2e | generic | Intel Corporation |
| /0/12.3 | generic | Intel Corporation |
| /0/12.4 | generic | Intel Corporation |
| /0/2f | generic | Intel Corporation |
| /0/30 | generic | Intel Corporation |
| /0/31 | generic | Intel Corporation |
| /0/32 | generic | Intel Corporation |
| /0/14.3 | generic | Intel Corporation |
| /0/14.4 | generic | Intel Corporation |
| /0/14.5 | generic | Intel Corporation |
| /0/33 | generic | Intel Corporation |
| /0/34 | generic | Intel Corporation |
| /0/35 | generic | Intel Corporation |
| /0/36 | generic | Intel Corporation |
| /0/37 | generic | Intel Corporation |

| Device | Type | Vendor |
|---|---|---|
| /0/15.3 | generic | Intel Corporation |
| /0/15.4 | generic | Intel Corporation |
| /0/15.5 | generic | Intel Corporation |
| /0/38 | generic | Intel Corporation |
| /0/39 | generic | Intel Corporation |
| /0/3a | generic | Intel Corporation |
| /0/3b | generic | Intel Corporation |
| /0/3c | generic | Intel Corporation |
| /0/16.3 | generic | Intel Corporation |
| /0/16.4 | generic | Intel Corporation |
| /0/16.5 | generic | Intel Corporation |
| /0/16.6 | generic | Intel Corporation |
| /0/16.7 | generic | Intel Corporation |
| /0/17 | generic | Intel Corporation |
| /0/17.1 | generic | Intel Corporation |
| /0/17.2 | generic | Intel Corporation |
| /0/17.3 | generic | Intel Corporation |
| /0/17.4 | generic | Intel Corporation |
| /0/17.5 | generic | Intel Corporation |
| /0/17.6 | generic | Intel Corporation |
| /0/17.7 | generic | Intel Corporation |
| /0/3d | generic | Intel Corporation |
| /0/3e | generic | Intel Corporation |
| /0/3f | generic | Intel Corporation |
| /0/18.3 | generic | Intel Corporation |
| /0/18.4 | generic | Intel Corporation |
| /0/18.5 | generic | Intel Corporation |
| /0/40 | generic | Intel Corporation |
| /0/41 | generic | Intel Corporation |
| /0/42 | generic | Intel Corporation |
| /0/1d.3 | generic | Intel Corporation |
| /0/43 | generic | Intel Corporation |
| /0/44 | generic | Intel Corporation |
| /0/45 | generic | Intel Corporation |
| /0/1e.3 | generic | Intel Corporation |
| /0/1e.4 | generic | Intel Corporation |
| /0/1e.5 | generic | Intel Corporation |
| /0/1e.6 | generic | Intel Corporation |
| /0/46 | generic | Intel Corporation |
| /0/47 | generic | Intel Corporation |
| /0/48 | generic | Intel Corporation |
| /0/49 | generic | Intel Corporation |
| /0/4a | generic | Intel Corporation |
| /0/4b | generic | Intel Corporation |
| /0/4c | generic | Intel Corporation |
| /0/4d | generic | Intel Corporation |
| /0/4e | generic | Intel Corporation |
| /0/4f | generic | Intel Corporation |
| /0/50 | generic | Intel Corporation |
| /0/51 | generic | Intel Corporation |
| /0/52 | generic | Intel Corporation |
| /0/53 | generic | Intel Corporation |
| /0/54 | generic | Intel Corporation |
| /0/55 | generic | Intel Corporation |
| /0/56 | generic | Intel Corporation |
| /0/57 | generic | Intel Corporation |
| /0/58 | generic | Intel Corporation |
| /0/59 | generic | Intel Corporation |
| /0/5a | generic | Intel Corporation |
| /0/5b | generic | Intel Corporation |
| /0/5c | generic | Intel Corporation |
| /0/5d | generic | Intel Corporation |
| /0/5e | generic | Intel Corporation |
| /0/5f | generic | Intel Corporation |
| /0/60 | generic | Intel Corporation |
| /0/61 | generic | Intel Corporation |
| /0/62 | generic | Intel Corporation |
| /0/63 | generic | Intel Corporation |
| /0/64 | generic | Intel Corporation |
| /0/65 | generic | Intel Corporation |
| /0/66 | generic | Intel Corporation |
| /0/67 | generic | Intel Corporation |
| /0/68 | generic | Intel Corporation |
| /0/69 | generic | Intel Corporation |
| /0/6a | generic | Intel Corporation |
| /0/6b | generic | Intel Corporation |
| /0/12.6 | generic | Intel Corporation |
| /0/12.7 | generic | Intel Corporation |
| /0/13 | generic | Intel Corporation |
| /0/13.1 | generic | Intel Corporation |
| /0/13.2 | generic | Intel Corporation |
| /0/13.5 | generic | Intel Corporation |

```
/0/13.6                 generic         Intel
↪ Corporation
/0/13.7                 generic         Intel
↪ Corporation
/0/6c           generic     Intel Corporation
/0/6d           generic     Intel Corporation
/0/6e           generic     Intel Corporation
/0/6f           generic     Intel Corporation
/0/70           generic     Intel Corporation
/0/71           generic     Intel Corporation
/0/72           generic     Intel Corporation
/0/73           generic     Intel Corporation
/0/74           generic     Intel Corporation
/0/75           generic     Intel Corporation
/0/76           generic     Intel Corporation
/0/77           generic     Intel Corporation
/0/78           generic     Intel Corporation
/0/79           generic     Intel Corporation
/0/7a           generic     Intel Corporation
/0/7b           generic     Intel Corporation
/0/19           generic     Intel Corporation
/0/19.1                 generic         Intel
↪ Corporation
/0/19.2                 generic         Intel
↪ Corporation
/0/1a           generic     Intel Corporation
/0/1a.1                 generic         Intel
↪ Corporation
/0/1a.2                 generic         Intel
↪ Corporation
/0/1b           generic     Intel Corporation
/0/1b.1                 generic         Intel
↪ Corporation
/0/1b.2                 generic         Intel
↪ Corporation
/0/1c           generic     Intel Corporation
/0/1c.1                 generic         Intel
↪ Corporation
/0/1c.2                 generic         Intel
↪ Corporation
/0/7c           generic     Intel Corporation
/0/7d           generic     Intel Corporation
/0/7e           generic     Intel Corporation
/0/7f           generic     Intel Corporation
/0/80           generic     Intel Corporation
/0/81           generic     Intel Corporation
/0/1f           generic     Intel Corporation
/0/1f.1                 generic         Intel
↪ Corporation
/0/1f.2                 generic         Intel
↪ Corporation
/1 bond0 network Ethernet interface
```