Massively Parallel Stencil Strategies for Radiation Transport Moment Model Simulations

Marco Berghoff[1(✉)], Martin Frank[1], and Benjamin Seibold[2]

[1] Steinbuch Centre for Computing, Karlsruhe Institute of Technology, Karlsruhe, Germany
{marco.berghoff,martin.frank}@kit.edu
[2] Department of Mathematics, Temple University, Philadelphia, PA 19122, USA
seibold@temple.edu

Abstract. The radiation transport equation is a mesoscopic equation in high dimensional phase space. Moment methods approximate it via a system of partial differential equations in traditional space-time. One challenge is the high computational intensity due to large vector sizes (1 600 components for P39) in each spatial grid point. In this work, we extend the calculable domain size in 3D simulations considerably, by implementing the StaRMAP methodology within the massively parallel HPC framework NAStJA, which is designed to use current supercomputers efficiently. We apply several optimization techniques, including a new memory layout and explicit SIMD vectorization. We showcase a simulation with 200 billion degrees of freedom, and argue how the implementations can be extended and used in many scientific domains.

Keywords: Radiation transport · Moment methods · Stencil code · Massively parallel

1 Introduction

The accurate computation of radiation transport is a key ingredient in many application problems, including astrophysics [28,35,44], nuclear engineering [8,9,32], climate science [24], nuclear medicine [20], and engineering [29]. A key challenge for solving the (energy-independent) radiation transport equation (RTE) (1) is that it is a mesoscopic equation in a phase space of dimension higher than the physical space coordinates. Moment methods provide a way to approximate the RTE via a system of macroscopic partial differential equations (PDEs) defined in traditional space-time. Here we consider the P_N method [7], which is based on an expansion of the solution of (1) in Spherical Harmonics. It can be interpreted as a moment method or, equivalently, as a spectral semi-discretization in the angular variable. Advantages of the P_N method over angular discretizations by collocation (discrete ordinates, S_N) [30] is that it preserves rotational invariance. A drawback, particular in comparison to nonlinear

© Springer Nature Switzerland AG 2020
V. V. Krzhizhanovskaya et al. (Eds.): ICCS 2020, LNCS 12143, pp. 242–256, 2020.
https://doi.org/10.1007/978-3-030-50436-6_18

moment methods [1,18,23,31,40,42], are spurious oscillations ("wave effects") due to Gibbs phenomena. To keep these at bay, it is crucial that the P_N method be implemented in a flexible fashion that preserves efficiency and scalability and allows large values of N.

Studies and applications of the P_N methods include [7,25,26,34]. An important tool for benchmarking and research on linear moment methods is the StaRMAP project [38], developed by two authors of this contribution. Based on a staggered grid stencil approach (see Sect. 2.1), the StaRMAP approach is implemented as an efficiently vectorized open-source MATLAB code [36]. The software's straightforward usability and flexibility have made it a popular research tool, used in particular in numerous dissertations, and it has been extended to other moment models (filtered [16] and simplified [33]), and applied in radiotherapy simulations [12,19]. For 2D problems, the vectorized MATLAB implementation allows for serial or shared memory (MATLAB's automatic usage of multiple cores) parallel execution speeds that are on par with comparable implementations of the methodology in C++. The purpose of this paper is to demonstrate that the same StaRMAP methodology also extends to large-scale, massively parallel computations and yields excellent scalability properties.

While S_N solvers for radiation transport are important production codes and major drivers for method development on supercomputers (one example is DENOVO [10], which is one of the most time-consuming codes that run in production mode on the Oak Ridge Leadership Computing Facility [27]), we are aware of only one work [14] that considers massively parallel implementations for moment models.

The enabler to transfer StaRMAP to current high-performance computing (HPC) systems is the open-source NAStJA framework [2,5], co-developed by one author of this contribution. NAStJA is a massively parallel framework for stencil-based algorithms on block-structured grids. The framework has been shown to efficiently scale up to more than ten thousand threads [2] and run simulations in several areas, using the phase-field method for water droplets [4], the phase-field crystal model for crystal–melt interfaces [15] and cellular Potts models for tissue growth and cancer simulations [6] with millions of grid points.

