Scalable Simulation of Realistic Volume Fraction Red Blood Cell Flows through Vascular Networks

Libin Lu*
Courant Institute of Mathematical Sciences
New York University
New York, NY
libin@cs.nyu.edu

Matthew J. Morse*
Courant Institute of Mathematical Sciences
New York University
New York, NY
mmorse@cs.nyu.edu

Abtin Rahimian
Department of Computer Science
University of Colorado Boulder
Boulder, CO
arahimian@acm.org

Georg Stadler
Courant Institute of Mathematical Sciences
New York University
New York, NY
stadler@cims.nyu.edu

Denis Zorin
Courant Institute of Mathematical Sciences
New York University
New York, NY
dzorin@cs.nyu.edu

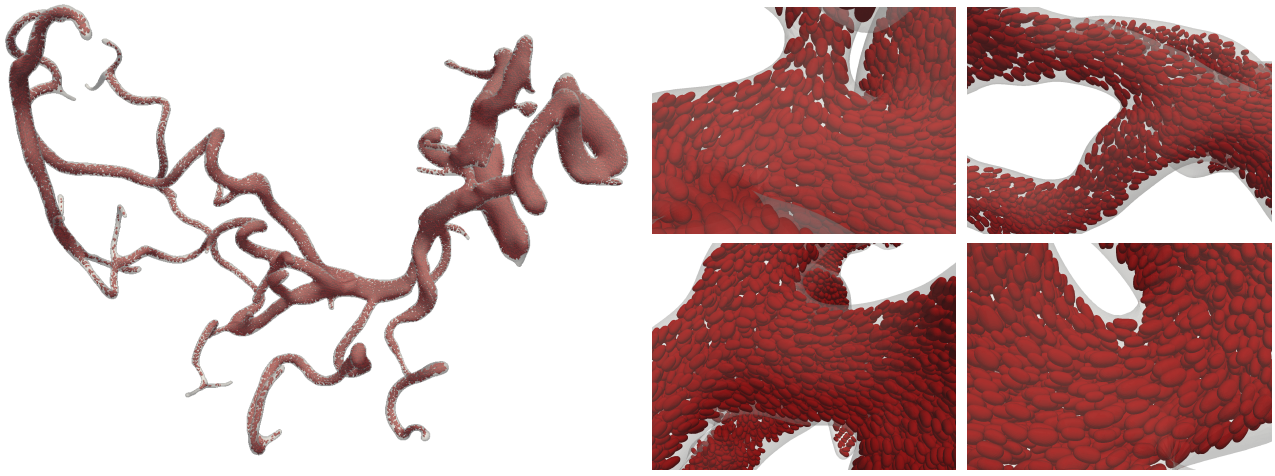


Figure 1: *Simulation results for 40,960 RBCs in a complex vessel geometry. For our strong scaling experiments, we use the vessel geometry shown on the left, with inflow-outflow boundary conditions at various regions of the vessel geometry. To setup the problem, we fill the vessel with nearly-touching RBCs of different sizes. The figure above shows a setup with overall 40,960 RBCs at a volume fraction of 19%, and 40,960 polynomial patches. The full simulation video is available at https://vimeo.com/329509229.*

ABSTRACT

High-resolution blood flow simulations have potential for developing better understanding biophysical phenomena at the microscale, such as vasodilation, vasoconstriction and overall vascular resistance. To this end, we present a scalable platform for the simulation of red blood cell (RBC) flows through complex capillaries by modeling the physical system as a viscous fluid with immersed deformable particles. We describe a parallel boundary integral equation solver for general elliptic partial differential equations, which we apply to Stokes flow through blood vessels. We also detail a parallel collision avoiding algorithm to ensure RBCs and the blood vessel remain contact-free. We have scaled our code on Stampede2 at the Texas Advanced Computing Center up to 34,816 cores. Our largest simulation enforces a contact-free state between four billion surface elements and solves for three billion degrees of freedom on one million RBCs and a blood vessel composed from two million patches.

*Both authors contributed equally to this research.

SC '19, November 17–22, 2019, Denver, CO, USA

ACM ISBN 978-1-4503-6229-0/19/11…$15.00
https://doi.org/10.1145/3295500.3356203

ACM Reference Format:
Libin Lu, Matthew J. Morse, Abtin Rahimian, Georg Stadler, and Denis Zorin. 2019. Scalable Simulation of Realistic Volume Fraction Red Blood Cell Flows through Vascular Networks. In *The International Conference for*

High Performance Computing, Networking, Storage, and Analysis (SC '19), November 17–22, 2019, Denver, CO, USA. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3295500.3356203

1 INTRODUCTION

The ability to simulate complex biological flows from first principles has the potential to provide insight into complicated physiological processes. Simulation of blood flow, in particular, is of paramount biological and clinical importance. Blood vessel constriction and dilation affects blood pressure, forces between RBCs can cause clotting, various cells migrate differently through microfluidic devices.

However, direct simulation of blood flow is an extremely challenging task. Even simulating the blood flow in smaller vessels requires modeling millions of cells (one microliter of blood contains around five million RBCs) along with a complex blood vessel. RBCs are highly deformable and cannot be well-approximated by rigid particles. The volume fraction of cells in human blood flow reaches 45%, which means that a very large fraction of cells are in close contact with other cells or vessel walls at any given time. These constraints preclude a large number of discretization points per cell and make an evolving mesh of the fluid domain impractical and costly at large scale.

Simulations capable of capturing these various types of flows faithfully must be

- *numerically accurate*, to solve the model equations without concern for numerical error;
- *robust*, to handle high-volume-fraction flows, close contact between cells and vessel walls, complex geometries, and long simulation times;
- *efficient and scalable*, to support a realistic number of cells in flows through complex blood vessels.

Achieving these objectives for a blood flow simulation requires that the system meets a number of stringent requirements. While previous work has made significant progress [25, 28, 37], we focus on several new infrastructure components essential for handling confined flows and arbitrarily long-time, high volume fractions RBC flows; in particular, our work is able to realize each of these goals.

We formulate the viscous flow in blood vessels as an integro-differential equation and make use of fast scalable summation algorithms for efficient implementation, as in prior RBC simulations [48]. This is the only approach to date that maintains high accuracy at the microscopic level while avoiding expensive discretization of fluid volume: all degrees of freedom reside on the surfaces of RBCs and blood vessels.

The most important novel aspects of our system include: (a) handling the RBC-blood vessel interaction with a fully parallel, high-order boundary integral equation solver; (b) explicit handling of collisions with a parallel constraint-based resolution and detection algorithm. The former is essential for modeling confined flows, while the latter is essential for handling high-volume fraction flows at long time scales without excessively small time steps or fine spatial discretizations.

Our contributions.

(1) We present a parallel platform for long-time simulations of RBCs through complex blood vessels. The extension to suspensions of various particulates (fibers, rigid bodies etc.) is straightforward from the boundary integral formulation. Flows through several complicated geometries are demonstrated.
(2) We have parallelized a boundary solver for elliptic PDEs on smooth complex geometries in 3D. By leveraging the parallel fast-multipole method of [26] and the parallel forest of quadtrees of [7], we are able to achieve good parallel performance and load balancing.
(3) We have extended the parallel collision handling of [25] to include rigid 3D boundaries composed of patches.
(4) We present weak and strong scalability results of our simulation on the Skylake cluster and weak scaling results on the Knights Landing cluster on Stampede2 at the Texas Advanced Computing Center along with several visualizations of long-time, large-scale blood cell flows through vessels. We observe 49% strong scaling efficiency for a 32-fold increase of compute cores. In our largest test on 12288 cores, we simulate 1,048,576 RBCs in a blood vessel composed of 2,097,152 patches with weak scaling efficiency of 71% compared to 192 cores (Fig. 5). In each time step, this test uses over three billion degrees of freedom and over four billion surface elements (triangles) for collision.
(5) We are able to simulate realistic human blood flows with RBC volume fractions over 47% (Fig. 7).

Limitations. Despite the advantages and contributions of the computational framework presented here, our work has some limitations. We have made several simplifications in our model for RBCs. We are restricted to the low Reynolds number regime, i.e., small arteries and capillaries. We use a simplified model for RBCs, assuming the cell membranes to be inextensible and with no in-plane shear rigidity. It has been shown that flows in arterioles and capillaries with diameter of <50 μm and RBCs with 5 μm diameter have a Reynolds number of $< 5 \times 10^{-3}$ [51][9, Section 5.4] with roughly 2% error in approximating confined flows [1]. This is sufficient for our interest in the qualitative behavior of particulate flows, with the possibility of investigating rheological dynamics in larger channels.

Regarding algorithms, each RBC is discretized with an equal number of points, despite the varied behavior of the velocity through the vessel. Adaptive refinement is required in order to resolve the velocity accurately. Finally, the blood vessel is constructed to satisfy certain geometric constraints that allow for the solution of Eq. (2.5) via singular integration. This can be overcome through uniform refinement, but a parallel adaptive algorithm is required to maintain good performance.

Related work: blood flow. Large-scale simulation of RBC flows typically fall into four categories: (a) *Immersed boundary (IB)* and *immersed interface methods*; (b) particle-based methods such as *dissipative particle dynamics (DPD)* and *smoothed particle hydrodynamics (SPH)* (c) multiscale network-based approaches and (d) *boundary integral equation (BIE)* approaches. For a comprehensive review of general blood flow simulation methods, see [12]. IB methods can produce high-quality simulations of heterogeneous particulate flows in complex blood vessels [3, 4, 52]. These methods typically require a finite element solve for each RBC to compute membrane

tensions and use IB to couple the stresses with the fluid. This approach quickly becomes costly, especially for high-order elements, and although reasonably large simulation have been achieved [41, 42], large-scale parallelization has remained a challenge. A different approach to simulating blood flow is with multiscale reduced-order models. By making simplifying assumptions about the fluid behavior throughout the domain and transforming the complex fluid system into a simpler flow problem, the macroscopic behaviors of enormous capillary systems can be characterized [33, 34] and scaled up to thousands of cores [32]. This comes at a cost of local accuracy; by simulating the flows directly, we are able to accurately resolve local RBC dynamics that are not captured by such schemes.

Particle-based methods have had the greatest degree of success at large-scale blood flow simulations [13, 16, 39, 40]. These types of approaches are extremely flexible in modeling the fluid and immersed particles, but are computationally demanding and usually suffer from numerical stiffness that requires very small time steps for a given target accuracy. For a comprehensive review, see [55]. There have also been recent advances in coupling a particle-based DPD-like scheme with IB in parallel [54, 56], but the number of RBCs simulated and the complexity of the boundary seems to be limited.

BIE methods have successfully realized large-scale simulations of millions of RBCs [37] in free space. Recently, new methods for robust handling of collisions between RBCs in high-volume fraction simulations have been introduced [25, 28]. This approach is versatile and efficient due to only requiring discretization of RBCs and blood vessel surfaces, while achieving high-order convergence and optimal complexity implementation due to fast summation methods [21, 38, 43, 44, 47, 48, 59]. To solve elliptic partial differential equations, BIE approaches have been successful in several application domains [6, 49, 50, 57]. However, to our knowledge, there has been no work combining a Stokes boundary solver on arbitrary complex geometries in 3D with a collision detection and resolution scheme to simulate RBC flows at large scale. This work aims to fill this gap, illustrating that this can be achieved in a scalable manner.

Related work: collisions. Parallel collision detection methods are a well-studied area in computer graphics for both shared memory and GPU parallelism [20, 23, 29]. [10, 19] detect collisions between rigid bodies in a distributed memory architecture via domain decomposition. [31] constructs a spatial hash to cull collision candidates and explicitly check candidates that hash to the same value. The parallel geometry and physics-based collision resolution scheme detailed in [53] is most similar to the scheme used in this work. However, such discrete collision detection schemes require small time steps to guarantee detections which can become costly for high-volume fraction simulations.

2 FORMULATION AND SOLVER OVERVIEW

2.1 Problem summary

We simulate the flow of N cells with deformable boundary surfaces γ_i, $i = 1, \ldots, N$ in a viscous Newtonian fluid in a domain $\Omega \subset \mathbb{R}^3$ with a fixed boundary Γ. The governing partial differential equations (PDEs) describing the conservation of momentum and mass are the incompressible Stokes equations for the velocity $\boldsymbol{u}$ and pressure p, combined with velocity boundary conditions on Γ. Additionally, we model cell membranes as massless, so the velocity $\mathbf{X}_t$ of the points on the cell surface coincides with the flow velocity:

$$-\mu\Delta\boldsymbol{u}(\boldsymbol{x}) + \nabla p(\boldsymbol{x}) = \mathbf{F}(\boldsymbol{x}) \quad \text{and} \quad \nabla\cdot\boldsymbol{u}(\boldsymbol{x}) = 0, \quad \boldsymbol{x}\in\Omega, \tag{2.1}$$

$$\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{x}), \quad \boldsymbol{x}\in\Gamma, \tag{2.2}$$

$$\mathbf{X}_t = \boldsymbol{u}(\mathbf{X}), \quad \boldsymbol{X}\in\gamma_i(t), \tag{2.3}$$

where μ is the viscosity of the ambient fluid; in our simulations, we use a simplified model with the viscosity of the fluid inside the cells also being μ although our code supports arbitrary viscosity contrast. The right-hand side force in the momentum equation is due to the sum of tension and bending forces $\mathbf{f} = \mathbf{f}_\sigma + \mathbf{f}_b$; it is concentrated on the cell surfaces. We assume that cell surfaces are inextensible, with bending forces determined by the Canham-Helfrich model [8, 18], based on the surface curvature, and surface tension determined by the surface incompressibility condition $\nabla_{\gamma_i}\cdot\boldsymbol{u} = 0$ resulting in

$$\mathbf{F}(\boldsymbol{x}) = \sum_i \int_{\gamma_i} \mathbf{f}(\boldsymbol{y})\delta(\boldsymbol{x}-\boldsymbol{y})d\boldsymbol{y}$$

(see, e.g., [38] for the expressions for $\mathbf{f}$). Except on inflow and outflow regions of the vascular network, the boundary condition $\boldsymbol{g}$ is zero, modeling no-slip boundary condition on blood vessel walls.

2.1.1 Boundary integral formulation. To enforce the boundary conditions on Γ, we use the standard approach of computing $\boldsymbol{u}$ as the sum of the solution $\boldsymbol{u}^{\text{fr}}$ of the free-space equation Eq. (2.1) without boundary conditions but with non-zero right-hand side $\mathbf{F}(\boldsymbol{x})$, and the second term $\boldsymbol{u}^\Gamma$ obtained by solving the homogeneous equation with boundary conditions on Γ given by $\boldsymbol{g} - \boldsymbol{u}^{\text{fr}}$.

Following the approach of [25, 30, 35, 36], we reformulate Eqs. (2.1) and (2.2) in the integral form. The free-space solution $\boldsymbol{u}^{\text{fr}}$ can be written directly as the sum of the single-layer Stokes potentials $\boldsymbol{u}^{\gamma_i}$:

$$\boldsymbol{u}^{\gamma_i}(\boldsymbol{x}) = (S_i\mathbf{f})(\boldsymbol{x}) = \int_{\gamma_i} S(\boldsymbol{x},\boldsymbol{y})\mathbf{f}(\boldsymbol{y})d\boldsymbol{y}, \quad \boldsymbol{x}\in\Omega, \tag{2.4}$$

where $S(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{8\pi\mu}\left(\frac{1}{\boldsymbol{r}} + \frac{\boldsymbol{r}\otimes\boldsymbol{r}}{|\boldsymbol{r}|^3}\right)$ for viscosity μ and $\boldsymbol{r} = \boldsymbol{x}-\boldsymbol{y}$.

To obtain $\boldsymbol{u}^\Gamma$, we reformulate the homogeneous volumetric PDE with nonzero boundary conditions as a boundary integral equation for an unknown double-layer density ϕ defined on the domain boundary Γ:

$$\left(\frac{1}{2}I + D + N\right)\phi = \tilde{D}_\Gamma\phi = \boldsymbol{g} - \boldsymbol{u}^{\text{fr}}, \quad \boldsymbol{x}\in\Gamma, \tag{2.5}$$

where the double-layer operator is $D\phi(\boldsymbol{x}) = \int_\Gamma D(\boldsymbol{x},\boldsymbol{y})\phi(\boldsymbol{y})d\boldsymbol{y}$ with double-layer Stokes kernel $D(\boldsymbol{x},\boldsymbol{y}) = \frac{6}{8\pi}\left(\frac{\boldsymbol{r}\otimes\boldsymbol{r}}{|\boldsymbol{r}|^5}(\boldsymbol{r}\cdot\boldsymbol{n}\right)$ for outward normal $\boldsymbol{n} = \boldsymbol{n}(\boldsymbol{y})$. The null-space operator needed to make the equations full-rank is defined as $(N\phi)(\boldsymbol{x}) = \int_\Gamma(\boldsymbol{n}(\boldsymbol{x})\cdot\phi(\boldsymbol{y}))\boldsymbol{n}(\boldsymbol{y})d\boldsymbol{y}$ (cf. [24]). The favorable eigenspectrum of the integral operator in Eq. (2.5) is well-known and allows GMRES to rapidly converge to a solution. One of the key differences between this work and previous free-space large-scale simulations is the need to solve this equation in a scalable way. Once the density ϕ is computed, the velocity correction $\boldsymbol{u}^\Gamma$ is evaluated directly as $\boldsymbol{u}^\Gamma = D\phi$.

The equation for the total velocity $\boldsymbol{u}(\boldsymbol{x})$ at any point $\boldsymbol{x}\in\Omega$ is then given by

$$\boldsymbol{u} = \boldsymbol{u}^{\text{fr}} + \boldsymbol{u}^\Gamma = \sum_{i=1}^N \boldsymbol{u}^{\gamma_i} + \boldsymbol{u}^\Gamma. \tag{2.6}$$

In particular, this determines the update equation for the boundary points of cells; see Eq. (2.3).

Contact formulation . In theory, the contacts between surfaces are prevented by the increasing fluid forces as surfaces approach each other closely. However, ensuring accuracy of resolving forces may require prohibitively fine sampling of surfaces and very small time steps, making large-scale simulations in space and time impractical. At the same time, as shown in [24], interpenetration of surfaces results in a catastrophic loss of accuracy due to singularities in the integrals.

To guarantee that our discretized cells remain interference-free, we augment Eqs. (2.1) and (2.2) with an explicit inequality constraint preventing collisions. We define a vector function $V(t)$ with components becoming strictly negative if any cell surfaces intersect each other, or intersect with the vessel boundaries Γ. More specifically, we use the *space-time interference volumes* introduced in [17] and applied to 3D cell flows in [25]. Each component of V corresponds to a single connected overlap. The interference-free constraint at time t is then simply $V(t) \geq 0$.

For this constraint to be satisfied, the forces $\mathbf{f}$ are augmented by an artificial collision force, i.e., $\mathbf{f} = \mathbf{f}_b + \mathbf{f}_\sigma + \mathbf{f}_c$, $\mathbf{f}_c = \nabla_u V^T \lambda$, where λ is the vector of Lagrange multipliers, which is determined by the additional *complementarity* conditions:

$$\lambda(t) \geq 0, \quad V(t) \geq 0, \quad \lambda(t) \cdot V(t) = 0, \tag{2.7}$$

at time t, where all inequalities are to be understood componentwise.

To summarize, the system that we solve at every time step can be formulated as follows, where we separate equations for different cells and global and local parts of the right-hand side, as it is important for our time discretization:

$$\boldsymbol{X}_t = \left(\sum_{j \neq i} S_j \mathbf{f}_j + D\phi \right) + S_i \mathbf{f}_i, \quad \text{for points on } \gamma_i, \tag{2.8}$$

$$\nabla_{\gamma_i} \cdot \boldsymbol{X}_t = 0, \quad \mathbf{f}_j = \mathbf{f}(\boldsymbol{X}_j, \sigma_j, \lambda), \tag{2.9}$$

$$B_\Gamma \phi = \boldsymbol{g} - \sum_j S_j \mathbf{f}_j, \quad \text{for points on } \Gamma, \tag{2.10}$$

$$\lambda(t) \geq 0, \quad V(t) \geq 0, \quad \lambda(t) \cdot V(t) = 0. \tag{2.11}$$

At every time step, (2.11) results in coupling of all close γ_i's, which requires a non-local computation. We follow the approach detailed in [24, 25] to define and solve the *nonlinear complementarity problem* (NCP) arising from cell-cell interactions in parallel, and extend it to prevent intersection of cells with the domain boundary Γ, as detailed in Section 4.

2.2 Algorithm Overview

Next, we summarize the algorithmic steps used to solve the constrained integral equations needed to compute cell surface positions and fluid velocities at each time step. In the subsequent sections, we detail the parallel algorithms we developed to obtain good weak and strong scalability, as shown in Section 5.

Overall Discretization. RBC surfaces are discretized using a spherical harmonic representation, with surfaces sampled uniformly in the standard latitude-longitude sphere parametrization. The blood vessel surfaces Γ are discretized using a collection of high-order tensor-product polynomial patches, each sampled at Clenshaw-Curtis quadrature points. The space-time interference volume function $V(t)$ is computed using a piecewise-linear approximation as described in [25]. For time discretization, we use a locally-implicit first order time-stepping (higher-order time stepping can be easily incorporated). Interactions between RBCs and the blood vessel surfaces are computed *explicitly*, while the self-interaction of a single RBC is computed *implicitly*.

The state of the system at every time step is given by a triple of distributed vectors $(\boldsymbol{X}, \sigma, \lambda)$. The first two (cell surface positions and tensions) are defined at the discretization points of cells. The vector λ has variable length and corresponds to connected components of collision volumes. We use the subscript i to denote the subvectors corresponding to i-the cell. $\boldsymbol{X}$ and σ are solved as a single system that includes the incompressibility constraint Eq. (2.9). To simplify exposition, we omit σ in our algorithm summary, which corresponds to dropping $\mathbf{f}_\sigma$ in the Stokes equation, and dropping the surface incompressibility constraint equation.