2 Model

The radiation transport equation (RTE) [8]

$$\begin{aligned} &\partial_t \psi(t,x,\Omega) + \Omega \cdot \nabla_x \psi(t,x,\Omega) + \Sigma_t(t,x)\psi(t,x,\Omega) \\ &\quad = \int_{S^2} \Sigma_s(t,x,\Omega\cdot\Omega')\psi(t,x,\Omega')\,\mathrm{d}\Omega' + q(t,x,\Omega), \end{aligned} \tag{1}$$

equipped with initial data $\psi(0,x,\Omega)$ and suitable boundary conditions, describes the evolution of the density ψ of particles undergoing scattering and absorption in a medium (units are chosen so that the speed of light $c = 1$). The phase space consists of time $t > 0$, position $x \in \mathbb{R}^3$, and flight direction $\Omega \in S^2$.

The medium is characterized by the cross-section Σ_t (see below) and scattering kernel Σ_s. Equation (1) stands representative for more general radiation problems, including electron and ion radiation [11] and energy-dependence [21].

Moment methods approximate (1) by a system of macroscopic equations. In 1D slab geometry, expand the Ω-dependence of ψ in a Fourier series, $\psi(t,x,\mu) = \sum_{\ell=0}^{\infty} \psi_\ell(t,x)\frac{2\ell+1}{2}P_\ell(\mu)$, where μ is the cosine of the angle between Ω and x-axis, and P_ℓ are the Legendre polynomials. Testing (1) with P_ℓ and integrating yields equations for the Fourier coefficients $\psi_\ell = \int_{-1}^{1} \psi P_\ell \,\mathrm{d}\mu$ as

$$\partial_t \psi_\ell + \partial_x \int_{-1}^{1} \mu P_\ell \psi \,\mathrm{d}\mu + \Sigma_{t\ell}\psi_\ell = q_\ell \qquad \text{for } \ell = 0, 1, \dots \,, \tag{2}$$

where $\Sigma_{t\ell} = \Sigma_t - \Sigma_{s\ell} = \Sigma_a + \Sigma_{s0} - \Sigma_{s\ell}$ and $\Sigma_{s\ell} = 2\pi \int_{-1}^{1} P_\ell(\mu)\Sigma_s(\mu)\,\mathrm{d}\mu$. Using the three-term recursion for Legendre polynomials, relation (2) becomes

$$\partial_t \psi_\ell + \partial_x \left(\tfrac{\ell+1}{2\ell+1}\psi_{\ell+1} + \tfrac{\ell}{2\ell+1}\psi_{\ell-1} \right) + \Sigma_{t\ell}\psi_\ell = q_\ell .$$

These equations can be assembled into an infinite system $\partial_t \boldsymbol{u} + M \cdot \partial_x \boldsymbol{u} + C \cdot \boldsymbol{u} = \boldsymbol{q}$, where $\boldsymbol{u} = (\psi_0, \psi_1, \dots)^T$ is the vector of moments, M is a tri-diagonal matrix with zero diagonal, and $C = \mathrm{diag}(\Sigma_{t0}, \Sigma_{t1}, \dots)$ is diagonal. The slab-geometry P_N equations are now obtained by omitting the dependence of ψ_N on ψ_{N+1} (alternative interpretations in [13,22,37]).

In 2D and 3D, there are multiple equivalent ways to define the P_N equations (cf. [7,38]). StaRMAP is based on the symmetric construction using the moments $\psi_\ell^m(t,x) = \int_{S^2} \overline{Y_\ell^m(\Omega)}\psi(t,x,\Omega)\,\mathrm{d}\Omega$, with the complex spherical harmonics $Y_\ell^m(\mu,\varphi) = (-1)^m \sqrt{\frac{2\ell+1}{4\pi}\frac{(\ell-m)!}{(\ell+m)!}}\, e^{im\varphi} P_\ell^m(\mu)$, where $\ell \geq 0$ is the moment order, and $-\ell \leq m \leq \ell$ the tensor components. Appropriate substitutions [38] lead to real-valued P_N equations. In 3D the moment system becomes

$$\partial_t \boldsymbol{u} + M_x \cdot \partial_x \boldsymbol{u} + M_y \cdot \partial_y \boldsymbol{u} + M_z \cdot \partial_z \boldsymbol{u} + C \cdot \boldsymbol{u} = \boldsymbol{q} \,, \tag{3}$$

where the symmetric system matrices M_x, M_y, M_z are sparse and possess a very special pattern of nonzero entries (see [36,38]). That coupling structure between unknowns (same in 2D and 1D) enables elegant and effective staggered grid discretizations upon which StaRMAP is based.

2.1 Numerical Methodology

We consider the moment system (3) in a rectangular computational domain $(0, L_x) \times (0, L_y) \times (0, L_z)$ with periodic boundary conditions (see below). The domain is divided into $n_x \times n_y \times n_z$ rectangular equi-sized cells of size $\Delta x \times \Delta y \times \Delta z$. The center points of these cells lie on the base grid

$$G_{111} = \left\{ \left(\left(i - \tfrac{1}{2}\right)\Delta x, \left(j - \tfrac{1}{2}\right)\Delta y, \left(k - \tfrac{1}{2}\right)\Delta z \right) \mid (1,1,1) \leq (i,j,k) \leq (n_x, n_y, n_z) \right\} .$$

The first component of $\boldsymbol{u}$ (the zeroth moment, which is the physically meaningful radiative intensity) is always placed on G_{111}. The other components of $\boldsymbol{u}$ are then

placed on the 7 other staggered grids $G_{211} = \{(i\Delta x, (j-1/2)\,\Delta y, (k-1/2)\,\Delta z)\}$, $G_{121} = \{((i-1/2)\,\Delta x, j\Delta y, (k-1/2)\,\Delta z)\}$, …, $G_{222} = \{(i\Delta x, j\Delta y, k\Delta z)\}$, following the fundamental principle that an x-derivative of a component in (3) that lives on a $(1, \bullet, \bullet)$ grid updates a component that lives on the corresponding $(2, \bullet, \bullet)$ grid. Likewise, x-derivatives of components on $(2, \bullet, \bullet)$ grids update information on the $(1, \bullet, \bullet)$ grids; and analogously for y- and z-derivative. A key result, proved in [38], is that this placement is, in fact, always possible.

Due to this construction, all spatial derivatives can be approximated via simple second-order half-grid centered finite difference stencils: two x-adjacent values, for instance living on the $(1, 1, 1)$ grid, generate the approximation

$$\partial_x u(i\Delta x, (j-\tfrac{1}{2})\Delta y, (k-\tfrac{1}{2})\Delta z) = \frac{u_{(i+\frac{1}{2}, j-\frac{1}{2}, k-\frac{1}{2})} - u_{(i-\frac{1}{2}, j-\frac{1}{2}, k-\frac{1}{2})}}{\Delta x} + O(\Delta x^2)$$

on the $(2, 1, 1)$ grid. We now call the G_{111}, G_{221}, G_{122}, and G_{212} grids "even", and the G_{211}, G_{121}, G_{112}, and G_{222} grids "odd".

The time-stepping of (3) is conducted via bootstrapping between the even and the odd grid variables. This is efficiently possible because of the approximate spatial derivatives of the even/odd grids update *only* the components that live on the odd/even grids. Those derivative components on the dual grids are considered "frozen" during a time-update of the other variables, leading to the decoupled update ODEs

$$\begin{cases} \partial_t \boldsymbol{u}^{\mathrm{e}} + C^{\mathrm{e}} \cdot \boldsymbol{u}^{\mathrm{e}} = \boldsymbol{q}^{\mathrm{e}} - (M_x^{\mathrm{eo}} \cdot D_x + M_y^{\mathrm{eo}} \cdot D_y + M_z^{\mathrm{eo}} \cdot D_z)\boldsymbol{u}^{\mathrm{o}} \\ \partial_t \boldsymbol{u}^{\mathrm{o}} + C^{\mathrm{o}} \cdot \boldsymbol{u}^{\mathrm{o}} = \boldsymbol{q}^{\mathrm{o}} - (M_x^{\mathrm{oe}} \cdot D_x + M_y^{\mathrm{oe}} \cdot D_y + M_z^{\mathrm{oe}} \cdot D_z)\boldsymbol{u}^{\mathrm{e}} \end{cases} \tag{4}$$

for the vector of even moments $\boldsymbol{u}^{\mathrm{e}}$ and the vector of odd moments $\boldsymbol{u}^{\mathrm{o}}$. In (4), the right-hand sides are constant in time (due to the freezing of the dual variables, as well as the source $\boldsymbol{q}$). Moreover, because C^{e} and C^{o} are diagonal, the equations in (4) decouple further into scalar ODEs of the form

$$\partial_t u_k(\boldsymbol{x}, t) + \bar{c}_k(\boldsymbol{x}) u_k(\boldsymbol{x}, t) = \bar{r}_k(\boldsymbol{x}),$$

whose exact solution is

$$u_k(\boldsymbol{x}, t + \Delta t) = u_k(\boldsymbol{x}, t) + \Delta t\,(\bar{r}_k(\boldsymbol{x}) - \bar{c}_k(\boldsymbol{x}) u_k(\boldsymbol{x}, t))\, E(-\bar{c}_k(\boldsymbol{x})\Delta t). \tag{5}$$