Algorithm summary. At each step t, we compute the new positions $\boldsymbol{X}_i^+$ and collision Lagrange multipliers λ^+ at time $t^+ = t + \Delta t$. We assume that in the initial configuration there are no collisions, so the Lagrange multiplier vector λ is zero. Discretizing in time, Eq. (2.8) becomes

$$\boldsymbol{X}_i^+ = \boldsymbol{X}_i + \Delta t \left(\sum_{j \neq i} S_j \mathbf{f}_j(\boldsymbol{X}_j, \lambda) + D\phi(\boldsymbol{X}_j, \lambda) \right) + \Delta t S_i \mathbf{f}_i(\boldsymbol{X}_i^+, \lambda^+).$$

At each single time step, we perform the following steps to obtain $(\boldsymbol{X}^+, \lambda^+)$ from $(\boldsymbol{X}, \lambda)$. Below evaluation of integrals implies using appropriate (smooth, near-singular or singular) quadrature rules on cell or blood vessel surfaces.

(1) Compute the explicit part $\boldsymbol{b}$ of the position update (first term in Eq. (2.8)).
   (a) Evaluate $\boldsymbol{u}^{\text{fr}}$ from $(\boldsymbol{X}, \lambda)$ on Γ with Eq. (2.4).
   (b) Solve Eq. (2.5) for the unknown density ϕ on Γ using GMRES.
   (c) For each cell, evaluate $\boldsymbol{u}_i^\Gamma = D\phi$ at all cell points $\boldsymbol{X}_i$.
   (d) For each cell i, compute the contributions of other cells to $\boldsymbol{X}_i^+$: $\boldsymbol{b}_i^c = \boldsymbol{u}^{\text{fr}} - u^{\gamma_i} = \sum_{j \neq i} S_j \mathbf{f}_j$.
   (e) Set $\boldsymbol{b}_i = \boldsymbol{u}_i^\Gamma + \boldsymbol{b}_i^c$.
(2) Perform the implicit part of the update: solve the NCP obtained by treating the second (self-interaction) term in Eq. (2.8) while enforcing the complementarity constraints Eq. (2.7), i.e., solve

$$\boldsymbol{X}_i^+ = \boldsymbol{X}_i + \Delta t (\boldsymbol{b}_i + S_i \boldsymbol{f}_i(\boldsymbol{X}_i^+, \lambda^+)), \tag{2.12}$$

$$\lambda(t^+) \geq 0, \quad V(t^+) \geq 0, \quad \lambda(t^+) \cdot V(t^+) = 0. \tag{2.13}$$

Items 1a to 1d all require evaluation of global integrals, evaluated as sums over quadrature points; we compute these sums in parallel with PVFMM. In particular, Item 1b uses PVFMM as a part of each matrix-vector product in the GMRES iteration. These matrix-vector product, as well as Items 1a, 1c and 1d require near-singular integration to compute the velocity accurately near RBC and blood vessel surfaces; this requires parallel communication to find non-local evaluation points. Details of these computations are discussed in Section 3.

The NCP is solved using a sequence of *linear complementarity problems* (LCPs). Algorithmically, this requires parallel searches of collision candidate pairs and the repeated application of the distributed LCP matrix to distributed vectors. Details of these computations are provided in Section 4.

Other parallel quadrature methods. Various other parallel algorithms are leveraged to perform boundary integrals for the vessel geometry and RBCs. To compute $\boldsymbol{u}^{\gamma_i}(\boldsymbol{X})$ for $\boldsymbol{X} \in \gamma_i$, the schemes presented in [48] are used to achieve spectral convergence for single-layer potentials by performing a spherical harmonic rotation and apply the quadrature rule of [14]. We use the improved algorithm in [28] to precompute the singular integration operator and substantially improve overall complexity. To compute $\boldsymbol{u}^{\gamma_i}(\boldsymbol{X})$ for $\boldsymbol{X}$ close to, but not on γ_i, we follow the approaches of [28, 43], which use a variation of the high-order near-singular evaluation scheme of [58]. Rather than extrapolating the velocity from nearby check points as in Section 3, we use [48] to compute the velocity on surface, upsampled quadrature on γ_i to compute the velocity at check points and interpolate the velocity between them to the desired location. We mention these schemes for the sake of completeness; they are not the primary contribution of this work, but are critical components of the overall simulation.

3 BOUNDARY SOLVER

The main challenge in incorporating prescribed flow boundary conditions $\boldsymbol{g}$ on the domain boundary Γ is the approximation and solution of the boundary integral problem Eq. (2.5). Upon spatial discretization, this is an extremely large, dense linear system that must be solved at every time step due to the changing free space solution $\boldsymbol{u}^{\text{fr}}$ on the right hand side. Since we aim at a scalable implementation, we do not assemble the operator on the left hand side but only implement the corresponding matrix-vector multiplication, i.e., its application to vectors. Combined with an iterative solver such as GMRES, this matrix-vector multiply is sufficient to solve Eq. (2.5). Application of the double-layer operator D to vectors amounts to a near-singular quadrature for points close to Γ. Controlling the error in this computation requires a tailored quadrature scheme. This scheme is detailed below, where we put a particular emphasis on the challenges due to our parallel implementation.

3.1 Quadrature for integral equation

The domain boundary Γ is given by a collection of non-overlapping patches $\Gamma = \bigcup_i P_i(Q)$, where $P_i : Q \to \mathbb{R}^3$ is defined on $Q = [-1, 1]^2$. We use the Nyström discretization for Eq. (2.5). Since $D(\boldsymbol{x}, \boldsymbol{y})$ is singular, this requires a singular quadrature scheme for the integral on the right-hand side. We proceed in several steps, starting with the direct non-singular discretization, followed by a distinct discretization for the singular and near-singular case.

Non-singular integral quadrature. We discretize the integral in Eq. (2.5), for $\boldsymbol{x} \notin \Gamma$, by rewriting it as an integral over a set of patches and then apply a tensor-product qth order Clenshaw-Curtis rule to each patch:

$$\boldsymbol{u}(\boldsymbol{x}) = \sum_i \int_{P_i} D(\boldsymbol{x}, \boldsymbol{y})\phi(\boldsymbol{y}) d\boldsymbol{y}_{P_i} \approx \sum_i \sum_{j=0}^{q^2} D(\boldsymbol{x}, \boldsymbol{y}_{ij}) w_{ij} \phi(\boldsymbol{y}_{ij}), \quad (3.1)$$

where $\boldsymbol{y}_{ij} = P_i(\boldsymbol{t}_j)$ and $\boldsymbol{t}_j \in [-1, 1]^2$ is the jth quadrature point and w_{ij} is the corresponding quadrature weight. We refer to the points $\boldsymbol{y}_{ij}$ as the *coarse discretization of* Γ and introduce a single global index $\boldsymbol{y}_\ell = \boldsymbol{y}_{ij}$ with $\ell = \ell(i, j) = (i-1)q^2 + j$, $\ell = 1, \ldots, N$, where N is the total number of quadrature nodes. We can then rewrite the right-hand side of (3.1) compactly as the vector dot product $W(\boldsymbol{x}) \cdot \phi$, where $\phi_\ell = \phi(\boldsymbol{y}_\ell)$ and $W_\ell(\boldsymbol{x}) = D(\boldsymbol{x}, \boldsymbol{y}_\ell) w_\ell$ are the quadrature weights in Eq. (3.1).

As $\boldsymbol{x} \to \Gamma$ for $\boldsymbol{x} \in \Omega$, the integrand becomes more singular and the accuracy of this quadrature rapidly decreases due to the singularity in the kernel D. This requires us to construct a singular integral discretization for $\boldsymbol{x} = \boldsymbol{y}_\ell$, $\ell = 1, \ldots, N$, and general points on Γ, which is discussed next. Note that the same method is used for evaluation of the velocity values at points close to the surface, once the equation is solved (*near-singular integration*).

Singular and near-singular integral quadrature. We take

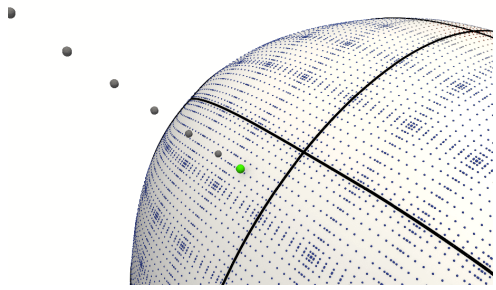


Figure 2: *Schematic of our unified singular/near-singular quadrature scheme. A boundary* Γ *is shown along with a set of patches (patch edges shown in black). We evaluate the velocity due to* Γ *at the check points (gray dots off-surface) using the fine discretization (small dots on-surface) and extrapolate these values to the target point (green). The target point may be on or near* Γ*. The fine discretization subdivides the patches in the coarse discretization into* 16 *patches, each with an* 11*th-order tensor-product Clenshaw-Curtis quadrature rule.*

an approach similar to [22]. The idea is to evaluate the integral sufficiently far from the surface using the non-singular quadrature rule (3.1) on an upsampled mesh, and then to extrapolate the accurate values towards the surface. Concretely, to compute the singular integral at a point $\boldsymbol{x}$ near or on Γ, we use the following steps:

(1) Upsample ϕ using qth order interpolation, i.e., $\phi^{\text{up}} = U\phi$, where ϕ^{up} is the vector of Nk samples of the density and U is the interpolation operator. To be precise, we subdivide each patch P_i into k square subdomains P_{ik} and use Clenshaw-Curtis nodes in each subdomain. We subdivide uniformly, i.e., P_i is split into $k = 4^\eta$ patches for an integer η. This is the *fine discretization of* Γ. We use W^{up} to denote the weights for Eq. (3.1) the fine discretization quadrature points.
(2) Find the closest point $\boldsymbol{y} = P(u^*, v^*)$ to $\boldsymbol{x}$ on Γ for some patch P on Γ with $u^*, v^* \in [-1, 1]$ ($\boldsymbol{y} = \boldsymbol{x}$ if $\boldsymbol{x} \in \Gamma$).
(3) Construct *check points* $c_q = c_q(\boldsymbol{x}) = \boldsymbol{y} - (R + ir)\boldsymbol{n}(u^*, v^*)$, $i = 0, \ldots, p$, where $\boldsymbol{n}(u, v)$ is the outward normal vector to Γ at $P(u, v)$.
(4) Evaluate the velocity at the check points:

$$\boldsymbol{u}(c_q(\boldsymbol{x})) \approx W^{\text{up}}(c_q) \cdot \phi^{\text{up}}, \quad i = 0, \ldots, p. \quad (3.2)$$

(5) Extrapolate the velocity from the check points to $\boldsymbol{x}$ with 1D polynomial extrapolation:

$$\boldsymbol{u}(\boldsymbol{x}) \approx \sum_q e_q \boldsymbol{u}(c_q(\boldsymbol{x})) = \left(\sum_q e_q W^{\text{up}}(c_q)\right) U\phi \tag{3.3}$$

$$= W^{\text{s}}(\boldsymbol{x}) \cdot \phi, \tag{3.4}$$

where e_q are the extrapolation weights.

In this work, the parameters R, p, r and η are chosen empirically to balance the error in the accuracy of $W^{\text{up}}(c_q) \cdot \phi^{\text{up}}$ and the extrapolation to $\boldsymbol{x}$. A schematic of this quadrature procedure is shown in Fig. 2.

Discretizing the integral equation. With the singular integration method described above, we take $\boldsymbol{x} = \boldsymbol{y}_\ell$, $\ell = 1 \ldots N$, and obtain the following discretization of Eq. (2.5):

$$\left(\frac{1}{2}I + A\right)\phi = \boldsymbol{g}, \quad A_{\ell m} = W^s_m(\boldsymbol{y}_\ell) + N_{ij}, \tag{3.5}$$

where $\boldsymbol{g}$ is the boundary condition evaluated at $\boldsymbol{y}_\ell$, $W^s_m(\boldsymbol{x})$ is the mth component of $W^s(\boldsymbol{x})$ and N_{ij} is the appropriate element of the rank-completing operator in Eq. (2.5).

The dense operator A is never assembled explicitly. We use GMRES to solve Eq. (3.5), which only requires application of A to vectors ϕ. This matrix-vector product is computed using the steps summarized above.

Extrapolation and upsampling are local computations that are parallelized trivially if all degrees of freedom for each patch are on a single processor. The main challenges in parallelization of the above singular evaluation are 1) initially distributing the patches among processors, 2) computing the closest point on Γ and 3) evaluating the velocity at the check points. The parallelization of these computations is detailed in the remainder of this section.

Far evaluation. To compute the fluid velocity away from Γ, where Eq. (2.5) is non-singular, i.e., at the check points, the integral can be directly evaluated using Eq. (3.1). Observing that Eq. (3.1) has the form of an N-body summation, we use the *fast-multipole method* [15] to evaluate it for all target points at once. We use the parallel, kernel-independent implementation Parallel Volume Fast Multipole Method (PVFMM) [26, 27], which has been demonstrated to scale to hundreds of thousands of cores. PVFMM handles all of the parallel communication required to aggregate and distribute the contribution of non-local patches in $O(N)$ time.

3.2 Distributing geometry and evaluation parallelization

We load pieces of the blood vessel geometry, which is provided as a quad mesh, separately on different processors. Each face of the mesh has a corresponding polynomial P_i defining the ith patch.

The k levels of patch subdivision induce a uniform quadtree structure within each quad. We use the `p4est` library [7] to manage this surface mesh hierarchy, keep track of neighbor information, distribute patch data and to refine and coarsen the discretization in parallel. The parallel quadtree algorithms provided by `p4est` are used to distribute the geometry without replicating the complete surface and polynomial patches across all processors. `p4est` also determines parent-child patch relationships between the coarse and fine discretizations and the coordinates of the child patches to which we interpolate.

Once the geometry is distributed, constructing check points, all necessary information for upsampling and extrapolation are either available on each processor or communicated by PVFMM. This allows these operations to be embarassingly parallel.

3.3 Parallel closest point search

To evaluate the solution at a point $\boldsymbol{x}$, we must find the closest point $\boldsymbol{y}$ on the boundary to $\boldsymbol{x}$. The distance $\|\boldsymbol{x} - \boldsymbol{y}\|_2$ determines whether or not near-singular integration is required to compute the velocity at $\boldsymbol{x}$. If it is, $\boldsymbol{y}$ is used to construct check points.

In the context of this work, the point $\boldsymbol{x}$ is on the surface of an RBC, which may be on a different processor than the patch containing $\boldsymbol{y}$. This necessitates a parallel algorithm to search for $\boldsymbol{y}$. For that purpose, we extend the spatial sorting algorithm from [25, Algorithm 1] to support our fixed patch-based boundary and detect near pairs of target points and patches.

a. *Construct a bounding box $B_{P,\epsilon}$ for the near-zone of each patch.* We choose a distance d_ϵ so that for all points $\boldsymbol{z}$ further away than d_ϵ from P, the quadrature error of integration over P is bounded by ϵ. The set of points closer to P than d_ϵ is the *near-zone of P*. We inflate the bounding box B_P of P by d_ϵ along the diagonal to obtain $B_{P,\epsilon}$ to contain all such points.
b. *Sample $B_{P,\epsilon}$ and compute a spatial hash of the samples and $\boldsymbol{x}$.* Let H be the average diagonal length of all $B_{P,\epsilon}$. We sample the volume contained in $B_{P,\epsilon}$ with equispaced samples of spacing $h_P < H$. Using a spatial hash function, (such as Morton ordering with a spatial grid of spacing H), we assign hash values to bounding box samples and $\boldsymbol{x}$ to be used as a sorting key. This results in a set of hash values that define the near-zone of Γ.
c. *Sort all samples by the sorting key.* Use the parallel sort of [45] on the sorting key of bounding box samples and that of $\boldsymbol{x}$. This collects all points with identical sorting key (i.e., close positions) and places them on the same processor. If the hash of $\boldsymbol{x}$ matches the hash of a bounding box sample, then $\boldsymbol{x}$ could require near-singular integration, which we check explicitly. Otherwise, we can assume $\boldsymbol{x}$ is sufficiently far from P and does not require singular integration.
d. *Compute distances $\|\boldsymbol{x} - P_i\|$.* For each patch P_i with a bounding box key of $\boldsymbol{x}$, we locally solve the minimization problem $\min_{(u,v)\in[-1,1]^2} \|\boldsymbol{x} - P_i(u,v)\|$ via Newton's method with a backtracking line search. This is a local computation since $\boldsymbol{x}$ and P_i were communicated during the Morton ID sort.
e. *Choose the closest patch P_i.* We perform a global reduce on the distances $\|\boldsymbol{x} - P_i\|$ to determine the closest P_i to $\boldsymbol{x}$ and communicate back all the relevant information required for singular evaluation back to $\boldsymbol{x}$'s processor.

4 PARALLEL COLLISION HANDLING

We prevent collisions of RBCs with other RBCs and with the vessel surface Γ by solving the NCP given in Eqs. (4.1) and (4.2). This is a nonsmooth and non-local problem, whose assembly and efficient solution is particularly challenging in parallel.In this section, we summarize our constraint-based approach and algorithm.

We have integrated piecewise polynomial patches into the framework of [25] for parallel collision handling, to which we refer the reader for a more detailed discussion. The key step to algorithmically unify RBCs and patches is to *form a linear triangle mesh approximation* of both objects. We now want to enforce that these meshes are collision-free subject to the physics constraints in Eq. (4.1).

We linearize the NCP and solve a sequence of LCPs whose solutions converge to the NCP solution. At a high-level, the collision algorithm proceeds as follows:

(1) Find triangle-vertex pairs of distinct meshes that are candidates for collision.
(2) Compute $V(t^+) = V(t^{+,0})$. If any triangle-vertex pairs on distinct meshes collide, the corresponding component of $V(t)$ will be negative.
(3) While $V_i(t^{+,k}) < 0$ for any i:
   (a) Suppose m components of $V(t)$ are negative
   (b) Solve the following linearized version of Eqs. (4.1) and (4.2)

$$\boldsymbol{X}_i^{+,k} = \boldsymbol{X}_i + \Delta t(\boldsymbol{b}_i + S_i(\mathbf{f}_i(\boldsymbol{X}_i^{+,k}, \lambda^{+,k})), \tag{4.1}$$

$$\lambda(t^+) \geq 0, \quad L(t^{+,k}) \geq 0, \quad \lambda(t^{+,k}) \cdot L(t^{+,k}) = 0, \tag{4.2}$$

$$\text{where} \quad L(t) = V(t) + \nabla_u V^T \Delta \boldsymbol{X}_i(t) \tag{4.3}$$

   for the kth iteration of the loop and $\boldsymbol{X}_i^{+,k} = \boldsymbol{X}_i + \Delta \boldsymbol{X}_i(t^{+,k})$.
   (c) Find new candidate triangle-vertex pairs and compute $V(t^{+,k})$.

Here, $t^{+,k}$ is the intermediate time step at which a new candidate position $\boldsymbol{X}_i^{+,k}$ occurs. This approach of iteratively solving an NCP with sequence of LCPs was shown to converge superlinearly in [11]. In [53], the authors demonstrate that one LCP linearization can approximate the NCP accurately; our algorithm uses around seven LCP solves to approximately solve the NCP. Upon convergence of this algorithm, we are guaranteed that our system is collision-free.

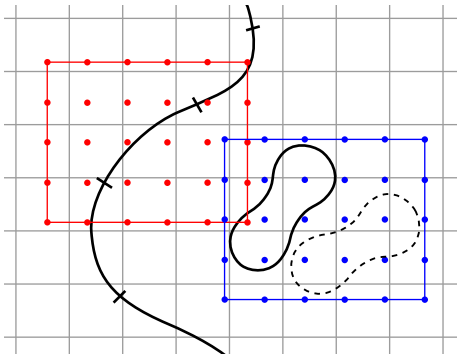


Figure 3: *A 2D depiction of the parallel candidate collision pair algorithm. Shown is the implicit spatial grid (gray), a piece of the blood vessel Γ (open black curve), an RBC γ_i at the current time step (closed black curve) and at the next time step (dotted closed back curve). Also shown is the space-time bounding box and bounding box samples of a single patch (red square and red dots) and an RBC (blue square and blue dots).*

To solve the LCP in Item 3b, we follow the approach detailed in [24, Section 3.2.2, Section 3.3]. We reformulate the problem first in standard LCP form with diagonally-dominant system matrix $\boldsymbol{B}$, then solve an equivalent root finding problem by applying a minimum-map Newton's method. This can be restructured to use GMRES, so we only need to repeatedly apply $\boldsymbol{B}$ to vectors to solve the LCP. Each entry $\boldsymbol{B}_{ij}$ is the change in the jth contact volume induced by the kth contact force, which is explicitly defined in [25, Algorithm 3]. This means that $\boldsymbol{B}$ is of size $m \times m$, where m is the number of collisions, but is extremely sparse. We need not store the entire matrix explicitly; we only compute the non-zero entries and store them in a distributed hash-map. Computing these matrix elements requires an accumulation of all coupled collision contributions to the velocity, which requires just a sparse `MPI_All_to_Allv` to send each local contribution to the process containing $V_i(t^{+,k})$.

An important step to ensure good scaling of our collision handling algorithm is to minimize the number of triangle-vertex pairs that are found in Item 1. One could explicitly compute an all-to-all collision detection on all meshes in the system, but this requires $O(N^2)$ work and global communication. We perform a high-level filtering first to find local *candidate collision mesh pairs*, then only communicate and compute the required $O(m)$ information. Since spatially-near mesh pairs may be on different processors, we need a parallel algorithm to compute these collision candidates.

To address this, we reuse Items a to c from Section 3.3 and adapt it to this problem. For each mesh in the system, we form the *space-time bounding box* of the mesh: the smallest axis-aligned bounding box containing the mesh at positions $\boldsymbol{X}_i$ and $\boldsymbol{X}_i^+$, as shown in Fig. 3. For patches P_i, note that $P_i^+ = P_i$. This means one can reuse the bounding box of P_i constructed in Section 3.3 for this purpose and simply set d_ϵ to zero. After forming all space-time bounding boxes for the meshes of all patches and RBCs, we apply steps Items b and c directly to these boxes. Item c will communicate meshes with the same spatial sorting key to the same processor; these meshes are collision candidate pairs. Once the computation is local and candidate collision pairs are identified, we can proceed with the NCP solution algorithm described above.