Here $\boldsymbol{x} = (x, y, z)$ is the spatial coordinate, and $E(c) = (\exp(c) - 1)/c$ (see [38] for a robust implementation of this function). To achieve second order in time, one full time-step (from t to $t + \Delta t$) is now conducted via a Strang splitting

$$\boldsymbol{u}(\boldsymbol{x}, t + \Delta t) = S^{\mathrm{o}}_{\frac{1}{2}\Delta t} \circ S^{\mathrm{e}}_{\Delta t} \circ S^{\mathrm{o}}_{\frac{1}{2}\Delta t} \boldsymbol{u}(\boldsymbol{x}, t), \tag{6}$$

where $S^{\mathrm{o}}_{\frac{1}{2}\Delta t}$ is the half-step update operator for the odd variables, and $S_\Delta t^{\mathrm{e}}$ the full-step update operator for the even variables, both defined via (5).

The convergence of this method, given that $\Delta t < \min\{\Delta x, \Delta y, \Delta z\}/3$, has been proven in [38]. Stability is generally given even for larger time-steps if scattering is present.

3 Implementation

In the following section, we present our implementation of the StaRMAP model and applied optimizations that are required to run on current HPC systems efficiently. NAStJA-StaRMAP v1.0 [3] is published under the Mozilla Public License 2.0 and the source-code is available at https://gitlab.com/nastja/starmap.

3.1 The NAStJA Framework

The StaRMAP methodology described above was implemented using the open-source NAStJA framework[1]. The framework was initially developed to explore non-collective communication strategies for simulations with a large number of MPI ranks, as will be used in exascale computing. It was developed in such a way that many multi-physics applications based on stencil algorithms can be efficiently implemented in a parallel way. The entire domain is build of blocks in a block-structured grid. These blocks are distributed over the MPI ranks. Inside each block, regular grids are allocated for the data fields. The blocks are extended with halo layers that hold a copy of the data from the neighboring blocks. This concept is flexible, so it can adaptively create blocks where the computing area moves. The regular structure within the blocks allows high-efficiency compute kernels, called sweeps. Every process holds information only about local and adjacent blocks. The framework is entirely written in modern C++ and makes use of template metaprogramming to achieve excellent performance without losing flexibility and usability. Sweeps and other actions are registered and executed by the processes in each time-step for their blocks so that functionality can be easily extended. Besides, sweeps can be replaced by optimized sweeps, making it easy to compare the optimized version with the initial one.

3.2 Optimizations

Starting with the 3D version of the MATLAB code of StaRMAP, the goal of this work was to develop a highly optimized and highly parallel code for future real-time simulations of radiation transports.

Basic Implementation. The first step was to port the MATLAB code to C++ into the NAStJA framework. Here we decide to use spatial coordinates (x, y, z) as the underlying memory layout. At each coordinate, the vector of moments $\boldsymbol{u}$ is stored. The sub-grids G_{111} to G_{222} are only considered during the calculation and are not saved separately. This means that the grid points on G_{111} and all staggered grid points are stored at the non-staggered (x, y, z)-coordinates. Thus it can be achieved that data that are needed for the update is close to each other in the memory. As for Eq. (4) described, all even components are used to calculate the odd components and vice versa. This layout also allows the usage of a relatively small stencil. The D3C7 stencil, which reads for three dimensions, the central data point and the first six direct neighbors, is sufficient.

[1] The MPL-2.0 source-code is available at https://gitlab.com/nastja/nastja.

For parallelization, we use NAStJA's block distribution and halo exchange mechanisms. The halo is one layer that holds a copy of the $\boldsymbol{u}$ vectors from the neighboring blocks. Since a D3C7 stencil is used, it is sufficient to exchange the six first neighboring sides. Figure 1 left shows the grid points in NAStJA's cells and the halo layer. For the implemented periodic boundary condition, we use this halo exchange to copy NAStJA-cells from one side to the opposite side, even if only half of the moments are needed to calculate the central differences.