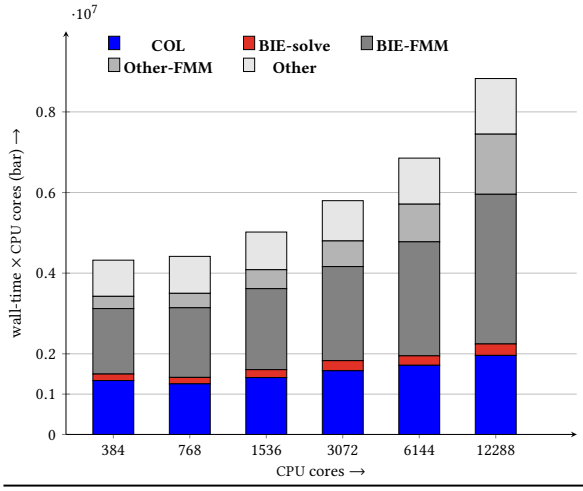
5 RESULTS

In this section, we present scalability results for our blood flow simulation framework on various test geometries, simulations with various volume fractions and demonstrate the convergence behavior of our numerical methods.

5.1 Implementation and example setup

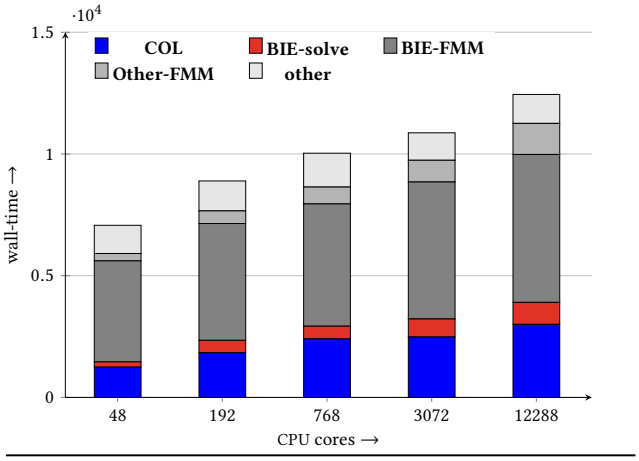
Architecture and software libraries. We use the Stampede2 system at the Texas Advanced Computing Center (TACC) to study the scalability of our algorithms and implementation. Stampede2 has two types of compute nodes, the Knights Landing (KNL) compute nodes and the Skylake (SKX) compute nodes. The SKX cluster has 1,736 dual-socket compute nodes, each with two 24-core 2.1GHz CPUs and 192GB of memory. The KNL cluster has 4,200 compute nodes, with a 68-core Intel Xeon Phi 7250 1.4Ghz CPUs and 96GB of memory plus 16GB of high-speed MCDRAM. We run our simulations in a hybrid distributed-shared memory fashion: we run one MPI process per node, with one OpenMP thread per hardware core. Our largest simulations use 256 SKX and 512 KNL nodes.

We leverage several high-performance libraries in our implementation. We use PETSc's [2] parallel matrix and vector operations, and



cores	384	768	1536	3072	6144	12288
total time (sec)	11257	5751	3268	1887	1116	718
efficiency	1.00	0.98	0.86	0.75	0.63	0.49
COL+BIE-solve (sec)	3901	1843	1046	596	317	183
efficiency	1.00	1.05	0.93	0.82	0.77	0.66

Figure 4: *Strong scalability of a simultion with* 40960 *RBCs on Stampede's SKX partition for the vessel network geometry shown in Fig. 1. The vessel is discretized with* 40960 *polynomial patches. Shown in the bar graph is a breakdown of the compute resources (wall-time × CPU cores) required by the individual components for a simulation with 10 time steps on 384 to 12288 cores. The compute resources used by the main algorithms presented in this paper are* ***COL*** *(collision handling),* ***BIE-solve*** *(computation of* $\boldsymbol{u}^{\Gamma}$*, not including FMM calls). Shown in different gray scales are the compute resources required by FMM (****BIE-FMM*** *and* ***Other-FMM****) and other operations (****Other****). Shown in the table are the compute time and the parallel efficiency for the overall computation and for the sum of* ***COL*** *and* ***BIE-solve****. For the collision avoidance and the boundary solve we observe a parallel efficiency of* 66% *for a 32-fold increase from 384 to 12288 CPU cores.*



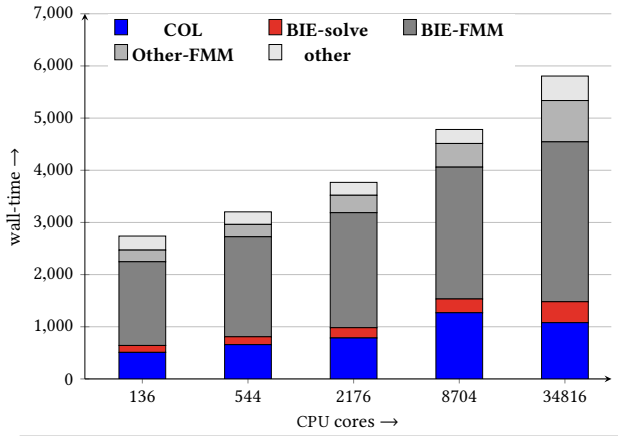
cores	48	192	768	3072	12288
vol fraction	19%	20%	23%	26%	27%
#collision/ #RBCs	15%	13%	17%	15%	16%
total time (sec)	7070	8892	10032	10869	12446
efficiency	–	1.00	0.88	0.81	0.71
COL+BIE-solve (sec)	1461	2345	2926	3222	3904
efficiency	–	1.00	0.80	0.73	0.60

Figure 5: *Weak scalability on Stampede's SKX partition with node grain size of* 4096 *RBCs and* 8192 *polynomial patches per compute node (each node has 48 cores) for the vessel geometry shown in Fig. 8. Increasing the number of RBCs and boundary patches is realized by decreasing the size of the RBCs as discussed in Section 5.2. Shown in the bar graph is a breakdown of wall-time spent in individual components for a simulation with 10 time steps on 136 to 12288 cores (i.e., 4 to 256 nodes). The explanation of the labels used in the legend is detailed in Fig. 4. Additionally, we show the volume fraction of RBCs for each simulation, as well as the percentage of vesicles where the RBC-RBC or RBC-vessel collision prevention is active. We report the parallel scalability with respect to 192 cores, as the smallest simulation is in a single node and no MPI communication is necessary. The largest simulation has 1,048,576 RBCs and 2,097,152 polynomial patches and an overall number of 3,042,967,552 unknowns per time step.*

its parallel GMRES solver. Management and distribution of patches describing the blood vessel geometry uses the `p4est` library [7], and we use PVFMM [26] for parallel FMM evaluation. We also heavily leverage `Intel MKL` for fast dense linear algebra routines at the core of our algorithms and `paraview` for our visualizations.

Discretization and example setup. For all test cases we present, we discretize each RBC with 544 quadrature points and 2,112 points for collision detection. The blood vessel geometry is represented with 8th order tensor-product polynomial patches with 121 quadrature points per patch and 484 equispaced points for collision detection. The parameters chosen for singular/near-singular integration are $p = 8$ and $\eta = 1$, with $R = r = .15L$ for strong scaling tests and $R = r = .1L$ for weak scaling tests. The value of L is the square root of the surface area of the patch containing the closest point to the target, called the *patch size*; this choice allows for a consistent extrapolation error over the entirety of Γ.

Since our scaling tests are performed on complex, realistic blood vessel geometries, we must algorithmically generate our initial simulation configuration. We prescribe portions of the blood vessel as inflow and outflow regions and appropriately prescribe positive and negative parabolic flows (inlet and outlet flow) as boundary conditions, such that the total fluid flux is zero. To populate the blood vessel with RBCs, we uniformly sample the volume of the bounding box of the vessel with a spacing h to find point locations inside the domain at which we place RBCs in a random orientation. We then slowly increase the size of each RBC until it collides with the vessel boundary or another RBC; this determines a single RBC's size. We continue this process until all RBCs stop expanding; this means that we are running a simulation of RBCs of various sizes. We refer to this process as *filling the blood vessel with* RBCs. This typically produces RBCs of radius r with $r_0 < r < 2r_0$ with r_0 chosen proportional to h. This is a precomputation for our simulation, so we do not include this step in the timings we report for weak and strong scaling. We emphasize that these simulations are primarily for scaling purposes of our algorithms and are not expected to



cores	136	544	2176	8704	34816
vol fraction	17%	19%	20%	23%	26%
#collision/ #RBCs	10%	15%	13%	17%	15%
total time (sec)	2739	3203	3768	4782	5806
efficiency	1.00	0.86	0.73	0.57	0.47
COL+BIE-solve (sec)	642	808	982	1532	1480
efficiency	1.00	0.79	0.65	0.42	0.43

Figure 6: *Same as Fig. 5 but on Stampede2's KNL partition with* 512 *RBCs and* 1024 *vessel boundary patches per node (each node has 68 cores). We find an overall parallel scalability of 47% for a 256-fold increase of the problem size.*

Figure 7: *Shown is a high-volume fraction sedimentation due to gravitational force. The initial configuration (top figures) has a volume fraction of* 47%. *As the cells sediment to the lower part of the domain (bottom figures), the local volume fraction of the final state in this lower part of the domain is around* 55%. *Shown on the right side are slices through the center of the domain together with the RBC boundaries in the initial and final configuration. The full simulation video is available at https://vimeo.com/329509435.*

represent true blood flows. The platform can of course be applied to length scales where viscous flow is a valid assumption.

Additionally, RBCs in such a confined flow will collide with the blood vessel wall if special care is not taken near the outflow part of the boundary. We define regions near the inlet and outlet flows where we can safely add and remove RBCs. When an RBC γ_i is within the outlet region, we subtract off the velocity due to γ_i from the entire system and move γ_i into an inlet region such that the arising RBC configuration is collision-free.

Limiting GMRES iterations. We have observed that the GMRES solver typically requires 30 iterations or less for convergence for almost all time steps, but the number of needed iterations may vary more in the first steps. To simulate the amount of work in a typical simulation time step, we cap the number of GMRES iterations at 30 and report weak and strong scaling for these iterations. A more detailed analysis of this behavior is needed.

5.2 Parallel scalability

Here, we present strong and weak scalability results for our RBC simulations. We decompose the time required for a complete simulation into the following categories:

- ***COL***: detection and resolution of collisions among RBCs and between RBCs and the vessel walls;
- ***BIE-solve***: computing $\boldsymbol{u}^\Gamma$, not including FMM calls. This includes all of the steps for singular/near-singular integration in Section 3 except the evaluation $\boldsymbol{u}^\Gamma$ at the check points.
- ***BIE-FMM***: FMM calls required to evaluate $\boldsymbol{u}^\Gamma$ at the check points and at points on RBCs
- ***Other-FMM***: FMM calls required by other algorithms
- ***Other***: all other operations

In the discussion below, we focus on ***COL*** and ***BIE-solve***, as they are the primary algorithmic contribution of this work, and discuss how to reduce the computational time required for ***BIE-FMM***.

Strong scalability. To study the strong scalability of our algorithms, we use the blood vessel geometry and RBC configuration in Fig. 1-left. This simulation contains 40,960 RBCs and the blood vessel is represented with 40,960 patches. With four degrees of freedom per RBC quadrature point and three per vessel quadrature point, this amounts to 89,128,960 and 14,868,480 degrees for the RBCs and blood vessel, respectively (103,997,440 in total). As can be seen from Fig. 4, we achieve a 15.7-fold speed-up in total wall-time scaling from 384 to 12288 cores, corresponding to 49% parallel efficiency. This level of parallel efficiency is partially due to the calls to the fmm library PVFMM. The strong scalability of PVFMM we observe is largely consistent with the results reported in [27]. Neglecting the time for calls to FMM, i.e., only counting the time for the boundary solver to compute $\boldsymbol{u}^\Gamma$ and for collision prevention, we find 66% parallel efficiency when scaling strongly from 384 to 12288 cores. We see that the parallel collision handling and integral

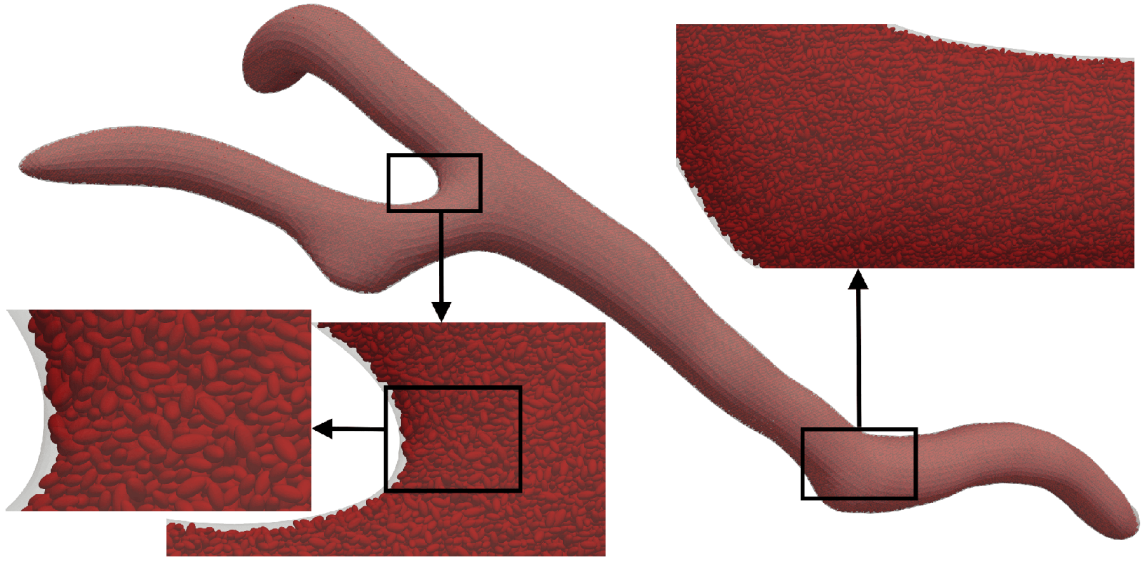


Figure 8: *For our weak scaling experiments, we use the the vessel geometry shown above with inflow boundary conditions on the right side and outflow boundary condition on the two left sides. To setup the problem, we fill the vessel with nearly-touching RBCs of different sizes to obtain a desired number, and refine the vessel geometry patches. The figure above shows a setup with overall 262,144 RBCs at a volume fraction of 26%.*

equation solver computations, excluding FMM, scale well as the number of cores is increased.

Weak Scalability. Our weak scalability results are shown in Figs. 5 and 6. Both tests are performed on the blood vessel displayed in Fig. 8. We use an initial boundary composed of a fixed number M of polynomial patches and fill the domain with roughly $M/2$ RBCs (which requires spacing h). To scale up our simulation by a factor of four, we: (1) subdivide the M polynomial patches into $4M$ new but equivalent polynomial patches (via subdivision rules for Bezier curves); (2) refill the domain with RBCs using spacing $h/\sqrt[3]{4}$. This places $2M$ RBCs in the domain volume. We repeat this process each time we increase the number of cores by a factor of four in order to keep the number of patches and RBCs per core constant. In the tables in Figs. 5 and 6, we report parallel efficiency with respect to the first multi-node run on both SKX and KNL architectures, i.e, with respect to 192 and 136 cores, respectively.

The largest weak scaling test contains 1,048,576 RBCs and 2,097,152 polynomial patches on the blood vessel; we solve for 3,042,967,552 unknowns at each time step and are able to maintain a collision-free state between 4,194,304,000 triangular surface elements at each time step. Comparing the weak scalability results for SKX (Fig. 5) and KNL (Fig. 6), we observe similar qualitative behavior. Note that the smallest test on the SKX architecture only uses a single node, i.e., no MPI communication was needed. This explains the increased time for the collision prevention algorithms when going from 1 (48 cores) to 4 nodes (192 cores). Note also that the simulation on the KNL architecture used a significantly lower number of RBCs and geometry patches per node. Thus, this simulation has a larger ratio of communication to local work. This explains the less perfect scalability compared to the results obtained on the SKX architecture. As with strong scaling, we see good parallel scaling of the non-FMM-related parts of the computation of $\boldsymbol{u}^{\Gamma}$ and the collision handling algorithm.

Note that there is a slight variation in the number of collisions for the run on 8704 cores on KNL. This is an artifact of the RBC filling algorithm. Since we place RBCs in random orientations and distribute RBCs randomly among processors, we do not have complete control over the percentage of collisions or the volume fraction for each simulation in Figs. 5 and 6, as can be seen from the tables under these figures. This can affect the overall scaling: For the run on 8704 cores, the percentage of collisions is larger, explaining the longer time spent in ***COL***. Despite this phenomenon, we achieve good weak scaling overall.

Discussion. The parts of the algorithm introduced in this paper scale as well as or better than the FMM implementation we are using. However, our overall run time is diminished by the multiple expensive FMM evaluations required for solving Eq. (2.5). This can be addressed by using a *local* singular quadrature scheme, i.e., compute a singular integral using the FMM on Eq. (3.1) directly, then compute a singular correction locally. This calculation has a three-fold impact on parallel scalability: (1) the FMM evaluation required is proportional to the size of the coarse discretization rather than the fine discretization ($O((p+1)N)$ vs. $O((k+p)N)$); (2) after the FMM evaluation, the local correction is embarrassingly parallel; (3) the linear operator Eq. (3.3) can be precomputed, making the entire calculation extremely fast with MKL linear algebra routines. These improvements together will allow our algorithm to scale well beyond the computational regime explored in this work.

5.3 Verification

There are few analytic results known about RBCs in confined Stokes flows against which we can verify our simulations. However, exact solutions can be obtained for a part of our setup, invariants (e.g., surface area) can be considered and solutions for smaller examples can be verified against solutions with fine spatial and temporal discretizations. In particular, in this section, we demonstrate the accuracy of the parallel boundary solver presented in Section 3 and numerically study the collision-free time-stepping in Section 4.

Boundary solver. The error of the boundary integral solver is determined by the error of integration and the GMRES error, the latter not depending on the number of discretization points due to good conditioning of the equation. The integration error, in turn, can be separated into smooth quadrature error and interpolation error. The former is high-order accurate [46]. Although our extrapolation is ill-conditioned, we observe good accuracy for $p \leq 8$. The singular evaluation in Section 3 converges with rate $O(L^p + L^q)$ corresponding to pth order extrapolation and qth order quadrature. To confirm this numerically, we solve an interior Stokes problem on the surface in Fig. 9-right. We evaluate a prescribed analytic solution at the discretization points to obtain the boundary condition. We then solve Eq. (3.5) and compare the numerical solution at on-surface samples different from discretization points, evaluated using the algorithms of Section 3. We use $\eta = 2$, $q = 16$, $p = 8$, $R = .04\sqrt{L}$ and $r = R/8$. In Fig. 9-left, we report the relative error in the infinity-norm of the velocity. By choosing check point distances proportional to $\sqrt{L}$, we observe the expected $O(L^7)$ convergence.

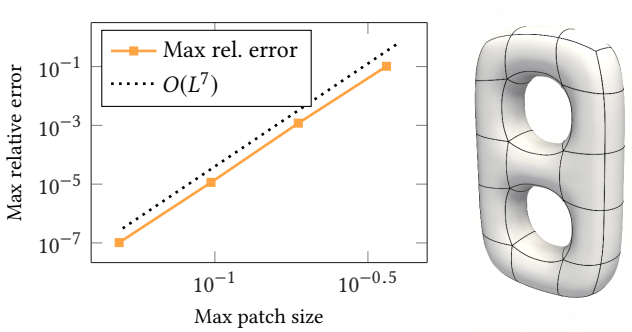


Figure 9: *Error convergence test solving Eq.* (3.5). *We evaluate a known solution on the coarse discretization and solve for ϕ. On the left, we plot the maximum relative error in the infinity norm of $\boldsymbol{u}^\Gamma$ evaluated on the surface. On the right, we show the coarse discretization of the domain boundary and patches.*

RBCs with collision resolution and convergence. Our choice of RBC representation and discretization is spectrally accurate in space for the approximation, differentiation and integration of functions on RBC surfaces, as shown in [48]. Although we use first-order time-stepping in this work, *spectral deferred correction* (SDC) can be incorporated into the algorithm exactly as in the 2D version described in [24]. This present work demonstrates second-order convergence in time; however, SDC can be made arbitrarily high-order accurate.

For collision-resolution accuracy verification, we study the convergence of our contact-free time-stepping with two RBCs in shear flow. As shown in Fig. 10, at $T = 0$, two RBCs are placed in a shear flow $\boldsymbol{u} = [z, 0, 0]$ in free-space. We first compute a reference solution without collision handling but with expensive adaptive fully implicit time-stepping to ensure accurate resolution of the lubrication layer between RBCs. This reference simulatation used spherical harmonics of order 32 and the time step had to be reduced to 6.5e-4 to prevent collisions. In Fig. 11, we show the convergence for the error in the centers of mass of each RBC as a function of the time-step size. We use spherical harmonic orders 16 and 32 for the spatial discretization to demonstrate the dominance of the time-stepping error. We observe first-order convergence with our locally-implicit backward Euler scheme which confirms that our collision resolution algorithm does not have a significant impact on time-stepping accuracy.

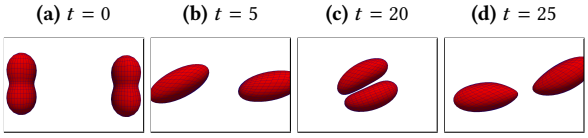


Figure 10: *Snapshots of two vesicles in shear flow.*

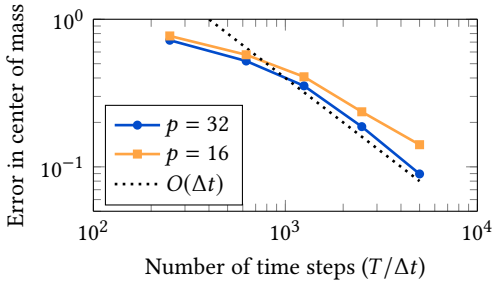


Figure 11: *Shown is the error in the final ($T = 25$) centroid location as we decrease the time step size for two spherical harmonic orders* 16 *and* 32. *We observe $O(\Delta t)$ convergence in time and hence the collission detection algorithm converges at the same order as the time stepper.*

5.4 High volume fraction

The RBC volume fraction, i.e., the ratio of volume occupied by RBCs compared to the overall blood volume is 36-48% in healthy women and 40-54% in healthy men [5]. As can be seen in the tables in Figs. 5 and 6, the volume fraction in our weak scaling simulations is below these values, which is mostly due to the procedure used to fill the blood vessel with RBCs (see the discussion in Section 5.1). However, RBC volume fractions in capillaries and small arteries is known to be be around 10-20% [41, 51], which our scaling simulations achieve. To demonstrate that we can simulate even higher volume fraction blood flows, Fig. 7 shows a test of 140 RBCs sedimenting under a gravitational force in a small capsule. The volume fraction for this example is 47%, calculated by dividing the amount of volume occupied by RBCs by the volume of the capsule. By the end of the simulation, we achieve a volume fraction of 55% in the lower part of the domain (determined by bounding the RBCs by a tighter cylinder

than the original domain boundary) since the RBCs have become more tightly packed. While such high volume fractions typically do not occur in capillary flow on average, in some scenarios (local fluctuations, sedimentation, microfluidics) these high concentrations need to be handled.

6 CONCLUSION

We have shown that our parallel platform for the simulation of red blood cell flows is capable of accurately resolved long-time simulation of red blood cell flows in complex vessel networks. We are able to achieve realistic cell volume fractions of over 47%, while avoiding collisions between cells or with the blood vessel walls. Incorporating blood vessels into red blood cell simulations requires solving a boundary integral equation, for which we use GMRES. Each GMRES iteration computes a matrix-vector product, which in turn involves singular quadrature and an FMM evaluation; the latter dominates the computation time. To avoid collisions, we solve a nonlinear complementarity problem in the implicit part of each time step. This requires repeated assembly of sparse matrices that, in principle, couple all cells globally. Nevertheless, solving this complementarity system yields close-to-optimal strong and weak scaling in our tests. Overall, the vast majority of compute time is spent in FMM evaluations, which implies that the scaling behavior of our simulation is dominated by the scalability of the FMM implementation. As discussed at the end of Section 5.2, in the future, we will employ a local singular quadrature scheme that will allow us to significantly reduce the time spent in FMM evaluations. This will not only speed up the overall simulation but also improve the weak and strong scalability of our simulation platform.

7 ACKNOWLEDGEMENTS

We would like to thank Dhairya Malhotra, Michael Shelley and Shenglong Wang for support and various discussions throughout about various aspects of this work. This work was supported by the US National Science Foundation (NSF) through grants DMS-1821334, DMS-1821305, DMS-1320621, DMS-1436591 and EAR-1646337. Computing time on TACC's Stampede2 supercomputer was provided through the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1548562.

REFERENCES

[1] N. Al Quddus, W. A. Moussa, and S. Bhattacharjee. "Motion of a spherical particle in a cylindrical channel using arbitrary Lagrangian–Eulerian method". In: *Journal of colloid and interface science* 317.2 (2008), pp. 620–630.

[2] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W Gropp, D. Kaushik, et al. *Petsc users manual revision 3.8*. Tech. rep. Argonne National Lab.(ANL), Argonne, IL (United States), 2017.

[3] P. Balogh and P. Bagchi. "A computational approach to modeling cellular-scale blood flow in complex geometry". In: *Journal of Computational Physics* 334 (2017), pp. 280–307.

[4] P. Balogh and P. Bagchi. "Direct numerical simulation of cellular-scale blood flow in 3D microvascular networks". In: *Biophysical journal* 113.12 (2017), pp. 2815–2826.

[5] H. H. Billett. "Hemoglobin and hematocrit". In: *Clinical Methods: The History, Physical, and Laboratory Examinations. 3rd edition*. Butterworths, 1990.

[6] O. P. Bruno and S. K. Lintner. "A high-order integral solver for scalar problems of diffraction by screens and apertures in three-dimensional space". In: *Journal of Computational Physics* 252 (2013), pp. 250–274.

[7] C. Burstedde, L. C. Wilcox, and O. Ghattas. "p4est: Scalable Algorithms for Parallel Adaptive Mesh Refinement on Forests of Octrees". In: *SIAM Journal on Scientific Computing* 33.3 (2011), pp. 1103–1133. DOI: 10.1137/100791634.

[8] P. B. Canham. "The minimum energy of bending as a possible explanation of the biconcave shape of the human red blood cell". In: *Journal of theoretical biology* 26.1 (1970), pp. 61–81.

[9] C. G. Caro, T. Pedley, R. Schroter, and W. Seed. *The mechanics of the circulation*. Cambridge University Press, 2012.

[10] P. Du, J. Zhao, W. Cao, and Y. Wang. "DCCD: Distributed N-Body Rigid Continuous Collision Detection for Large-Scale Virtual Environments". In: *Arabian Journal for Science and Engineering* 42.8 (2017), pp. 3141–3147.

[11] S. Fang. "A linearization method for generalized complementarity problems". In: *IEEE transactions on automatic control* 29.10 (1984), pp. 930–933.

[12] J. B. Freund. "Numerical simulation of flowing blood cells". In: *Annual review of fluid mechanics* 46 (2014), pp. 67–95.

[13] J. Gounley, M. Vardhan, and A. Randles. "A Computational Framework to Assess the Influence of Changes in Vascular Geometry on Blood Flow". In: *Proceedings of the Platform for Advanced Scientific Computing Conference*. ACM. 2017, p. 2.

[14] I. G. Graham and I. H. Sloan. "Fully discrete spectral boundary integral methods for Helmholtz problems on smooth closed surfaces in $\mathbb{R}^3$". In: *Numerische Mathematik* 92.2 (2002), pp. 289–323.

[15] L. Greengard and V. Rokhlin. "A fast algorithm for particle simulations". In: *Journal of computational physics* 73.2 (1987), pp. 325–348.

[16] L. Grinberg, J. A. Insley, V. Morozov, M. E. Papka, G. E. Karniadakis, D. Fedosov, and K. Kumaran. "A new computational paradigm in multiscale simulations: Application to brain blood flow". In: *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM. 2011, p. 5.

[17] D. Harmon, D. Panozzo, O. Sorkine, and D. Zorin. "Interference-aware geometric modeling". In: *ACM Transactions on Graphics* 30.6 (Dec. 2011), p. 1.

[18] W. Helfrich. "Elastic properties of lipid bilayers: theory and possible experiments". In: *Zeitschrift für Naturforschung C* 28.11-12 (1973), pp. 693–703.

[19] K. Iglberger and U. Rüde. "A Parallel Rigid Body Dynamics Algorithm". In: *Euro-Par 2009 Parallel Processing*. Ed. by H. Sips, D. Epema, and H.-X. Lin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 760–771.

[20] D. Kim, J.-P. Heo, J. Huh, J. Kim, and S.-e. Yoon. "HPCCD: Hybrid Parallel Continuous Collision Detection using CPUs and GPUs". In: *Computer Graphics Forum* (2009).

[21] L. af Klinteberg and A.-K. Tornberg. "A fast integral equation method for solid particles in viscous flow using quadrature by expansion". In: *Journal of Computational Physics* 326 (2016), pp. 420–445.

[22] A. Klöckner, A. Barnett, L. Greengard, and M. O'Neil. "Quadrature by expansion: A new method for the evaluation of layer potentials". In: *Journal of Computational Physics* 252 (2013), pp. 332–349.

[23] F. Liu, T. Harada, Y. Lee, and Y. J. Kim. "Real-time Collision Culling of a Million Bodies on Graphics Processing Units". In: *ACM SIGGRAPH Asia 2010 Papers*. SIGGRAPH ASIA '10. Seoul, South Korea: ACM, 2010, 154:1–154:8.

[24] L. Lu, A. Rahimian, and D. Zorin. "Contact-aware simulations of particulate Stokesian suspensions". In: *Journal of Computational Physics* 347C (Nov. 2017), pp. 160–182. DOI: 10.1016/j.jcp.2017.06.039. arXiv: 1612.02057.

[25] L. Lu, A. Rahimian, and D. Zorin. "Parallel contact-aware simulations of deformable particles in 3D Stokes flow". In: *arXiv preprint arXiv:1812.04719* (2018).

[26] D. Malhotra and G. Biros. "PVFMM: A Parallel Kernel Independent FMM for Particle and Volume Potentials". In: *Communications in Computational Physics* 18 (2015), pp. 808–830.

[27] D. Malhotra and G. Biros. "Algorithm 967: A distributed-memory fast multipole method for volume potentials". In: *ACM Transactions on Mathematical Software (TOMS)* 43.2 (2016), p. 17.

[28] D. Malhotra, A. Rahimian, D. Zorin, and G. Biros. "A parallel algorithm for long-timescale simulation of concentrated vesicle suspensions in three dimensions". In: (2017).

[29] H. Mazhar, T. Heyn, and D. Negrut. "A scalable parallel method for large collision detection problems". In: 26 (June 2011), pp. 37–55.

[30] E. Nazockdast, A. Rahimian, D. Zorin, and M. Shelley. "Fast and high-order methods for simulating fiber suspensions applied to cellular mechanics". preprint. 2015.

[31] S. Pabst, A. Koch, and W. Strasser. "Fast and Scalable CPU/GPU Collision Detection for Rigid and Deformable Surfaces". In: *Computer Graphics Forum* (2010).

[32] P. Perdikaris, L. Grinberg, and G. E. Karniadakis. "An effective fractal-tree closure model for simulating blood flow in large arterial networks". In: *Annals of biomedical engineering* 43.6 (2015), pp. 1432–1442.

[33] P. Perdikaris, L. Grinberg, and G. E. Karniadakis. "Multiscale modeling and simulation of brain blood flow". In: *Physics of Fluids* 28.2 (2016), p. 021304.

[34] M. Peyrounette, Y. Davit, M. Quintard, and S. Lorthois. "Multiscale modelling of blood flow in cerebral microcirculation: Details at capillary scale control accuracy at the level of the cortex". In: *PloS one* 13.1 (2018), e0189474.

[35] H. Power and G. Miranda. "Second kind integral equation formulation of Stokes' flows past a particle of arbitrary shape". In: *SIAM Journal on Applied Mathematics* 47.4 (1987), p. 689.

[36] C. Pozrikidis. *Boundary integral and singularity methods for linearized viscous flow*. Cambridge Texts in Applied Mathematics. Cambridge University Press, Cambridge, 1992.

[37] A. Rahimian, I. Lashuk, S. Veerapaneni, A. Chandramowlishwaran, D. Malhotra, L. Moon, R. Sampath, A. Shringarpure, J. Vetter, R. Vuduc, et al. "Petascale direct numerical simulation of blood flow on 200k cores and heterogeneous architectures". In: *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society. 2010, pp. 1–11.

[38] A. Rahimian, S. K. Veerapaneni, D. Zorin, and G. Biros. "Boundary integral method for the flow of vesicles with viscosity contrast in three dimensions". In: *Journal of Computational Physics* 298 (2015), pp. 766–786.

[39] A. Randles, E. W. Draeger, T. Oppelstrup, L. Krauss, and J. A. Gunnels. "Massively parallel models of the human circulatory system". In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM. 2015, p. 1.

[40] D. Rossinelli, Y.-H. Tang, K. Lykov, D. Alexeev, M. Bernaschi, P. Hadjidoukas, M. Bisson, W. Joubert, C. Conti, G. Karniadakis, et al. "The in-silico lab-on-a-chip: petascale and high-throughput simulations of microfluidics at cell resolution". In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM. 2015, p. 2.

[41] A. Saadat, C. J. Guido, and E. S. Shaqfeh. "Simulation of Red Blood Cell Migration in Small Arterioles: Effect of Cytoplasmic Viscosity". In: *bioRxiv* (2019), p. 572933.

[42] A. Saadat, C. J. Guido, G. Iaccarino, and E. S. Shaqfeh. "Immersed-finite-element method for deformable particle suspensions in viscous and viscoelastic media". In: *Physical Review E* 98.6 (2018), p. 063316.

[43] C. Sorgentone and A.-K. Tornberg. "A highly accurate boundary integral equation method for surfactant-laden drops in 3D". In: *Journal of Computational Physics* 360 (2018), pp. 167–191.

[44] C. Sorgentone, A.-K. Tornberg, and P. M. Vlahovska. "A 3D boundary integral method for the electrohydrodynamics of surfactant-covered drops". In: *Journal of Computational Physics* (2019).

[45] H. Sundar, D. Malhotra, and G. Biros. "HykSort: A New Variant of Hypercube Quicksort on Distributed Memory Architectures". In: *Proceedings of the 27th International ACM Conference on International Conference on Supercomputing*. ICS '13. Eugene, Oregon, USA: ACM, 2013, pp. 293–302.

[46] L. N. Trefethen. *Approximation theory and approximation practice*. Vol. 128. Siam, 2013.

[47] S. K. Veerapaneni, D. Gueyffier, G. Biros, and D. Zorin. "A numerical method for simulating the dynamics of 3D axisymmetric vesicles suspended in viscous flows". In: *Journal of Computational Physics* 228.19 (Apr. 2009), pp. 7233–7249.

[48] S. K. Veerapaneni, A. Rahimian, G. Biros, and D. Zorin. "A fast algorithm for simulating vesicle flows in three dimensions". In: *Journal of Computational Physics* 230.14 (2011), pp. 5610–5634.

[49] M. Wala and A. Klöckner. "A Fast Algorithm with Error Bounds for Quadrature by Expansion". In: *arXiv preprint arXiv:1801.04070* (2018).

[50] M. Wala and A. Klöckner. "Optimization of Fast Algorithms for Global Quadrature by Expansion Using Target-Specific Expansions". In: *arXiv preprint arXiv:1811.01110* (2018).

[51] W. Wang, T. G. Diacovo, J. Chen, J. B. Freund, and M. R. King. "Simulation of platelet, thrombus and erythrocyte hydrodynamic interactions in a 3D arteriole with in vivo comparison". In: *PLoS One* 8.10 (2013), e76949.

[52] D. Xu, E. Kaliviotis, A. Munjiza, E. Avital, C. Ji, and J. Williams. "Large scale simulation of red blood cell aggregation in shear flows". In: *Journal of Biomechanics* 46.11 (2013), pp. 1810–1817.

[53] W. Yan, H. Zhang, and M. J. Shelley. "Computing collision stress in assemblies of active spherocylinders: Applications of a fast and generic geometric method". In: *The Journal of chemical physics* 150.6 (2019), p. 064109.

[54] T. Ye, L. Peng, and Y. Li. "Three-dimensional motion and deformation of a red blood cell in bifurcated microvessels". In: *Journal of Applied Physics* 123.6 (2018), p. 064701.

[55] T. Ye, N. Phan-Thien, and C. T. Lim. "Particle-based simulations of red blood cellsâĂŤA review". In: *Journal of biomechanics* 49.11 (2016), pp. 2255–2266.

[56] T. Ye, N. Phan-Thien, C. T. Lim, L. Peng, and H. Shi. "Hybrid smoothed dissipative particle dynamics and immersed boundary method for simulation of red blood cells in flows". In: *Physical Review E* 95.6 (2017), p. 063314.

[57] L. Ying, G. Biros, and D. Zorin. "A high-order 3D boundary integral equation solver for elliptic PDEs in smooth domains". In: *Journal of Computational Physics* 219.1 (2006), pp. 247–275.

[58] L. Ying, G. Biros, and D. Zorin. "A high-order 3D boundary integral equation solver for elliptic PDEs in smooth domains". In: *Journal of Computational Physics* 219.1 (2006), pp. 247–275.

[59] H. Zhao, A. H. Isfahani, L. N. Olson, and J. B. Freund. "A spectral boundary integral method for flowing blood cells". In: *Journal of Computational Physics* 229.10 (May 2010), pp. 3726–3744.

Appendix: Artifact Description/Artifact Evaluation

SUMMARY OF THE EXPERIMENTS REPORTED

We ran our weak and strong scalability tests on Stampede2 system at the Texas Advanced Computing Center (TACC) with the Knights Landing (KNL) compute 788 nodes and the Skylake (SKX) compute nodes. I use intel 18.0.2 , Intel MPI 18.0.2, PETSc 3.10, p4est 2.0, FFTW 3 3.3.8, boost 1.68, PVFMM libraries for the simulation tests.

We use the Stampede2 system at the Texas Advanced Computing Center (TACC) to study the scalability of our algorithms and implementation. Stampede2 has two types of compute nodes, the Knights Landing (KNL) compute nodes and the Skylake (SKX) compute nodes. The SKX cluster has 1,736 dual-socket compute nodes, each with two 24-core 2.1GHz CPUs and 192GB of memory. The KNL cluster has 4,200 compute nodes, with a 68-core Intel Xeon Phi 7250 1.4Ghz CPUs and 96GB of memory plus 16GB of high-speed MCDRAM. We run our simulations in a hybrid distributed-shared memory fashion: we run one MPI process per node, with one OpenMP thread per hardware core. Our largest simulations use 256 SKX and 512 KNL nodes.

We leverage several high-performance libraries in our implementation. We use PETSc's parallel matrix and vector operations, and its parallel GMRES solver. Management and distribution of patches describing the blood vessel geometry uses the p4est library, and we use PVFMM for parallel FMM evaluation. We also heavily leverage Intel MKL for fast dense linear algebra routines at the core of our algorithms and paraview for our visualizations.

Our largest run on Stampede2 finished at 2019-04-08 01:38:16(US central time), the date range of access to Stampede2 for all tests is around 2019-03-20 to 2019-04-09.

ARTIFACT AVAILABILITY

Software Artifact Availability: Some author-created software artifacts are NOT maintained in a public repository or are NOT available under an OSI-approved license.

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

Data Artifact Availability: Some author-created data artifacts are NOT maintained in a public repository or are NOT available 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:

pvfmm “https://github.com/dmalhotra/pvfmm”
petsc “https://www.mcs.anl.gov/petsc/”
p4est “http://www.p4est.org/”
intel-parallel-studio “https://software.intel.com/en-us/articles/intel-c-compiler-180-for-linux-release-notes-for-intel-parallel-studio-xe-2018"
fftw3 “http://www.fftw.org/”
boost “https://www.boost.org/users/history/version_1_68_0.html”

BASELINE EXPERIMENTAL SETUP, AND MODIFICATIONS MADE FOR THE PAPER

Relevant hardware details: Stampede2 skylake compute nodes and Knights Landing compute nodes

Operating systems and versions: Linux version 3.10.0-957.5.1.el7.x86_64

Compilers and versions: intel 18.0.2, impi 18.0.2

Libraries and versions: intel 18.0.2, impi 18.0.2 , eigen 3.3.1, CGAL 4.10, fftw3 3.3.8, boost 1.68 , vtk 8.1.1, pvfmm, p4est 2.0, petsc 3.10-i64

Key algorithms: fast multipole method, boundary integral equation method

Output from scripts that gathers execution environment information.

```
On Stampde2 skx nodes, the execution environment information is the following:
c506-033[skx](508)$ ./collect_environment.sh
SLURM_NODELIST=c506-033
SLURM_CHECKPOINT_IMAGE_DIR=/var/slurm/checkpoint
LMOD_FAMILY_COMPILER_VERSION=18.0.2
MKLROOT=/opt/intel/compilers_and_libraries_2018.2.199/linux/mkl
I_MPI_STARTUP_MODE=pmi_shm_netmod
P4EST_DIR=/home1/apps/intel18/impi18_0/p4est/2.0
SLURM_JOB_NAME=idv31559
MANPATH=/opt/apps/libfabric/1.7.0/share/man:/opt/intel/compilers_and_libraries_2018.2.199/linux/mpi/man:/opt/intel/documentation_2018/en/man/common:/opt/intel/documentation_2018/en/debugger/gdb-ifx/man:/opt/intel/documentation_2018/en/debugger/gdb-ia/man:/opt/apps/intel18/impi18_0/fftw3/3.3.8/man
TACC_IMPI_DIR=/opt/intel/compilers_and_libraries_2018.2.199/linux/mpi
TACC_FFTW3_INC=/opt/apps/intel18/impi18_0/fftw3/3.3.8/include
VTK_DIR=/home1/apps/intel18/impi18_0/vtk/8.1.1
XDG_SESSION_ID=283415
TACC_BOOST_INC=/opt/apps/intel18/boost/1.68/include
HOSTNAME=c506-033
SLURMD_NODENAME=c506-033
SLURM_TOPOLOGY_ADDR=c506-033
TACC_INTEL_INC=/opt/intel/compilers_and_libraries_2018.2.199/linux/compiler/include/intel64
```

```
_ModuleTable003_=LjAuMiIsWyJsb2FkT3JkZXIiXT0xLHByb3B
↪ UPXt9LFsic3RhY2tEZXB0aCJdPTAsWyJzdGF0dXMiXT0iYWN
↪ 0aXZlIixbInVzZXJOYW1lIl09ImludGVsIix9LGxpYmZhYnJ
↪ pYz17WyJmbiJdPSIvb3B0L2FwcHMvbW9kdWxlZmlsZXMvbGl
↪ iZmFicmljLzEuNy4wLmx1YSIsWyJmdWxsTmFtZSJdPSJsaWJ
↪ mYWJyaWMvMS43LjAiLFsibG9hZE9yZGVyIl09Mixwcm9wVD1
↪ 7fSxbInJlZl9jb3VudCJdPTEsWyJzdGFja0RlcHRoIl09MSx
↪ bInN0YXR1cyJdPSJhY3RpdmUiLFsidXNlck5hbWUiXT0ibGl
↪ iZmFicmljIix9LHA0ZXN0PXtbImZuIl09Ii9vcHQvYXBwcy9
↪ pbnRlbDE4L2ltcGkxOF8wL21vZHVsZWZpbGVzL3A0ZXN0LzI
↪ uMC5sdWEiLFsiZnVsbE5hbWUiXT0icDRlc3QvMi4wIixsb
TACC_FAMILY_QT_VERSION=5.11.2
SLURM_PRIO_PROCESS=0
SLURM_NODE_ALIASES=(null)
INTEL_LICENSE_FILE=/home1/anonymous/USER/intel/licen
↪ ses:/opt/intel/licenses:/opt/intel/compilers_and
↪ _libraries_2018.2.199/linux/licenses
IPPROOT=/opt/intel/compilers_and_libraries_2018.2.19
↪ 9/linux/ipp
I_MPI_F77=ifort
MPICH_HOME=/opt/intel/compilers_and_libraries_2018.2
↪ .199/linux/mpi
TACC_PETSC_BIN=/home1/apps/intel18/impi18_0/petsc/3.
↪ 10/skylake-i64/bin
SHELL=/bin/bash
TERM=xterm-256color
__LMOD_REF_COUNT_MODULEPATH=/opt/apps/qt5.11.2/modul
↪ efiles:1;/opt/apps/intel18/impi18_0/modulefiles:
↪ 1;/opt/apps/intel18/modulefiles:1;/opt/apps/xsed
↪ e/modulefiles:1;/opt/apps/modulefiles:1;/opt/mod
↪ ulefiles:1
NO_HOSTSORT=1
TACC_INTEL_DIR=/opt/intel/compilers_and_libraries_20
↪ 18.2.199/linux
TACC_LIBFABRIC_BIN=/opt/apps/libfabric/1.7.0/bin
TACC_FAMILY_QT=qt5
FFTW_ROOT=/opt/apps/intel18/impi18_0/fftw3/3.3.8
HISTSIZE=1000
IDEV_SETUP_BYPASS=1.0
SLURM_JOB_QOS=normal
I_MPI_FABRICS=shm:ofi
TACC_IMPI_BIN=/opt/intel/compilers_and_libraries_201
↪ 8.2.199/linux/mpi/intel64/bin
VES3D_DIR=/home1/anonymous/USER/projects/boundary/ve
↪ s3d-cxx
SSH_CLIENT=206.76.192.52 60610 22
LMOD_SYSTEM_DEFAULT_MODULES=TACC
TMPDIR=/tmp
SLURM_TOPOLOGY_ADDR_PATTERN=node
TACC_LIBFABRIC_DIR=/opt/apps/libfabric/1.7.0
PETSC_ARCH=skylake-i64
QT_QPA_PLATFORM_PLUGIN_PATH=/opt/apps/qt5/5.11.2/plu
↪ gins
XALT_DIR=/opt/apps/xalt/xalt/
LIBRARY_PATH=/opt/intel/compilers_and_libraries_2018
↪ .2.199/linux/daal/../tbb/lib/intel64_lin/gcc4.4:
↪ /opt/intel/compilers_and_libraries_2018.2.199/li
↪ nux/daal/lib/intel64_lin:/opt/intel/compilers_an
↪ d_libraries_2018.2.199/linux/tbb/lib/intel64/gcc
↪ 4.7:/opt/intel/compilers_and_libraries_2018.2.19
↪ 9/linux/mkl/lib/intel64_lin:/opt/intel/compilers
↪ _and_libraries_2018.2.199/linux/compiler/lib/int
↪ el64_lin:/opt/intel/compilers_and_libraries_2018
↪ .2.199/linux/ipp/lib/intel64
TACC_LIBFABRIC_INC=/opt/apps/libfabric/1.7.0/include
TACC_BOOST_BIN=/opt/apps/intel18/boost/1.68/bin
LMOD_PKG=/opt/apps/lmod/lmod
TACC_FAMILY_COMPILER_VERSION=18.0.2
TACC_FAMILY_VTK_VERSION=8.1.1
QTDIR=/opt/apps/qt5/5.11.2
TACC_P4EST_BIN=/home1/apps/intel18/impi18_0/p4est/2.
↪ 0/bin
QTINC=/usr/lib64/qt-3.3/include
LMOD_VERSION=7.8.21
SSH_TTY=/dev/pts/0
SLURM_TACC_RUNLIMIT_MINS=30
__LMOD_REF_COUNT_LOADEDMODULES=intel/18.0.2:1;libfab
↪ ric/1.7.0:1;impi/18.0.2:1;fftw3/3.3.8:1;petsc/3.
↪ 10-i64:1;boost/1.68:1;qt5/5.11.2:1;vtk/8.1.1:1;p
↪ 4est/2.0:1
I_MPI_JOB_FAST_STARTUP=1
VES3D_PLATFORM=stampede
FFTW_DIR=/opt/apps/intel18/impi18_0/fftw3/3.3.8
FACEMAP_DIR=/home1/anonymous/USER/projects/boundary/
↪ mobo-temp/face_map
QT_GRAPHICSSYSTEM_CHECKED=1
TACC_INTEL_BIN=/opt/intel/compilers_and_libraries_20
↪ 18.2.199/linux/bin/intel64
TACC_IMPI_INC=/opt/intel/compilers_and_libraries_201
↪ 8.2.199/linux/mpi/intel64/include
TACC_FFTW3_DIR=/opt/apps/intel18/impi18_0/fftw3/3.3.8
USER=USER
SLURM_NNODES=1
IDEV_QDEL=scancel
__LMOD_REF_COUNT_QT_QPA_PLATFORM_PLUGIN_PATH=/opt/ap
↪ ps/qt5/5.11.2/plugins:1
```

```
LS_COLORS=rs=0:di=38;5;27:ln=38;5;51:mh=44;38;5;15:p
↪  i=40;38;5;11:so=38;5;13:do=38;5;5:bd=48;5;232;38
↪  ;5;11:cd=48;5;232;38;5;3:or=48;5;232;38;5;9:mi=0
↪  5;48;5;232;38;5;15:su=48;5;196;38;5;15:sg=48;5;1
↪  1;38;5;16:ca=48;5;196;38;5;226:tw=48;5;10;38;5;1
↪  6:ow=48;5;10;38;5;21:st=48;5;21;38;5;15:ex=38;5;
↪  34:*.tar=38;5;9:*.tgz=38;5;9:*.arc=38;5;9:*.arj=
↪  38;5;9:*.taz=38;5;9:*.lha=38;5;9:*.lz4=38;5;9:*.
↪  lzh=38;5;9:*.lzma=38;5;9:*.tlz=38;5;9:*.txz=38;5
↪  ;9:*.tzo=38;5;9:*.t7z=38;5;9:*.zip=38;5;9:*.z=38
↪  ;5;9:*.Z=38;5;9:*.dz=38;5;9:*.gz=38;5;9:*.lrz=38
↪  ;5;9:*.lz=38;5;9:*.lzo=38;5;9:*.xz=38;5;9:*.bz2=
↪  38;5;9:*.bz=38;5;9:*.tbz=38;5;9:*.tbz2=38;5;9:*.
↪  tz=38;5;9:*.deb=38;5;9:*.rpm=38;5;9:*.jar=38;5;9
↪  :*.war=38;5;9:*.ear=38;5;9:*.sar=38;5;9:*.rar=38
↪  ;5;9:*.alz=38;5;9:*.ace=38;5;9:*.zoo=38;5;9:*.cp
↪  io=38;5;9:*.7z=38;5;9:*.rz=38;5;9:*.cab=38;5;9:*
↪  .jpg=38;5;13:*.jpeg=38;5;13:*.gif=38;5;13:*.bmp=
↪  38;5;13:*.pbm=38;5;13:*.pgm=38;5;13:*.ppm=38;5;1
↪  3:*.tga=38;5;13:*.xbm=38;5;13:*.xpm=38;5;13:*.ti
↪  f=38;5;13:*.tiff=38;5;13:*.png=38;5;13:*.svg=38;
↪  5;13:*.svgz=38;5;13:*.mng=38;5;13:*.pcx=38;5;13:
↪  *.mov=38;5;13:*.mpg=38;5;13:*.mpeg=38;5;13:*.m2v
↪  =38;5;13:*.mkv=38;5;13:*.webm=38;5;13:*.ogm=38;5
↪  ;13:*.mp4=38;5;13:*.m4v=38;5;13:*.mp4v=38;5;13:*
↪  .vob=38;5;13:*.qt=38;5;13:*.nuv=38;5;13:*.wmv=38
↪  ;5;13:*.asf=38;5;13:*.rm=38;5;13:*.rmvb=38;5;13:
↪  *.flc=38;5;13:*.avi=38;5;13:*.fli=38;5;13:*.flv=
↪  38;5;13:*.gl=38;5;13:*.dl=38;5;13:*.xcf=38;5;13:
↪  *.xwd=38;5;13:*.yuv=38;5;13:*.cgm=38;5;13:*.emf=
↪  38;5;13:*.axv=38;5;13:*.anx=38;5;13:*.ogv=38;5;1
↪  3:*.ogx=38;5;13:*.aac=38;5;45:*.au=38;5;45:*.fla
↪  c=38;5;45:*.mid=38;5;45:*.midi=38;5;45:*.mka=38;
↪  5;45:*.mp3=38;5;45:*.mpc=38;5;45:*.ogg=38;5;45:*
↪  .ra=38;5;45:*.wav=38;5;45:*.axa=38;5;45:*.oga=38
↪  ;5;45:*.spx=38;5;45:*.xspf=38;5;45:
LD_LIBRARY_PATH=/home1/apps/intel18/impi18_0/p4est/2
↪  .0/lib:/home1/apps/intel18/impi18_0/vtk/8.1.1/li
↪  b:/opt/apps/gcc/6.3.0/lib64:/opt/apps/gcc/6.3.0/
↪  lib:/opt/apps/qt5/5.11.2/lib:/opt/apps/intel18/b
↪  oost/1.68/lib:/home1/apps/intel18/impi18_0/petsc
↪  /3.10/skylake-i64/lib:/opt/apps/libfabric/1.7.0/
↪  lib:/opt/intel/compilers_and_libraries_2018.2.19
↪  9/linux/mpi/intel64/lib:/opt/intel/debugger_2018
↪  /libipt/intel64/lib:/opt/intel/debugger_2018/iga
↪  /lib:/opt/intel/compilers_and_libraries_2018.2.1
↪  99/linux/daal/../tbb/lib/intel64_lin/gcc4.4:/opt
↪  /intel/compilers_and_libraries_2018.2.199/linux/
↪  daal/lib/intel64_lin:/opt/intel/compilers_and_li
↪  braries_2018.2.199/linux/tbb/lib/intel64/gcc4.7:
↪  /opt/intel/compilers_and_libraries_2018.2.199/li
↪  nux/mkl/lib/intel64_lin:/opt/intel/compilers_and
↪  _libraries_2018.2.199/linux/compiler/lib/intel64
↪  _lin:/opt/intel/compilers_and_libraries_2018.2.1
↪  99/linux/ipp/lib/intel64:/opt/intel/compilers_an
↪  d_libraries_2018.2.199/linux/compiler/lib/intel6
↪  4:/opt/apps/intel18/impi18_0/fftw3/3.3.8/lib:/ho
↪  me1/anonymous/USER/installs/cgal/lib64:/home1/an
↪  onymous/USER/installs/gmp/lib:/home1/anonymous/U
↪  SER/installs/mpfr/lib:/home1/anonymous/USER/proj
↪  ects/boundary/contact3d/lib
TACC_QT5_DIR=/opt/apps/qt5/5.11.2
PSTLROOT=/opt/intel/compilers_and_libraries_2018.2.1
↪  99/linux/pstl
TACC_PETSC_DIR=/home1/apps/intel18/impi18_0/petsc/3.
↪  10/
__TRACKER__=1
XALT_DATE_TIME=2019_05_20_15_39_13_1805
TACC_NODE_TYPE=skx
SLURM_TACC_NODES=1
SLURM_JOBID=3614475
__PERSONAL_PATH__=1
CPATH=/opt/intel/compilers_and_libraries_2018.2.199/
↪  linux/pstl/include:/opt/intel/compilers_and_libr
↪  aries_2018.2.199/linux/daal/include:/opt/intel/c
↪  ompilers_and_libraries_2018.2.199/linux/tbb/incl
↪  ude:/opt/intel/compilers_and_libraries_2018.2.19
↪  9/linux/mkl/include:/opt/intel/compilers_and_lib
↪  raries_2018.2.199/linux/ipp/include
IFC_BIN=/opt/intel/compilers_and_libraries_2018.2.19
↪  9/linux/bin/intel64
TACC_MKL_LIB=/opt/intel/compilers_and_libraries_2018
↪  .2.199/linux/mkl/lib/intel64
```

```
_ModuleTable004_=ImxvYWRPcmRlciJdPTkscHJvcFQ9e30sWyJ
↪ zdGFja0RlcHRoIl09MCxbInN0YXR1cyJdPSJhY3RpdmUiLFs
↪ idXNlck5hbWUiXT0icDRlc3QiLH0scGV0c2M9e1siZm4iXT0
↪ iL29wdC9hcHBzL2ludGVsMTgvaW1waTE4XzAvbW9kdWxlZml
↪ sZXMvcGV0c2MvMy4xMC1pNjQubHVhIixbImZ1bGxOYW1lIl0
↪ 9InBldHNjLzMuMTAtaTY0IixbImxvYWRPcmRlciJdPTUscHJ
↪ vcFQ9e30sWyJzdGFja0RlcHRoIl09MCxbInN0YXR1cyJdPSJ
↪ hY3RpdmUiLFsidXNlck5hbWUiXT0icGV0c2MvMy4xMC1pNjQ
↪ iLH0scXQ1PXtbImZuIl09Ii9vcHQvYXBwcy9tb2R1bGVmaWx
↪ lcy9xdDUvNS4xMS4yLmx1YSIsWyJmdWxsTmFtZSJdPSJxdDU
↪ vNS4xMS4yIixbImxvYWRPcmRlciJdPTcscHJvcFQ9e30s
XALT_RUN_UUID=4b841ccf-cc37-40f7-9c17-e7bb2cb0fdad
OLDSCRATCH=/oldscratch/anonymous/USER
SLURM_COMMAND=sbatch
__LMOD_REF_COUNT__LMFILES_=/opt/apps/modulefiles/int
↪ el/18.0.2.lua:1;/opt/apps/modulefiles/libfabric/
↪ 1.7.0.lua:1;/opt/apps/intel18/modulefiles/impi/1
↪ 8.0.2.lua:1;/opt/apps/intel18/impi18_0/modulefil
↪ es/fftw3/3.3.8.lua:1;/opt/apps/intel18/impi18_0/
↪ modulefiles/petsc/3.10-i64.lua:1;/opt/apps/intel
↪ 18/modulefiles/boost/1.68.lua:1;/opt/apps/module
↪ files/qt5/5.11.2.lua:1;/opt/apps/intel18/impi18_
↪ 0/modulefiles/vtk/8.1.1.lua:1;/opt/apps/intel18/
↪ impi18_0/modulefiles/p4est/2.0.lua:1
TACC_BOOST_DIR=/opt/apps/intel18/boost/1.68
PVFMM_INC=/home1/anonymous/USER/installs/pvfmm/inclu
↪ de/pvfmm
SLURM_NTASKS=1
PETSC_ROOT=/home1/apps/intel18/impi18_0/petsc/3.10/
SLURM_TACC_JOBNAME=idv31559
LMOD_FAMILY_MPI_VERSION=18.0.2
TACC_PETSC_LIB=/home1/apps/intel18/impi18_0/petsc/3.
↪ 10/skylake-i64/lib
ARCHIVER=ranch.tacc.utexas.edu
NLSPATH=/opt/intel/debugger_2018/gdb/intel64/share/l
↪ ocale/%l_%t/%N:/opt/intel/compilers_and_librarie
↪ s_2018.2.199/linux/mkl/lib/intel64/locale/%l_%t/
↪ %N:/opt/intel/compilers_and_libraries_2018.2.199
↪ /linux/compiler/lib/intel64/locale/%l_%t/%N
__LMOD_REF_COUNT_I_MPI_ROOT=/opt/intel/compilers_and
↪ _libraries_2018.2.199/linux/mpi:1
LMOD_FAMILY_QT_VERSION=5.11.2
TACC_QT5_BIN=/opt/apps/qt5/5.11.2/bin
PATH=/home1/apps/intel18/impi18_0/p4est/2.0/bin:/hom
↪ e1/apps/intel18/impi18_0/vtk/8.1.1/bin:/opt/apps
↪ /gcc/6.3.0/bin:/opt/apps/qt5/5.11.2/bin:/opt/app
↪ s/intel18/boost/1.68/bin:/home1/apps/intel18/imp
↪ i18_0/petsc/3.10/skylake-i64/bin:/opt/apps/libfa
↪ bric/1.7.0/bin:/opt/apps/intel18/impi/18.0.2/bin
↪ :/opt/intel/compilers_and_libraries_2018.2.199/l
↪ inux/mpi/intel64/bin:/opt/intel/compilers_and_li
↪ braries_2018.2.199/linux/bin/intel64:/usr/lib64/
↪ qt-3.3/bin:/usr/local/bin:/bin:/usr/bin:/opt/del
↪ l/srvadmin/bin:/opt/apps/intel18/impi18_0/fftw3/
↪ 3.3.8/bin
MAIL=/var/spool/mail/USER
TACC_SYSTEM=stampede2
_ModuleTable001_=X01vZHVsZVRhYmxlXz17WyJNVHZlcnNpb24
↪ iXT0zLFsiY19yZWJ1aWxkVGltZSJdPWZhbHNlLFsiY19zaG9
↪ ydFRpbWUiXT1mYWxzZSxkZXB0aFQ9e30sZmFtaWx5PXtbIk1
↪ QSSJdPSJpbXBpIixbImNvbXBpbGVyIl09ImludGVsIixbInF
↪ 0Il09InF0NSIsWyJ2dGsiXT0idnRrIix9LG1UPXtib29zdD1
↪ 7WyJmbiJdPSIvb3B0L2FwcHMvaW50ZWwxOC9tb2R1bGVmaWx
↪ lcy9ib29zdC8xLjY4Lmx1YSIsWyJmdWxsTmFtZSJdPSJib29
↪ zdC8xLjY4IixbImxvYWRPcmRlciJdPTYscHJvcFQ9e30sWyJ
↪ zdGFja0RlcHRoIl09MCxbInN0YXR1cyJdPSJhY3RpdmUiLFs
↪ idXNlck5hbWUiXT0iYm9vc3QiLH0sZmZ0dzM9e1siZm4iXT0
↪ iL29wdC9hcHBzL2ludGVsMTgvaW1waTE4XzAvbW9kdWxl
SLURM_TASKS_PER_NODE=1
ICC_BIN=/opt/intel/compilers_and_libraries_2018.2.19
↪ 9/linux/bin/intel64
__LMOD_REF_COUNT_NLSPATH=/opt/intel/debugger_2018/gd
↪ b/intel64/share/locale/%l_%t/%N:1;/opt/intel/com
↪ pilers_and_libraries_2018.2.199/linux/mkl/lib/in
↪ tel64/locale/%l_%t/%N:1;/opt/intel/compilers_and
↪ _libraries_2018.2.199/linux/compiler/lib/intel64
↪ /locale/%l_%t/%N:1
I_MPI_OFI_LIBRARY=/opt/apps/libfabric/1.7.0/lib/libf
↪ abric.so
STOCKYARD=/work/anonymous/USER
RUNNING_IDEV=1
SLURM_WORKING_CLUSTER=stampede2:206.76.192.2:6820:84
↪ 48
GEOGRAM_DIR=/home1/anonymous/USER/projects/boundary/
↪ mobo-temp/geogram_1.6.3
_=/bin/env
WORK=/work/anonymous/USER/stampede2
TACCINFO=/usr/local/etc/taccinfo
SLURM_JOB_ID=3614475
TACC_VTK_LIB=/home1/apps/intel18/impi18_0/vtk/8.1.1/
↪ lib
OLDWORK=/work/anonymous/USER
TBBROOT=/opt/intel/compilers_and_libraries_2018.2.19
↪ 9/linux/tbb
TACC_LIBFABRIC_LIB=/opt/apps/libfabric/1.7.0/lib
PWD=/home1/anonymous/USER
INPUTRC=/etc/inputrc
SLURM_JOB_USER=USER
SLURM_QUEUE=skx-dev
SLURM_TACC_NCORES_SET=1
_LMFILES_=/opt/apps/modulefiles/intel/18.0.2.lua:/op
↪ t/apps/modulefiles/libfabric/1.7.0.lua:/opt/apps
↪ /intel18/modulefiles/impi/18.0.2.lua:/opt/apps/i
↪ ntel18/impi18_0/modulefiles/fftw3/3.3.8.lua:/opt
↪ /apps/intel18/impi18_0/modulefiles/petsc/3.10-i6
↪ 4.lua:/opt/apps/intel18/modulefiles/boost/1.68.l
↪ ua:/opt/apps/modulefiles/qt5/5.11.2.lua:/opt/app
↪ s/intel18/impi18_0/modulefiles/vtk/8.1.1.lua:/op
↪ t/apps/intel18/impi18_0/modulefiles/p4est/2.0.lua
TACC_P4EST_DIR=/home1/apps/intel18/impi18_0/p4est/2.0
LANG=en_US.UTF-8
```

```
__LMOD_REF_COUNT_PYTHONPATH=/home1/apps/intel18/impi
↪  18_0/vtk/8.1.1/lib/site-packages/mpi4py:1;/home1
↪  /apps/intel18/impi18_0/vtk/8.1.1/lib/python2.7/s
↪  ite-packages/vtk:1
HEDGEHOG_CC=mpicc
VTK_INC_DIR=/include/vtk-8.1/
MODULEPATH=/opt/apps/qt5.11.2/modulefiles:/opt/apps/
↪  intel18/impi18_0/modulefiles:/opt/apps/intel18/m
↪  odulefiles:/opt/apps/xsede/modulefiles:/opt/apps
↪  /modulefiles:/opt/modulefiles
_ModuleTable_Sz_=6
SLURM_JOB_UID=856401
LOADEDMODULES=intel/18.0.2:libfabric/1.7.0:impi/18.0
↪  .2:fftw3/3.3.8:petsc/3.10-i64:boost/1.68:qt5/5.1
↪  1.2:vtk/8.1.1:p4est/2.0
SLURM_NODEID=0
idev_has_user_PERL5LIB=no
__BASHRC_SOURCED__=1
SLURM_TACC_ACCOUNT=TG-DPP130002
TACC_VTK_DIR=/home1/apps/intel18/impi18_0/vtk/8.1.1
SLURM_SUBMIT_DIR=/home1/anonymous/USER
TACC_VEC_FLAGS=-xCORE-AVX2 -axCORE-AVX512,MIC-AVX512
I_MPI_F90=ifort
LMOD_CMD=/opt/apps/lmod/lmod/libexec/lmod
SLURM_TASK_PID=258435
SLURM_NPROCS=1
I_MPI_CC=icc
_ModuleTable005_=WyJzdGFja0RlcHRoIl09MCxbInN0YXR1cyJ
↪  dPSJhY3RpdmUiLFsidXNlck5hbWUiXT0icXQ1Iix9LHZ0az1
↪  7WyJmbiJdPSIvb3B0L2FwcHMvaW50ZWwxOC9pbXBpMThfMC9
↪  tb2R1bGVmaWxlcy92dGsvOC4xLjEubHVhIixbImZ1bGxOYW1
↪  lIl09InZ0ay84LjEuMSIsWyJsb2FkT3JkZXIiXT04LHByb3B
↪  UPXt9LFsic3RhY2tEZXB0aCJdPTAsWyJzdGF0dXMiXT0iYWN
↪  0aXZlIixbInVzZXJOYW1lIl09InZ0ayIsfSx9LG1wYXRoQT1
↪  7Ii9vcHQvYXBwcy9xdDUuMTEuMi9tb2R1bGVmaWxlcyIsIi9
↪  vcHQvYXBwcy9pbnRlbDE4L2ltcGkxOF8wL21vZHVsZWZpbGV
↪  zIiwiL29wdC9hcHBzL2ludGVsMTgvbW9kdWxlZmlsZXMiLCI
↪  vb3B0L2FwcHMveHNlZGUvbW9kdWxlZmlsZXMiLCIvb3B0
MOBO_DIR=/home1/anonymous/USER/projects/boundary/mob
↪  o-temp
SLURM_CPUS_ON_NODE=96
DAALROOT=/opt/intel/compilers_and_libraries_2018.2.1
↪  99/linux/daal
TACC_P4EST_LIB=/home1/apps/intel18/impi18_0/p4est/2.
↪  0/lib
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass
SLURM_PROCID=0
ENVIRONMENT=BATCH
TACC_MKL_INC=/opt/intel/compilers_and_libraries_2018
↪  .2.199/linux/mkl/include
SLURM_JOB_NODELIST=c506-033
HOME=/home1/anonymous/USER
SHLVL=4
FI_PROVIDER=psm2
TACC_DOMAIN=stampede2
SLURM_LOCALID=0
__XALT_INITIAL_STATE__=LD_PRELOAD
__LMOD_REF_COUNT_PATH=/home1/apps/intel18/impi18_0/p
↪  4est/2.0/bin:1;/home1/apps/intel18/impi18_0/vtk/
↪  8.1.1/bin:1;/opt/apps/gcc/6.3.0/bin:2;/opt/apps/
↪  qt5/5.11.2/bin:1;/opt/apps/intel18/boost/1.68/bi
↪  n:1;/home1/apps/intel18/impi18_0/petsc/3.10/skyl
↪  ake-i64/bin:1;/opt/apps/libfabric/1.7.0/bin:1;/o
↪  pt/apps/intel18/impi/18.0.2/bin:1;/opt/intel/com
↪  pilers_and_libraries_2018.2.199/linux/mpi/intel6
↪  4/bin:1;/opt/intel/compilers_and_libraries_2018.
↪  2.199/linux/bin/intel64:1;/usr/lib64/qt-3.3/bin:
↪  1;/usr/local/bin:1;/bin:1;/usr/bin:1;/opt/dell/s
↪  rvadmin/bin:1;/opt/apps/intel18/impi18_0/fftw3/3
↪  .3.8/bin:1
TACC_FAMILY_COMPILER=intel
TACC_INTEL_LIB=/opt/intel/compilers_and_libraries_20
↪  18.2.199/linux/compiler/lib/intel64
I_MPI_CXX=icpc
TACC_PETSC_INC=/home1/apps/intel18/impi18_0/petsc/3.
↪  10/skylake-i64/include
TACC_FAMILY_VTK=vtk
__LMOD_REF_COUNT_CPATH=/opt/intel/compilers_and_libr
↪  aries_2018.2.199/linux/pstl/include:1;/opt/intel
↪  /compilers_and_libraries_2018.2.199/linux/daal/i
↪  nclude:1;/opt/intel/compilers_and_libraries_2018
↪  .2.199/linux/tbb/include:1;/opt/intel/compilers_
↪  and_libraries_2018.2.199/linux/mkl/include:1;/op
↪  t/intel/compilers_and_libraries_2018.2.199/linux
↪  /ipp/include:1
TACC_BOOST_LIB=/opt/apps/intel18/boost/1.68/lib
TACC_P4EST_INC=/home1/apps/intel18/impi18_0/p4est/2.
↪  0/include
_ModuleTable002_=ZmlsZXMvZmZ0dzMvMy4zLjgubHVhIixbImZ
↪  1bGxOYW1lIl09ImZmdHczLzMuMy44IixbImxvYWRPcmRlciJ
↪  dPTQscHJvcFQ9e30sWyJzdGFja0RlcHRoIl09MCxbInN0YXR
↪  1cyJdPSJhY3RpdmUiLFsidXNlck5hbWUiXT0iZmZ0dzMiLH0
↪  saW1waT17WyJmbiJdPSIvb3B0L2FwcHMvaW50ZWwxOC9tb2R
↪  1bGVmaWxlcy9pbXBpLzE4LjAuMi5sdWEiLFsiZnVsbE5hbWU
↪  iXT0iaW1waS8xOC4wLjIiLFsibG9hZE9yZGVyIl09Myxwcm9
↪  wVD17fSxbInN0YWNrRGVwdGgiXT0wLFsic3RhdHVzIl09ImF
↪  jdGl2ZSIsWyJ1c2VyTmFtZSJdPSJpbXBpIix9LGludGVsPXt
↪  bImZuIl09Ii9vcHQvYXBwcy9tb2R1bGVmaWxlcy9pbnRlbC8
↪  xOC4wLjIubHVhIixbImZ1bGxOYW1lIl09ImludGVsLzE4
SLURM_CLUSTER_NAME=stampede2
SLURM_JOB_CPUS_PER_NODE=96
SLURM_JOB_GID=814474
IFC_LIB=/opt/intel/compilers_and_libraries_2018.2.19
↪  9/linux/compiler/lib/intel64
SLURM_GTIDS=0
SLURM_SUBMIT_HOST=login2.stampede2.tacc.utexas.edu
__LMOD_REF_COUNT_INCLUDE=/home1/apps/intel18/impi18_
↪  0/vtk/8.1.1/include:1;/opt/apps/gcc/6.3.0/includ
↪  e:1
BLENDSURF_DIR=/home1/anonymous/USER/projects/boundar
↪  y/mobo-temp/blendsurf3
BASH_ENV=/etc/tacc/tacc_functions
```

```
idev_ip=c506-033
SLURM_JOB_PARTITION=skx-dev
I_MPI_FC=ifort
TACC_VTK_INC=/home1/apps/intel18/impi18_0/vtk/8.1.1/include
PVFMM_LIB=/home1/anonymous/USER/installs/pvfmm/lib/pvfmm
LOGNAME=USER
ICC_LIB=/opt/intel/compilers_and_libraries_2018.2.199/linux/compiler/lib/intel64
TACC_FAMILY_MPI=impi
LMOD_FAMILY_VTK=vtk
PYTHONPATH=/home1/apps/intel18/impi18_0/vtk/8.1.1/lib/site-packages/mpi4py:/home1/apps/intel18/impi18_0/vtk/8.1.1/lib/python2.7/site-packages/vtk
CVS_RSH=ssh
QTLIB=/usr/lib64/qt-3.3/lib
LMOD_SETTARG_TITLE_BAR=yes
BOOST_ROOT=/opt/apps/intel18/boost/1.68
SSH_CONNECTION=206.76.192.52 60610 206.76.217.53 22
XDG_DATA_DIRS=/home1/anonymous/USER/.local/share/flatpak/exports/share:/var/lib/flatpak/exports/share:/usr/local/share:/usr/share
LC_CTYPE=UTF-8
SLURM_JOB_ACCOUNT=TG-DPP130002
HEDGEHOG_CXX=mpicxx
MODULESHOME=/opt/apps/lmod/lmod
SLURM_JOB_NUM_NODES=1
__LMOD_REF_COUNT_LIBRARY_PATH=/opt/intel/compilers_and_libraries_2018.2.199/linux/daal/../tbb/lib/intel64_lin/gcc4.4:1;/opt/intel/compilers_and_libraries_2018.2.199/linux/daal/lib/intel64_lin:1;/opt/intel/compilers_and_libraries_2018.2.199/linux/tbb/lib/intel64/gcc4.7:1;/opt/intel/compilers_and_libraries_2018.2.199/linux/mkl/lib/intel64_lin:1;/opt/intel/compilers_and_libraries_2018.2.199/linux/compiler/lib/intel64_lin:1;/opt/intel/compilers_and_libraries_2018.2.199/linux/ipp/lib/intel64:1
MPI_HOME=/opt/intel/compilers_and_libraries_2018.2.199/linux/mpi
LESSOPEN=||/usr/bin/lesspipe.sh %s
LMOD_SETTARG_FULL_SUPPORT=full
OMP_NUM_THREADS=48
__LMOD_REF_COUNT_LD_LIBRARY_PATH=/home1/apps/intel18/impi18_0/p4est/2.0/lib:1;/home1/apps/intel18/impi18_0/vtk/8.1.1/lib:1;/opt/apps/gcc/6.3.0/lib64:2;/opt/apps/gcc/6.3.0/lib:2;/opt/apps/qt5/5.11.2/lib:1;/opt/apps/intel18/boost/1.68/lib:1;/home1/apps/intel18/impi18_0/petsc/3.10/skylake-i64/lib:1;/opt/apps/libfabric/1.7.0/lib:1;/opt/intel/compilers_and_libraries_2018.2.199/linux/mpi/intel64/lib:1;/opt/intel/debugger_2018/libipt/intel64/lib:1;/opt/intel/debugger_2018/iga/lib:1;/opt/intel/compilers_and_libraries_2018.2.199/linux/daal/../tbb/lib/intel64_lin/gcc4.4:1;/opt/intel/compilers_and_libraries_2018.2.199/linux/daal/lib/intel64_lin:1;/opt/intel/compilers_and_libraries_2018.2.199/linux/tbb/lib/intel64/gcc4.7:1;/opt/intel/compilers_and_libraries_2018.2.199/linux/mkl/lib/intel64_lin:1;/opt/intel/compilers_and_libraries_2018.2.199/linux/compiler/lib/intel64_lin:2;/opt/intel/compilers_and_libraries_2018.2.199/linux/ipp/lib/intel64:1;/opt/intel/compilers_and_libraries_2018.2.199/linux/compiler/lib/intel64:1;/opt/apps/intel18/impi18_0/fftw3/3.3.8/lib:1
PKG_CONFIG_PATH=/opt/intel/compilers_and_libraries_2018.2.199/linux/mkl/bin/pkgconfig:/opt/apps/intel18/impi18_0/fftw3/3.3.8/lib/pkgconfig
HEDGEHOG_DIR=/home1/anonymous/USER/projects/boundary/mobo-temp
PROMPT_COMMAND=${X_SET_TITLE_BAR:-:} "$USER@${SHOST}:${PWD/#$HOME/~}"
__Init_Default_Modules=1
TACC_FFTW3_LIB=/opt/apps/intel18/impi18_0/fftw3/3.3.8/lib
LMOD_FAMILY_COMPILER=intel
TACC_IMPI_LIB=/opt/intel/compilers_and_libraries_2018.2.199/linux/mpi/intel64/lib
TACC_VTK_BIN=/home1/apps/intel18/impi18_0/vtk/8.1.1/bin
VTK_LOCATION=/home1/apps/intel18/impi18_0/vtk/8.1.1
XDG_RUNTIME_DIR=/run/user/856401
ARCHIVE=/home/anonymous/USER
OLDHOME=/oldhome1/anonymous/USER
IDEV_PWD=/home1/anonymous/USER
__LMOD_REF_COUNT_INTEL_LICENSE_FILE=/home1/anonymous/USER/intel/licenses:1;/opt/intel/licenses:1;/opt/intel/compilers_and_libraries_2018.2.199/linux/licenses:1
TACC_FAMILY_MPI_VERSION=18.0.2
__LMOD_REF_COUNT_PKG_CONFIG_PATH=/opt/intel/compilers_and_libraries_2018.2.199/linux/mkl/bin/pkgconfig:1;/opt/apps/intel18/impi18_0/fftw3/3.3.8/lib/pkgconfig:1
TACC_QT5_LIB=/opt/apps/qt5/5.11.2/lib
LMOD_FAMILY_VTK_VERSION=8.1.1
PVFMM_DIR=/home1/anonymous/USER/installs/pvfmm/share/pvfmm
```

```
LMOD_DIR=/opt/apps/lmod/lmod/libexec
__LMOD_REF_COUNT_MANPATH=/opt/apps/libfabric/1.7.0/s
↪  hare/man:1;/opt/intel/compilers_and_libraries_20
↪  18.2.199/linux/mpi/man:1;/opt/intel/documentatio
↪  n_2018/en/man/common:1;/opt/intel/documentation_
↪  2018/en/debugger/gdb-igfx/man:1;/opt/intel/docum
↪  entation_2018/en/debugger/gdb-ia/man:1;/opt/apps
↪  /intel18/impi18_0/fftw3/3.3.8/man:1
INCLUDE=/home1/apps/intel18/impi18_0/vtk/8.1.1/inclu
↪  de:/opt/apps/gcc/6.3.0/include
_ModuleTable006_=L2FwcHMvbW9kdWxlZmlsZXMiLCIvb3B0L21
↪  vZHVsZWZpbGVzIix9LFsic3lzdGVtQmFzZU1QQVRIIl09Ii9
↪  vcHQvYXBwcy94c2VkZS9tb2R1bGVmaWxlczovb3B0L2FwcHM
↪  vbW9kdWxlZmlsZXM6L29wdC9tb2R1bGVmaWxlcyIsfQ==
PETSC_DIR=/home1/apps/intel18/impi18_0/petsc/3.10/
LMOD_FAMILY_QT=qt5
SCRATCH=/scratch/anonymous/USER
SLURM_TACC_NNODES_SET=1
SLURM_TACC_CORES=1
TACC_MKL_DIR=/opt/intel/compilers_and_libraries_2018
↪  .2.199/linux/mkl
FI_PSM2_LAZY_CONN=1
LMOD_FAMILY_MPI=impi
TACC_MPI_GETMODE=impi_hydra
I_MPI_ROOT=/opt/intel/compilers_and_libraries_2018.2
↪  .199/linux/mpi
TACC_QT5_INC=/opt/apps/qt5/5.11.2/include
MACHINE_NAME=stampede
BASH_FUNC_sbatch()=() {  echo -e "\nNOTIFICATION:
↪  sbatch not available on compute nodes. Use a login
↪  node.\n"
}
BASH_FUNC_module()=() {  if [ -z
↪  "${LMOD_SH_DBG_ON+x}" ]; then
 case "$-" in
 *v*x*)
 __lmod_sh_dbg='vx'
 ;;
 *v*)
 __lmod_sh_dbg='v'
 ;;
 *x*)
 __lmod_sh_dbg='x'
 ;;
 esac;
 fi;
 if [ -n "${__lmod_sh_dbg:-}" ]; then
 set +$__lmod_sh_dbg;
 echo "Shell debugging temporarily silenced: export
 ↪  LMOD_SH_DBG_ON=1 for Lmod's output";
 fi;
 eval $($LMOD_CMD bash "$@") && eval
 ↪  $(${LMOD_SETTARG_CMD:-:} -s sh);
 local _lmod_my_status=$?;
 if [ -n "${__lmod_sh_dbg:-}" ]; then
 echo "Shell debugging restarted";
 set -$__lmod_sh_dbg;
 unset __lmod_sh_dbg;
 fi;
 return $_lmod_my_status
}
BASH_FUNC_ml()=() {  eval $($LMOD_DIR/ml_cmd "$@")
}
+ lsb_release -a
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
+ uname -a
Linux c506-033.stampede2.tacc.utexas.edu
↪  3.10.0-957.5.1.el7.x86_64 #1 SMP Fri Feb 1
↪  14:54:57 UTC 2019 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):                96
On-line CPU(s) list:   0-95
Thread(s) per core:    2
Core(s) per socket:    24
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 85
Model name:            Intel(R) Xeon(R) Platinum 8160
↪  CPU @ 2.10GHz
Stepping:              4
CPU MHz:               2100.000
BogoMIPS:              4200.00
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              1024K
L3 cache:              33792K
NUMA node0 CPU(s):
↪  0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34
↪  ,36,38,40,42,44,46,48,50,52,54,56,58,60,62,64,66
↪  ,68,70,72,74,76,78,80,82,84,86,88,90,92,94
NUMA node1 CPU(s):
↪  1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35
↪  ,37,39,41,43,45,47,49,51,53,55,57,59,61,63,65,67
↪  ,69,71,73,75,77,79,81,83,85,87,89,91,93,95
```

```
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 art 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 3dnowprefetch epb cat_l3 cdp_l3
↪  intel_ppin intel_pt ssbd mba ibrs ibpb stibp
↪  tpr_shadow vnmi flexpriority ept vpid fsgsbase
↪  tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid
↪  rtm cqm mpx rdt_a avx512f avx512dq rdseed adx smap
↪  clflushopt clwb avx512cd avx512bw avx512vl
↪  xsaveopt xsavec xgetbv1 cqm_llc cqm_occup_llc
↪  cqm_mbm_total cqm_mbm_local dtherm ida arat pln
↪  pts pku ospke spec_ctrl intel_stibp flush_l1d
+ cat /proc/meminfo
MemTotal:       196438176 kB
MemFree:        190492500 kB
MemAvailable:   190025288 kB
Buffers:               0 kB
Cached:           138176 kB
SwapCached:            0 kB
Active:           158396 kB
Inactive:         111156 kB
Active(anon):     132300 kB
Inactive(anon):    84508 kB
Active(file):      26096 kB
Inactive(file):    26648 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:             0 kB
SwapFree:              0 kB
Dirty:                 0 kB
Writeback:             0 kB
AnonPages:        132028 kB
Mapped:            27260 kB
Shmem:             84788 kB
Slab:            2495120 kB
SReclaimable:     580428 kB
SUnreclaim:      1914692 kB
KernelStack:       25280 kB
PageTables:         7492 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:    182687500 kB
Committed_AS:     671048 kB
VmallocTotal:   34359738367 kB
VmallocUsed:     1826416 kB
VmallocChunk:   34257106468 kB
HardwareCorrupted:     0 kB
AnonHugePages:     53248 kB
CmaTotal:              0 kB
CmaFree:               0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
DirectMap4k:      673600 kB
DirectMap2M:    14667776 kB
DirectMap1G:    186646528 kB
+ inxi -F -c0
./collect_environment.sh: line 14: inxi: command not
↪  found
+ lsblk -a
NAME              MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda                 8:0    0 223.6G  0 disk
|-sda1              8:1    0     1M  0 part
|-sda2              8:2    0     1G  0 part /boot
`-sda3              8:3    0 222.6G  0 part
  |-rootvg01-lv01 253:0    0    75G  0 lvm  /
  |-rootvg01-tmp  253:1    0 143.6G  0 lvm  /tmp
  `-rootvg01-var  253:2    0     4G  0 lvm  /var
sdb                 8:16   0         0 disk
loop0               7:0    0         1 loop
loop1               7:1    0         1 loop
loop2               7:2    0         1 loop
loop3               7:3    0         1 loop
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
loop28              7:28   0         1 loop
loop29              7:29   0         1 loop
loop30              7:30   0         1 loop
loop31              7:31   0         1 loop
loop32              7:32   0         1 loop
loop33              7:33   0         1 loop
loop34              7:34   0         1 loop
```

```
loop35              7:35   0         1 loop
+ lsscsi -s
[2:0:0:0]    disk    ATA      MZ7KM240HMHQ0D3  GD53
↪  /dev/sda    240GB
[14:0:0:0]   disk    Generic  MassStorageClass WS01
↪  /dev/sdb        -
+ module list
+ '[' -z '' ']'
+ case "$-" in
+ __lmod_sh_dbg=x
+ '[' -n x ']'
+ set +x
Shell debugging temporarily silenced: export
↪  LMOD_SH_DBG_ON=1 for Lmod's output

Currently Loaded Modules:
  1) intel/18.0.2      3) impi/18.0.2   5)
  ↪  petsc/3.10-i64   7) qt5/5.11.2   9) p4est/2.0
  2) libfabric/1.7.0   4) fftw3/3.3.8   6) boost/1.68
  ↪       8) vtk/8.1.1



Shell debugging restarted
+ unset __lmod_sh_dbg
+ return 0
+ nvidia-smi
./collect_environment.sh: line 18: nvidia-smi:
↪  command not found
+ lshw -short -quiet -sanitize
+ cat
./collect_environment.sh: line 19: lshw: command not
↪  found
+ lspci
./collect_environment.sh: line 19: lspci: command not
↪  found

On Stampde2 knl nodes, the execution environment
↪  information is the following:
c455-061[knl](1020)$ ./collect_environment.sh
SLURM_NODELIST=c455-061
SLURM_CHECKPOINT_IMAGE_DIR=/var/slurm/checkpoint
LMOD_FAMILY_COMPILER_VERSION=18.0.2
MKLROOT=/opt/intel/compilers_and_libraries_2018.2.19
↪  9/linux/mkl
I_MPI_STARTUP_MODE=pmi_shm_netmod
P4EST_DIR=/home1/apps/intel18/impi18_0/p4est/2.0
SLURM_JOB_NAME=idv63880
MANPATH=/opt/apps/libfabric/1.7.0/share/man:/opt/int
↪  el/compilers_and_libraries_2018.2.199/linux/mpi/
↪  man:/opt/intel/documentation_2018/en/man/common:
↪  /opt/intel/documentation_2018/en/debugger/gdb-ig
↪  fx/man:/opt/intel/documentation_2018/en/debugger
↪  /gdb-ia/man:/opt/apps/intel18/impi18_0/fftw3/3.3
↪  .8/man
TACC_IMPI_DIR=/opt/intel/compilers_and_libraries_201
↪  8.2.199/linux/mpi
TACC_FFTW3_INC=/opt/apps/intel18/impi18_0/fftw3/3.3.
↪  8/include
VTK_DIR=/home1/apps/intel18/impi18_0/vtk/8.1.1
XDG_SESSION_ID=371971
TACC_BOOST_INC=/opt/apps/intel18/boost/1.68/include
HOSTNAME=c455-061
SLURMD_NODENAME=c455-061
SLURM_TOPOLOGY_ADDR=c455-061
TACC_INTEL_INC=/opt/intel/compilers_and_libraries_20
↪  18.2.199/linux/compiler/include/intel64
_ModuleTable003_=LjAuMiIsWyJsb2FkT3JkZXIiXT0xLHByb3B
↪  UPXt9LFsic3RhY2tEZXB0aCJdPTAsWyJzdGF0dXMiXT0iYWN
↪  0aXZlIixbInVzZXJOYW1lIl09ImludGVsIix9LGxpYmZhYnJ
↪  pYz17WyJmbiJdPSIvb3B0L2FwcHMvbW9kdWxlZmlsZXMvbGl
↪  iZmFicmljLzEuNy4wLmx1YSIsWyJmdWxsTmFtZSJdPSJsaWJ
↪  mYWJyaWMvMS43LjAiLFsibG9hZE9yZGVyIl09Mixwcm9wVD1
↪  7fSxbInJlZl9jb3VudCJdPTEsWyJzdGFja0RlcHRoIl09MSx
↪  bInN0YXR1cyJdPSJhY3RpdmUiLFsidXNlck5hbWUiXT0ibGl
↪  iZmFicmljIix9LHA0ZXN0PXtbImZuIl09Ii9vcHQvYXBwcy9
↪  pbnRlbDE4L2ltcGkxOF8wL21vZHVsZWZpbGVzL3A0ZXN0LzI
↪  uMC5sdWEiLFsiZnVsbE5hbWUiXT0icDRlc3QvMi4wIixb
TACC_FAMILY_QT_VERSION=5.11.2
SLURM_PRIO_PROCESS=0
SLURM_NODE_ALIASES=(null)
INTEL_LICENSE_FILE=/home1/anonymous/USER/intel/licen
↪  ses:/opt/intel/licenses:/opt/intel/compilers_and
↪  _libraries_2018.2.199/linux/licenses
IPPROOT=/opt/intel/compilers_and_libraries_2018.2.19
↪  9/linux/ipp
I_MPI_F77=ifort
MPICH_HOME=/opt/intel/compilers_and_libraries_2018.2
↪  .199/linux/mpi
TACC_PETSC_BIN=/home1/apps/intel18/impi18_0/petsc/3.
↪  10/skylake-i64/bin
SHELL=/bin/bash
TERM=xterm-256color
__LMOD_REF_COUNT_MODULEPATH=/opt/apps/qt5.11.2/modul
↪  efiles:1;/opt/apps/intel18/impi18_0/modulefiles:
↪  1;/opt/apps/intel18/modulefiles:1;/opt/apps/xsed
↪  e/modulefiles:1;/opt/apps/modulefiles:1;/opt/mod
↪  ulefiles:1
NO_HOSTSORT=1
TACC_INTEL_DIR=/opt/intel/compilers_and_libraries_20
↪  18.2.199/linux
TACC_LIBFABRIC_BIN=/opt/apps/libfabric/1.7.0/bin
TACC_FAMILY_QT=qt5
FFTW_ROOT=/opt/apps/intel18/impi18_0/fftw3/3.3.8
HISTSIZE=1000
IDEV_SETUP_BYPASS=1.0
SLURM_JOB_QOS=normal
I_MPI_FABRICS=shm:ofi
TACC_IMPI_BIN=/opt/intel/compilers_and_libraries_201
↪  8.2.199/linux/mpi/intel64/bin
```

```
VES3D_DIR=/home1/anonymous/USER/projects/boundary/ve
↪ s3d-cxx
SSH_CLIENT=206.76.192.54 47394 22
LMOD_SYSTEM_DEFAULT_MODULES=TACC
TMPDIR=/tmp
SLURM_TOPOLOGY_ADDR_PATTERN=node
TACC_LIBFABRIC_DIR=/opt/apps/libfabric/1.7.0
PETSC_ARCH=skylake-i64
QT_QPA_PLATFORM_PLUGIN_PATH=/opt/apps/qt5/5.11.2/plu
↪ gins
XALT_DIR=/opt/apps/xalt/xalt/
LIBRARY_PATH=/opt/intel/compilers_and_libraries_2018
↪ .2.199/linux/daal/../tbb/lib/intel64_lin/gcc4.4:
↪ /opt/intel/compilers_and_libraries_2018.2.199/li
↪ nux/daal/lib/intel64_lin:/opt/intel/compilers_an
↪ d_libraries_2018.2.199/linux/tbb/lib/intel64/gcc
↪ 4.7:/opt/intel/compilers_and_libraries_2018.2.19
↪ 9/linux/mkl/lib/intel64_lin:/opt/intel/compilers
↪ _and_libraries_2018.2.199/linux/compiler/lib/int
↪ el64_lin:/opt/intel/compilers_and_libraries_2018
↪ .2.199/linux/ipp/lib/intel64
TACC_LIBFABRIC_INC=/opt/apps/libfabric/1.7.0/include
TACC_BOOST_BIN=/opt/apps/intel18/boost/1.68/bin
LMOD_PKG=/opt/apps/lmod/lmod
TACC_FAMILY_COMPILER_VERSION=18.0.2
TACC_FAMILY_VTK_VERSION=8.1.1
QTDIR=/opt/apps/qt5/5.11.2
TACC_P4EST_BIN=/home1/apps/intel18/impi18_0/p4est/2.
↪ 0/bin
QTINC=/usr/lib64/qt-3.3/include
LMOD_VERSION=7.8.21
SSH_TTY=/dev/pts/0
SLURM_TACC_RUNLIMIT_MINS=30
LC_ALL=en_US.UTF-8
__LMOD_REF_COUNT_LOADEDMODULES=intel/18.0.2:1;libfab
↪ ric/1.7.0:1;impi/18.0.2:1;fftw3/3.3.8:1;petsc/3.
↪ 10-i64:1;boost/1.68:1;qt5/5.11.2:1;vtk/8.1.1:1;p
↪ 4est/2.0:1
I_MPI_JOB_FAST_STARTUP=1
VES3D_PLATFORM=stampede
FFTW_DIR=/opt/apps/intel18/impi18_0/fftw3/3.3.8
FACEMAP_DIR=/home1/anonymous/USER/projects/boundary/
↪ mobo-temp/face_map
QT_GRAPHICSSYSTEM_CHECKED=1
TACC_INTEL_BIN=/opt/intel/compilers_and_libraries_20
↪ 18.2.199/linux/bin/intel64
TACC_IMPI_INC=/opt/intel/compilers_and_libraries_201
↪ 8.2.199/linux/mpi/intel64/include
TACC_FFTW3_DIR=/opt/apps/intel18/impi18_0/fftw3/3.3.8
USER=USER
SLURM_NNODES=1
IDEV_QDEL=scancel
__LMOD_REF_COUNT_QT_QPA_PLATFORM_PLUGIN_PATH=/opt/ap
↪ ps/qt5/5.11.2/plugins:1
LS_COLORS=rs=0:di=38;5;27:ln=38;5;51:mh=44;38;5;15:p
↪ i=40;38;5;11:so=38;5;13:do=38;5;5:bd=48;5;232;38
↪ ;5;11:cd=48;5;232;38;5;3:or=48;5;232;38;5;9:mi=0
↪ 5;48;5;232;38;5;15:su=48;5;196;38;5;15:sg=48;5;1
↪ 1;38;5;16:ca=48;5;196;38;5;226:tw=48;5;10;38;5;1
↪ 6:ow=48;5;10;38;5;21:st=48;5;21;38;5;15:ex=38;5;
↪ 34:*.tar=38;5;9:*.tgz=38;5;9:*.arc=38;5;9:*.arj=
↪ 38;5;9:*.taz=38;5;9:*.lha=38;5;9:*.lz4=38;5;9:*.
↪ lzh=38;5;9:*.lzma=38;5;9:*.tlz=38;5;9:*.txz=38;5
↪ ;9:*.tzo=38;5;9:*.t7z=38;5;9:*.zip=38;5;9:*.z=38
↪ ;5;9:*.Z=38;5;9:*.dz=38;5;9:*.gz=38;5;9:*.lrz=38
↪ ;5;9:*.lz=38;5;9:*.lzo=38;5;9:*.xz=38;5;9:*.bz2=
↪ 38;5;9:*.bz=38;5;9:*.tbz=38;5;9:*.tbz2=38;5;9:*.
↪ tz=38;5;9:*.deb=38;5;9:*.rpm=38;5;9:*.jar=38;5;9
↪ :*.war=38;5;9:*.ear=38;5;9:*.sar=38;5;9:*.rar=38
↪ ;5;9:*.alz=38;5;9:*.ace=38;5;9:*.zoo=38;5;9:*.cp
↪ io=38;5;9:*.7z=38;5;9:*.rz=38;5;9:*.cab=38;5;9:*
↪ .jpg=38;5;13:*.jpeg=38;5;13:*.gif=38;5;13:*.bmp=
↪ 38;5;13:*.pbm=38;5;13:*.pgm=38;5;13:*.ppm=38;5;1
↪ 3:*.tga=38;5;13:*.xbm=38;5;13:*.xpm=38;5;13:*.ti
↪ f=38;5;13:*.tiff=38;5;13:*.png=38;5;13:*.svg=38;
↪ 5;13:*.svgz=38;5;13:*.mng=38;5;13:*.pcx=38;5;13:
↪ *.mov=38;5;13:*.mpg=38;5;13:*.mpeg=38;5;13:*.m2v
↪ =38;5;13:*.mkv=38;5;13:*.webm=38;5;13:*.ogm=38;5
↪ ;13:*.mp4=38;5;13:*.m4v=38;5;13:*.mp4v=38;5;13:*
↪ .vob=38;5;13:*.qt=38;5;13:*.nuv=38;5;13:*.wmv=38
↪ ;5;13:*.asf=38;5;13:*.rm=38;5;13:*.rmvb=38;5;13:
↪ *.flc=38;5;13:*.avi=38;5;13:*.fli=38;5;13:*.flv=
↪ 38;5;13:*.gl=38;5;13:*.dl=38;5;13:*.xcf=38;5;13:
↪ *.xwd=38;5;13:*.yuv=38;5;13:*.cgm=38;5;13:*.emf=
↪ 38;5;13:*.axv=38;5;13:*.anx=38;5;13:*.ogv=38;5;1
↪ 3:*.ogx=38;5;13:*.aac=38;5;45:*.au=38;5;45:*.fla
↪ c=38;5;45:*.mid=38;5;45:*.midi=38;5;45:*.mka=38;
↪ 5;45:*.mp3=38;5;45:*.mpc=38;5;45:*.ogg=38;5;45:*
↪ .ra=38;5;45:*.wav=38;5;45:*.axa=38;5;45:*.oga=38
↪ ;5;45:*.spx=38;5;45:*.xspf=38;5;45:
```

```
LD_LIBRARY_PATH=/home1/apps/intel18/impi18_0/p4est/2
↪ .0/lib:/home1/apps/intel18/impi18_0/vtk/8.1.1/li
↪ b:/opt/apps/gcc/6.3.0/lib64:/opt/apps/gcc/6.3.0/
↪ lib:/opt/apps/qt5/5.11.2/lib:/opt/apps/intel18/b
↪ oost/1.68/lib:/home1/apps/intel18/impi18_0/petsc
↪ /3.10/skylake-i64/lib:/opt/apps/libfabric/1.7.0/
↪ lib:/opt/intel/compilers_and_libraries_2018.2.19
↪ 9/linux/mpi/intel64/lib:/opt/intel/debugger_2018
↪ /libipt/intel64/lib:/opt/intel/debugger_2018/iga
↪ /lib:/opt/intel/compilers_and_libraries_2018.2.1
↪ 99/linux/daal/../tbb/lib/intel64_lin/gcc4.4:/opt
↪ /intel/compilers_and_libraries_2018.2.199/linux/
↪ daal/lib/intel64_lin:/opt/intel/compilers_and_li
↪ braries_2018.2.199/linux/tbb/lib/intel64/gcc4.7:
↪ /opt/intel/compilers_and_libraries_2018.2.199/li
↪ nux/mkl/lib/intel64_lin:/opt/intel/compilers_and
↪ _libraries_2018.2.199/linux/compiler/lib/intel64
↪ _lin:/opt/intel/compilers_and_libraries_2018.2.1
↪ 99/linux/ipp/lib/intel64:/opt/intel/compilers_an
↪ d_libraries_2018.2.199/linux/compiler/lib/intel6
↪ 4:/opt/apps/intel18/impi18_0/fftw3/3.3.8/lib:/ho
↪ me1/anonymous/USER/installs/cgal/lib64:/home1/an
↪ onymous/USER/installs/gmp/lib:/home1/anonymous/U
↪ SER/installs/mpfr/lib:/home1/anonymous/USER/proj
↪ ects/boundary/contact3d/lib
TACC_QT5_DIR=/opt/apps/qt5/5.11.2
PSTLROOT=/opt/intel/compilers_and_libraries_2018.2.1
↪ 99/linux/pstl
TACC_PETSC_DIR=/home1/apps/intel18/impi18_0/petsc/3.
↪ 10/
__TRACKER__=1
XALT_DATE_TIME=2019_05_20_14_47_32_9791
TACC_NODE_TYPE=knl
SLURM_TACC_NODES=1
SLURM_JOBID=3614206
__PERSONAL_PATH__=1
CPATH=/opt/intel/compilers_and_libraries_2018.2.199/
↪ linux/pstl/include:/opt/intel/compilers_and_libr
↪ aries_2018.2.199/linux/daal/include:/opt/intel/c
↪ ompilers_and_libraries_2018.2.199/linux/tbb/incl
↪ ude:/opt/intel/compilers_and_libraries_2018.2.19
↪ 9/linux/mkl/include:/opt/intel/compilers_and_lib
↪ raries_2018.2.199/linux/ipp/include
IFC_BIN=/opt/intel/compilers_and_libraries_2018.2.19
↪ 9/linux/bin/intel64
TACC_MKL_LIB=/opt/intel/compilers_and_libraries_2018
↪ .2.199/linux/mkl/lib/intel64
_ModuleTable004_=ImxvYWRPcmRlciJdPTkscHJvcFQ9e30sWyJ
↪ zdGFja0RlcHRoIl09MCxbInN0YXR1cyJdPSJhY3RpdmUiLFs
↪ idXNlck5hbWUiXT0icDRlc3QiLH0scGV0c2M9e1siZm4iXT0
↪ iL29wdC9hcHBzL2ludGVsMTgvaW1waTE4XzAvbW9kdWxlZml
↪ sZXMvcGV0c2MvMy4xMC1pNjQubHVhIixbImZ1bGxOYW1lIl0
↪ 9InBldHNjLzMuMTAtaTY0IixbImxvYWRPcmRlciJdPTUscHJ
↪ vcFQ9e30sWyJzdGFja0RlcHRoIl09MCxbInN0YXR1cyJdPSJ
↪ hY3RpdmUiLFsidXNlck5hbWUiXT0icGV0c2MvMy4xMC1pNjQ
↪ iLH0scXQ1PXtbImZuIl09Ii9vcHQvYXBwcy9tb2R1bGVmaWx
↪ lcy9xdDUvNS4xMS4yLmx1YSIsWyJmdWxsTmFtZSJdPSJxdDU
↪ vNS4xMS4yIixbImxvYWRPcmRlciJdPTcscHJvcFQ9e30s
XALT_RUN_UUID=f56fe2e9-d1e7-498b-9289-9ce6973d60f2
OLDSCRATCH=/oldscratch/anonymous/USER
SLURM_COMMAND=sbatch
__LMOD_REF_COUNT__LMFILES_=/opt/apps/modulefiles/int
↪ el/18.0.2.lua:1;/opt/apps/modulefiles/libfabric/
↪ 1.7.0.lua:1;/opt/apps/intel18/modulefiles/impi/1
↪ 8.0.2.lua:1;/opt/apps/intel18/impi18_0/modulefil
↪ es/fftw3/3.3.8.lua:1;/opt/apps/intel18/impi18_0/
↪ modulefiles/petsc/3.10-i64.lua:1;/opt/apps/intel
↪ 18/modulefiles/boost/1.68.lua:1;/opt/apps/module
↪ files/qt5/5.11.2.lua:1;/opt/apps/intel18/impi18_
↪ 0/modulefiles/vtk/8.1.1.lua:1;/opt/apps/intel18/
↪ impi18_0/modulefiles/p4est/2.0.lua:1
TACC_BOOST_DIR=/opt/apps/intel18/boost/1.68
PVFMM_INC=/home1/anonymous/USER/installs/pvfmm/inclu
↪ de/pvfmm
SLURM_NTASKS=1
PETSC_ROOT=/home1/apps/intel18/impi18_0/petsc/3.10/
SLURM_TACC_JOBNAME=idv63880
LMOD_FAMILY_MPI_VERSION=18.0.2
TACC_PETSC_LIB=/home1/apps/intel18/impi18_0/petsc/3.
↪ 10/skylake-i64/lib
ARCHIVER=ranch.tacc.utexas.edu
NLSPATH=/opt/intel/debugger_2018/gdb/intel64/share/l
↪ ocale/%l_%t/%N:/opt/intel/compilers_and_librarie
↪ s_2018.2.199/linux/mkl/lib/intel64/locale/%l_%t/
↪ %N:/opt/intel/compilers_and_libraries_2018.2.199
↪ /linux/compiler/lib/intel64/locale/%l_%t/%N
__LMOD_REF_COUNT_I_MPI_ROOT=/opt/intel/compilers_and
↪ _libraries_2018.2.199/linux/mpi:1
LMOD_FAMILY_QT_VERSION=5.11.2
TACC_QT5_BIN=/opt/apps/qt5/5.11.2/bin
PATH=/home1/apps/intel18/impi18_0/p4est/2.0/bin:/hom
↪ e1/apps/intel18/impi18_0/vtk/8.1.1/bin:/opt/apps
↪ /gcc/6.3.0/bin:/opt/apps/qt5/5.11.2/bin:/opt/app
↪ s/intel18/boost/1.68/bin:/home1/apps/intel18/imp
↪ i18_0/petsc/3.10/skylake-i64/bin:/opt/apps/libfa
↪ bric/1.7.0/bin:/opt/apps/intel18/impi/18.0.2/bin
↪ :/opt/intel/compilers_and_libraries_2018.2.199/l
↪ inux/mpi/intel64/bin:/opt/intel/compilers_and_li
↪ braries_2018.2.199/linux/bin/intel64:/usr/lib64/
↪ qt-3.3/bin:/usr/local/bin:/bin:/usr/bin:/opt/del
↪ l/srvadmin/bin:/opt/apps/intel18/impi18_0/fftw3/
↪ 3.3.8/bin
MAIL=/var/spool/mail/USER
```

```
TACC_SYSTEM=stampede2
_ModuleTable001_=X01vZHVsZVRhYmxlXz17WyJNVHZlcnNpb24
↪ iXT0zLFsiY19yZWJ1aWxkVGltZSJdPWZhbHNlLFsiY19zaG9
↪ ydFRpbWUiXT1mYWxzZSxkZXB0aFQ9e30sZmFtaWx5PXtbIk1
↪ QSSJdPSJpbXBpIixbImNvbXBpbGVyIl09ImludGVsIixbInF
↪ 0Il09InF0NSIsWyJ2dGsiXT0idnRrIix9LG1UPXtib29zdD1
↪ 7WyJmbiJdPSIvb3B0L2FwcHMvaW50ZWwxOC9tb2R1bGVmaWx
↪ lcy9ib29zdC8xLjY4Lmx1YSIsWyJmdWxsTmFtZSJdPSJib29
↪ zdC8xLjY4IixbImxvYWRPcmRlciJdPTYscHJvcFQ9e30sWyJ
↪ zdGFja0RlcHRoIl09MCxbInN0YXR1cyJdPSJhY3RpdmUiLFs
↪ idXNlck5hbWUiXT0iYm9vc3QiLH0sZmZ0dzM9e1siZm4iXT0
↪ iL29wdC9hcHBzL2ludGVsMTgvaW1waTE4XzAvbW9kdWxl
SLURM_TASKS_PER_NODE=1
ICC_BIN=/opt/intel/compilers_and_libraries_2018.2.19
↪ 9/linux/bin/intel64
__LMOD_REF_COUNT_NLSPATH=/opt/intel/debugger_2018/gd
↪ b/intel64/share/locale/%l_%t/%N:1;/opt/intel/com
↪ pilers_and_libraries_2018.2.199/linux/mkl/lib/in
↪ tel64/locale/%l_%t/%N:1;/opt/intel/compilers_and
↪ _libraries_2018.2.199/linux/compiler/lib/intel64
↪ /locale/%l_%t/%N:1
I_MPI_OFI_LIBRARY=/opt/apps/libfabric/1.7.0/lib/libf
↪ abric.so
STOCKYARD=/work/anonymous/USER
RUNNING_IDEV=1
SLURM_WORKING_CLUSTER=stampede2:206.76.192.2:6820:84
↪ 48
GEOGRAM_DIR=/home1/anonymous/USER/projects/boundary/
↪ mobo-temp/geogram_1.6.3
_=/bin/env
WORK=/work/anonymous/USER/stampede2
TACCINFO=/usr/local/etc/taccinfo
SLURM_JOB_ID=3614206
TACC_VTK_LIB=/home1/apps/intel18/impi18_0/vtk/8.1.1/
↪ lib
OLDWORK=/work/anonymous/USER
TBBROOT=/opt/intel/compilers_and_libraries_2018.2.19
↪ 9/linux/tbb
TACC_LIBFABRIC_LIB=/opt/apps/libfabric/1.7.0/lib
PWD=/home1/anonymous/USER
INPUTRC=/etc/inputrc
SLURM_JOB_USER=USER
SLURM_QUEUE=development
SLURM_TACC_NCORES_SET=1
_LMFILES_=/opt/apps/modulefiles/intel/18.0.2.lua:/op
↪ t/apps/modulefiles/libfabric/1.7.0.lua:/opt/apps
↪ /intel18/modulefiles/impi/18.0.2.lua:/opt/apps/i
↪ ntel18/impi18_0/modulefiles/fftw3/3.3.8.lua:/opt
↪ /apps/intel18/impi18_0/modulefiles/petsc/3.10-i6
↪ 4.lua:/opt/apps/intel18/modulefiles/boost/1.68.l
↪ ua:/opt/apps/modulefiles/qt5/5.11.2.lua:/opt/app
↪ s/intel18/impi18_0/modulefiles/vtk/8.1.1.lua:/op
↪ t/apps/intel18/impi18_0/modulefiles/p4est/2.0.lua
TACC_P4EST_DIR=/home1/apps/intel18/impi18_0/p4est/2.0
LANG=en_US.UTF-8
__LMOD_REF_COUNT_PYTHONPATH=/home1/apps/intel18/impi
↪ 18_0/vtk/8.1.1/lib/site-packages/mpi4py:1;/home1
↪ /apps/intel18/impi18_0/vtk/8.1.1/lib/python2.7/s
↪ ite-packages/vtk:1
HEDGEHOG_CC=mpicc
VTK_INC_DIR=/include/vtk-8.1/
MODULEPATH=/opt/apps/qt5.11.2/modulefiles:/opt/apps/
↪ intel18/impi18_0/modulefiles:/opt/apps/intel18/m
↪ odulefiles:/opt/apps/xsede/modulefiles:/opt/apps
↪ /modulefiles:/opt/modulefiles
_ModuleTable_Sz_=6
SLURM_JOB_UID=846024
LOADEDMODULES=intel/18.0.2:libfabric/1.7.0:impi/18.0
↪ .2:fftw3/3.3.8:petsc/3.10-i64:boost/1.68:qt5/5.1
↪ 1.2:vtk/8.1.1:p4est/2.0
SLURM_NODEID=0
idev_has_user_PERL5LIB=no
__BASHRC_SOURCED__=1
TACC_VTK_DIR=/home1/apps/intel18/impi18_0/vtk/8.1.1
SLURM_SUBMIT_DIR=/home1/anonymous/USER
TACC_VEC_FLAGS=-xCORE-AVX2 -axCORE-AVX512,MIC-AVX512
I_MPI_F90=ifort
LMOD_CMD=/opt/apps/lmod/lmod/libexec/lmod
SLURM_TASK_PID=239883
SLURM_NPROCS=1
I_MPI_CC=icc
_ModuleTable005_=WyJzdGFja0RlcHRoIl09MCxbInN0YXR1cyJ
↪ dPSJhY3RpdmUiLFsidXNlck5hbWUiXT0icXQ1Iix9LHZ0az1
↪ 7WyJmbiJdPSIvb3B0L2FwcHMvaW50ZWwxOC9pbXBpMThfMC9
↪ tb2R1bGVmaWxlcy92dGsvOC4xLjEubHVhIixbImZ1bGxOYW1
↪ lIl09InZ0ay84LjEuMSIsWyJsb2FkT3JkZXIiXT04LHByb3B
↪ UPXt9LFsic3RhY2tEZXB0aCJdPTAsWyJzdGF0dXMiXT0iYWN
↪ 0aXZlIixbInVzZXJOYW1lIl09InZ0ayIsfSx9LG1wYXRoQT1
↪ 7Ii9vcHQvYXBwcy9xdDUuMTEuMi9tb2R1bGVmaWxlcyIsIi9
↪ vcHQvYXBwcy9pbnRlbDE4L2ltcGkxOF8wL21vZHVsZWZpbGV
↪ zIiwiL29wdC9hcHBzL2ludGVsMTgvbW9kdWxlZmlsZXMiLCI
↪ vb3B0L2FwcHMveHNlZGUvbW9kdWxlZmlsZXMiLCIvb3B0
MOBO_DIR=/home1/anonymous/USER/projects/boundary/mob
↪ o-temp
SLURM_CPUS_ON_NODE=272
DAALROOT=/opt/intel/compilers_and_libraries_2018.2.1
↪ 99/linux/daal
TACC_P4EST_LIB=/home1/apps/intel18/impi18_0/p4est/2.
↪ 0/lib
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass
SLURM_PROCID=0
ENVIRONMENT=BATCH
TACC_MKL_INC=/opt/intel/compilers_and_libraries_2018
↪ .2.199/linux/mkl/include
SLURM_JOB_NODELIST=c455-061
HOME=/home1/anonymous/USER
SHLVL=4
FI_PROVIDER=psm2
TACC_DOMAIN=stampede2
SLURM_LOCALID=0
__XALT_INITIAL_STATE__=LD_PRELOAD
```

```
LANGUAGE=en_US.UTF-8
__LMOD_REF_COUNT_PATH=/home1/apps/intel18/impi18_0/p
↪  4est/2.0/bin:1;/home1/apps/intel18/impi18_0/vtk/
↪  8.1.1/bin:1;/opt/apps/gcc/6.3.0/bin:2;/opt/apps/
↪  qt5/5.11.2/bin:1;/opt/apps/intel18/boost/1.68/bi
↪  n:1;/home1/apps/intel18/impi18_0/petsc/3.10/skyl
↪  ake-i64/bin:1;/opt/apps/libfabric/1.7.0/bin:1;/o
↪  pt/apps/intel18/impi/18.0.2/bin:1;/opt/intel/com
↪  pilers_and_libraries_2018.2.199/linux/mpi/intel6
↪  4/bin:1;/opt/intel/compilers_and_libraries_2018.
↪  2.199/linux/bin/intel64:1;/usr/lib64/qt-3.3/bin:
↪  1;/usr/local/bin:1;/bin:1;/usr/bin:1;/opt/dell/s
↪  rvadmin/bin:1;/opt/apps/intel18/impi18_0/fftw3/3
↪  .3.8/bin:1
TACC_FAMILY_COMPILER=intel
TACC_INTEL_LIB=/opt/intel/compilers_and_libraries_20
↪  18.2.199/linux/compiler/lib/intel64
I_MPI_CXX=icpc
TACC_PETSC_INC=/home1/apps/intel18/impi18_0/petsc/3.
↪  10/skylake-i64/include
TACC_FAMILY_VTK=vtk
__LMOD_REF_COUNT_CPATH=/opt/intel/compilers_and_libr
↪  aries_2018.2.199/linux/pstl/include:1;/opt/intel
↪  /compilers_and_libraries_2018.2.199/linux/daal/i
↪  nclude:1;/opt/intel/compilers_and_libraries_2018
↪  .2.199/linux/tbb/include:1;/opt/intel/compilers_
↪  and_libraries_2018.2.199/linux/mkl/include:1;/op
↪  t/intel/compilers_and_libraries_2018.2.199/linux
↪  /ipp/include:1
TACC_BOOST_LIB=/opt/apps/intel18/boost/1.68/lib
TACC_P4EST_INC=/home1/apps/intel18/impi18_0/p4est/2.
↪  0/include
_ModuleTable002_=ZmlsZXMvZmZ0dzMvMy4zLjgubHVhIixbImZ
↪  1bGxOYW1lIl09ImZmdHczLzMuMy44IixbImxvYWRPcmRlciJ
↪  dPTQscHJvcFQ9e30sWyJzdGFja0RlcHRoIl09MCxbInN0YXR
↪  1cyJdPSJhY3RpdmUiLFsidXNlck5hbWUiXT0iZmZ0dzMiLH0
↪  saW1waT17WyJmbiJdPSIvb3B0L2FwcHMvaW50ZWwxOC9tb2R
↪  1bGVmaWxlcy9pbXBpLzE4LjAuMi5sdWEiLFsiZnVsbE5hbWU
↪  iXT0iaW1waS8xOC4wLjIiLFsibG9hZE9yZGVyIl09Myxwcm9
↪  wVD17fSxbInN0YWNrRGVwdGgiXT0wLFsic3RhdHVzIl09ImF
↪  jdGl2ZSIsWyJ1c2VyTmFtZSJdPSJpbXBpIix9LGludGVsPXt
↪  bImZuIl09Ii9vcHQvYXBwcy9tb2R1bGVmaWxlcy9pbnRlbC8
↪  xOC4wLjIubHVhIixbImZ1bGxOYW1lIl09ImludGVsLzE4
SLURM_CLUSTER_NAME=stampede2
SLURM_JOB_CPUS_PER_NODE=272
SLURM_JOB_GID=814474
IFC_LIB=/opt/intel/compilers_and_libraries_2018.2.19
↪  9/linux/compiler/lib/intel64
SLURM_GTIDS=0
SLURM_SUBMIT_HOST=login4.stampede2.tacc.utexas.edu
__LMOD_REF_COUNT_INCLUDE=/home1/apps/intel18/impi18_
↪  0/vtk/8.1.1/include:1;/opt/apps/gcc/6.3.0/includ
↪  e:1
BLENDSURF_DIR=/home1/anonymous/USER/projects/boundar
↪  y/mobo-temp/blendsurf3
BASH_ENV=/etc/tacc/tacc_functions
idev_ip=c455-061
SLURM_JOB_PARTITION=development
I_MPI_FC=ifort
TACC_VTK_INC=/home1/apps/intel18/impi18_0/vtk/8.1.1/
↪  include
PVFMM_LIB=/home1/anonymous/USER/installs/pvfmm/lib/p
↪  vfmm
LOGNAME=USER
ICC_LIB=/opt/intel/compilers_and_libraries_2018.2.19
↪  9/linux/compiler/lib/intel64
TACC_FAMILY_MPI=impi
LMOD_FAMILY_VTK=vtk
PYTHONPATH=/home1/apps/intel18/impi18_0/vtk/8.1.1/li
↪  b/site-packages/mpi4py:/home1/apps/intel18/impi1
↪  8_0/vtk/8.1.1/lib/python2.7/site-packages/vtk
CVS_RSH=ssh
QTLIB=/usr/lib64/qt-3.3/lib
LMOD_SETTARG_TITLE_BAR=yes
BOOST_ROOT=/opt/apps/intel18/boost/1.68
SSH_CONNECTION=206.76.192.54 47394 206.76.206.1 22
XDG_DATA_DIRS=/home1/anonymous/USER/.local/share/fla
↪  tpak/exports/share:/var/lib/flatpak/exports/shar
↪  e:/usr/local/share:/usr/share
LC_CTYPE=UTF-8
SLURM_JOB_ACCOUNT=TG-DPP130002
HEDGEHOG_CXX=mpicxx
MODULESHOME=/opt/apps/lmod/lmod
SLURM_JOB_NUM_NODES=1
__LMOD_REF_COUNT_LIBRARY_PATH=/opt/intel/compilers_a
↪  nd_libraries_2018.2.199/linux/daal/../tbb/lib/in
↪  tel64_lin/gcc4.4:1;/opt/intel/compilers_and_libr
↪  aries_2018.2.199/linux/daal/lib/intel64_lin:1;/o
↪  pt/intel/compilers_and_libraries_2018.2.199/linu
↪  x/tbb/lib/intel64/gcc4.7:1;/opt/intel/compilers_
↪  and_libraries_2018.2.199/linux/mkl/lib/intel64_l
↪  in:1;/opt/intel/compilers_and_libraries_2018.2.1
↪  99/linux/compiler/lib/intel64_lin:1;/opt/intel/c
↪  ompilers_and_libraries_2018.2.199/linux/ipp/lib/
↪  intel64:1
MPI_HOME=/opt/intel/compilers_and_libraries_2018.2.1
↪  99/linux/mpi
LESSOPEN=||/usr/bin/lesspipe.sh %s
LMOD_SETTARG_FULL_SUPPORT=full
OMP_NUM_THREADS=48
```

```
__LMOD_REF_COUNT_LD_LIBRARY_PATH=/home1/apps/intel18
↪  /impi18_0/p4est/2.0/lib:1;/home1/apps/intel18/im
↪  pi18_0/vtk/8.1.1/lib:1;/opt/apps/gcc/6.3.0/lib64
↪  :2;/opt/apps/gcc/6.3.0/lib:2;/opt/apps/qt5/5.11.
↪  2/lib:1;/opt/apps/intel18/boost/1.68/lib:1;/home
↪  1/apps/intel18/impi18_0/petsc/3.10/skylake-i64/l
↪  ib:1;/opt/apps/libfabric/1.7.0/lib:1;/opt/intel/
↪  compilers_and_libraries_2018.2.199/linux/mpi/int
↪  el64/lib:1;/opt/intel/debugger_2018/libipt/intel
↪  64/lib:1;/opt/intel/debugger_2018/iga/lib:1;/opt
↪  /intel/compilers_and_libraries_2018.2.199/linux/
↪  daal/../tbb/lib/intel64_lin/gcc4.4:1;/opt/intel/
↪  compilers_and_libraries_2018.2.199/linux/daal/li
↪  b/intel64_lin:1;/opt/intel/compilers_and_librari
↪  es_2018.2.199/linux/tbb/lib/intel64/gcc4.7:1;/op
↪  t/intel/compilers_and_libraries_2018.2.199/linux
↪  /mkl/lib/intel64_lin:1;/opt/intel/compilers_and_
↪  libraries_2018.2.199/linux/compiler/lib/intel64_
↪  lin:2;/opt/intel/compilers_and_libraries_2018.2.
↪  199/linux/ipp/lib/intel64:1;/opt/intel/compilers
↪  _and_libraries_2018.2.199/linux/compiler/lib/int
↪  el64:1;/opt/apps/intel18/impi18_0/fftw3/3.3.8/li
↪  b:1
PKG_CONFIG_PATH=/opt/intel/compilers_and_libraries_2
↪  018.2.199/linux/mkl/bin/pkgconfig:/opt/apps/inte
↪  l18/impi18_0/fftw3/3.3.8/lib/pkgconfig
HEDGEHOG_DIR=/home1/anonymous/USER/projects/boundary
↪  /mobo-temp
TACC_OVERRIDE_PROJECT=TG-DPP130002
PROMPT_COMMAND=${X_SET_TITLE_BAR:-:}
↪  "$USER@${SHOST}:${PWD/#$HOME/~}"
__Init_Default_Modules=1
TACC_FFTW3_LIB=/opt/apps/intel18/impi18_0/fftw3/3.3.
↪  8/lib
LMOD_FAMILY_COMPILER=intel
TACC_IMPI_LIB=/opt/intel/compilers_and_libraries_201
↪  8.2.199/linux/mpi/intel64/lib
TACC_VTK_BIN=/home1/apps/intel18/impi18_0/vtk/8.1.1/
↪  bin
VTK_LOCATION=/home1/apps/intel18/impi18_0/vtk/8.1.1
XDG_RUNTIME_DIR=/run/user/846024
ARCHIVE=/home/anonymous/USER
OLDHOME=/oldhome1/anonymous/USER
IDEV_PWD=/home1/anonymous/USER
__LMOD_REF_COUNT_INTEL_LICENSE_FILE=/home1/anonymous
↪  /USER/intel/licenses:1;/opt/intel/licenses:1;/op
↪  t/intel/compilers_and_libraries_2018.2.199/linux
↪  /licenses:1
TACC_FAMILY_MPI_VERSION=18.0.2
__LMOD_REF_COUNT_PKG_CONFIG_PATH=/opt/intel/compiler
↪  s_and_libraries_2018.2.199/linux/mkl/bin/pkgconf
↪  ig:1;/opt/apps/intel18/impi18_0/fftw3/3.3.8/lib/
↪  pkgconfig:1
TACC_QT5_LIB=/opt/apps/qt5/5.11.2/lib
LMOD_FAMILY_VTK_VERSION=8.1.1
PVFMM_DIR=/home1/anonymous/USER/installs/pvfmm/share
↪  /pvfmm
LMOD_DIR=/opt/apps/lmod/lmod/libexec
__LMOD_REF_COUNT_MANPATH=/opt/apps/libfabric/1.7.0/s
↪  hare/man:1;/opt/intel/compilers_and_libraries_20
↪  18.2.199/linux/mpi/man:1;/opt/intel/documentatio
↪  n_2018/en/man/common:1;/opt/intel/documentation_
↪  2018/en/debugger/gdb-igfx/man:1;/opt/intel/docum
↪  entation_2018/en/debugger/gdb-ia/man:1;/opt/apps
↪  /intel18/impi18_0/fftw3/3.3.8/man:1
INCLUDE=/home1/apps/intel18/impi18_0/vtk/8.1.1/inclu
↪  de:/opt/apps/gcc/6.3.0/include
_ModuleTable006_=L2FwcHMvbW9kdWxlZmlsZXMiLCIvb3B0L21
↪  vZHVsZWZpbGVzIix9LFsic3lzdGVtQmFzZU1QQVRIIl09Ii9
↪  vcHQvYXBwcy94c2VkZS9tb2R1bGVmaWxlczovb3B0L2FwcHM
↪  vbW9kdWxlZmlsZXM6L29wdC9tb2R1bGVmaWxlcyIsfQ==
PETSC_DIR=/home1/apps/intel18/impi18_0/petsc/3.10/
LMOD_FAMILY_QT=qt5
SCRATCH=/scratch/anonymous/USER
SLURM_TACC_NNODES_SET=1
SLURM_TACC_CORES=1
TACC_MKL_DIR=/opt/intel/compilers_and_libraries_2018
↪  .2.199/linux/mkl
FI_PSM2_LAZY_CONN=1
LMOD_FAMILY_MPI=impi
TACC_MPI_GETMODE=impi_hydra
I_MPI_ROOT=/opt/intel/compilers_and_libraries_2018.2
↪  .199/linux/mpi
TACC_QT5_INC=/opt/apps/qt5/5.11.2/include
MACHINE_NAME=stampede
BASH_FUNC_sbatch()=() {  echo -e "\nNOTIFICATION:
↪  sbatch not available on compute nodes. Use a login
↪  node.\n"
}
BASH_FUNC_module()=() {  if [ -z
↪  "${LMOD_SH_DBG_ON+x}" ]; then
 case "$-" in
 *v*x*)
 __lmod_sh_dbg='vx'
 ;;
 *v*)
 __lmod_sh_dbg='v'
 ;;
 *x*)
 __lmod_sh_dbg='x'
 ;;
 esac;
 fi;
 if [ -n "${__lmod_sh_dbg:-}" ]; then
 set +$__lmod_sh_dbg;
 echo "Shell debugging temporarily silenced: export
 ↪  LMOD_SH_DBG_ON=1 for Lmod's output";
 fi;
 eval $($LMOD_CMD bash "$@") && eval
 ↪  $(${LMOD_SETTARG_CMD:-:} -s sh);
 local _lmod_my_status=$?;
```

```
 if [ -n "${__lmod_sh_dbg:-}" ]; then
 echo "Shell debugging restarted";
 set -$__lmod_sh_dbg;
 unset __lmod_sh_dbg;
 fi;
 return $_lmod_my_status
}
BASH_FUNC_ml()=() {  eval $($LMOD_DIR/ml_cmd "$@")
}
+ lsb_release -a
LSB Version:    :core-4.1-amd64:core-4.1-noarch:cxx-
↪  4.1-amd64:cxx-4.1-noarch:desktop-4.1-amd64:deskt
↪  op-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
+ uname -a
Linux c455-061.stampede2.tacc.utexas.edu
↪  3.10.0-957.5.1.el7.x86_64 #1 SMP Fri Feb 1
↪  14:54:57 UTC 2019 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):                272
On-line CPU(s) list:   0-271
Thread(s) per core:    4
Core(s) per socket:    68
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 87
Model name:            Intel(R) Xeon Phi(TM) CPU 7250
↪  @ 1.40GHz
Stepping:              1
CPU MHz:               1494.165
CPU max MHz:           1600.0000
CPU min MHz:           1000.0000
BogoMIPS:              2793.29
L1d cache:             32K
L1i cache:             32K
L2 cache:              1024K
NUMA node0 CPU(s):     0-271
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 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 ring3mwait epb
↪  ibrs ibpb fsgsbase tsc_adjust bmi1 avx2 smep bmi2
↪  erms avx512f rdseed adx avx512pf avx512er avx512cd
↪  xsaveopt dtherm ida arat pln pts spec_ctrl
+ cat /proc/meminfo
MemTotal:       98698220 kB
MemFree:        88660500 kB
MemAvailable:   88227780 kB
Buffers:               0 kB
Cached:          1177244 kB
SwapCached:            0 kB
Active:           151664 kB
Inactive:        1147140 kB
Active(anon):     122740 kB
Inactive(anon):  1107700 kB
Active(file):      28924 kB
Inactive(file):    39440 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:             0 kB
SwapFree:              0 kB
Dirty:                 0 kB
Writeback:             0 kB
AnonPages:        121724 kB
Mapped:            74876 kB
Shmem:           1108868 kB
Slab:            4957208 kB
SReclaimable:     633376 kB
SUnreclaim:      4323832 kB
KernelStack:       43088 kB
PageTables:         8012 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:    90802360 kB
Committed_AS:    1521944 kB
VmallocTotal:   34359738367 kB
VmallocUsed:     1585460 kB
VmallocChunk:   34357935928 kB
HardwareCorrupted:     0 kB
AnonHugePages:     38912 kB
CmaTotal:              0 kB
CmaFree:               0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
DirectMap4k:     1181920 kB
```

```
DirectMap2M:    53245952 kB
DirectMap1G:    48234496 kB
+ inxi -F -c0
./collect_environment.sh: line 14: inxi: command not
↪  found
+ lsblk -a
NAME              MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda                 8:0    0 111.8G  0 disk
├─sda1              8:1    0     1M  0 part
├─sda2              8:2    0     1G  0 part /boot
└─sda3              8:3    0 110.8G  0 part
  ├─rootvg01-lv01 253:0    0    75G  0 lvm  /
  ├─rootvg01-tmp  253:1    0  31.8G  0 lvm  /tmp
  └─rootvg01-var  253:2    0     4G  0 lvm  /var
loop0               7:0    0          1 loop
loop1               7:1    0          1 loop
loop2               7:2    0          1 loop
loop3               7:3    0          1 loop
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
loop28              7:28   0          1 loop
loop29              7:29   0          1 loop
loop30              7:30   0          1 loop
loop31              7:31   0          1 loop
+ lsscsi -s
[4:0:0:0]    disk    ATA      INTEL SSDSC2BB12 0140
↪  /dev/sda    120GB
+ module list
+ '[' -z '' ']'
+ case "$-" in
+ __lmod_sh_dbg=x
+ '[' -n x ']'
+ set +x
Shell debugging temporarily silenced: export
↪  LMOD_SH_DBG_ON=1 for Lmod's output

Currently Loaded Modules:
  1) intel/18.0.2      3) impi/18.0.2   5)
  ↪  petsc/3.10-i64   7) qt5/5.11.2   9) p4est/2.0
  2) libfabric/1.7.0   4) fftw3/3.3.8   6) boost/1.68
  ↪       8) vtk/8.1.1



Shell debugging restarted
+ unset __lmod_sh_dbg
+ return 0
+ nvidia-smi
./collect_environment.sh: line 18: nvidia-smi:
↪  command not found
+ lshw -short -quiet -sanitize
+ cat
./collect_environment.sh: line 19: lshw: command not
↪  found
+ lspci
./collect_environment.sh: line 19: lspci: command not
↪  found
```