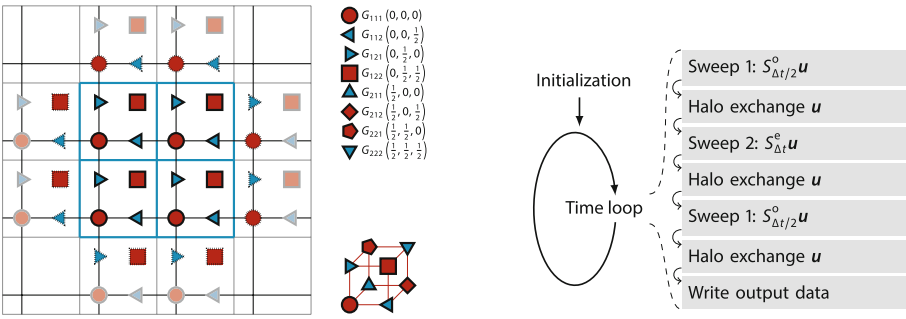

Fig. 1. Left: Staggered grids for the first z-layer. The odd coordinates are blue triangles and the even coordinates are marked by red shapes. The NAStJA-cells are blue squares. The base (111) grid is denoted by the black lines. The grid points with dotted border are the halo layer or the periodic boundary copy, the light grid points are not used. Center: 3D NAStJA-cell with the base grid point G_{111} (red circle) and the seven staggered grid points. Right: Action and sweep setup in NAStJA. (Color figure online)

For the calculation of the four substeps in Eq. (6), two different sweeps are implemented, each sweep swipes over the spatial domain in z, y, x order. The updates of each $\boldsymbol{u}$ component for each cell is calculated as followed. Beginning with the first substep, sweep S^{o} calculates $d_x\boldsymbol{u}, d_y\boldsymbol{u}, d_z\boldsymbol{u}$ of the even components as central differences, laying on the odd components. Then, the update of the odd components using this currently calculated $d_x\boldsymbol{u}, d_y\boldsymbol{u}, d_z\boldsymbol{u}$ is calculated. After the halo layer exchange, sweep S^{e} calculates the second substep. Therefore, first, the $d_x\boldsymbol{u}, d_y\boldsymbol{u}, d_z\boldsymbol{u}$ of the odd components are calculated, followed by the update of the even components. A second halo layer exchange proceeds before the sweep S^{o} is called again to complete with the third substep. The time-step is finalized by a third halo layer exchange and an optional output. Figure 1 right shows the whole sweep setup of one time-step in the NAStJA framework.

The time-independent parameter values as $\boldsymbol{q}$, $\bar{c}_k(\boldsymbol{x})$, and $E(-\bar{c}_k(\boldsymbol{x})\Delta t/2)$ are stored in an extra field on the non-staggered coordinates. Here, $\bar{c}_k(\boldsymbol{x})$ for $k \geq 1$ are identical. Their values on the staggered grid positions are interpolated.

Reorder Components. For optimization purposes, the calculation sweeps can easily exchange in NAStJA. Two new calculation sweeps are added for each of the following optimization steps. The computational instructions for the finite

differences of the components on one sub-grid are the same, as well as the interpolated parameter values. Components of the vector $\boldsymbol{u}$ are reordered, in that way that components of individual sub-grids are stored sequentially in memory. First, the even then, the odd sub-grid components follow, namely G_{111}, G_{221}, G_{212}, G_{122}, G_{211}, G_{121}, G_{112}, and G_{222}.

Unroll Multiplications. The calculation of $w = M_x \cdot d_x\boldsymbol{u} + M_y \cdot d_y\boldsymbol{u} + M_z \cdot d_z\boldsymbol{u}$ is optimized by manually unroll and skipping multiplication. The Matrices M_x and M_y have in each row one to four non-zero entries while the Matrix M_z has zero to two non-zero entries. Only these non-zero multiplication have to sum up to w. The first if-conditions for the non-zero entries in M_x and M_y is always true so that it can be skipped. A manual loop-unroll with ten multiplications and eight if-conditions is used.

SIMD Intrinsics. The automatic vectorization by the compilers results in worse run times. So we decide to manually instruct the code with intrinsics using the Advanced Vector Extensions 2 (AVX2), as supported by the test systems. Therefore, we reinterpret the four-dimensional data field (z, y, x, u) as a fifth-dimensional data field (z, y, X, u, x'), where x' holds the four x values that fit into the AVX vector register, and X is the x-dimension shrink by factor 4. Currently, we only support multiples of 4 for the x-dimension. The changed calculation sweeps allow calculating four neighbored values at once. The fact that the studied number of moments are multiples of 4 ensures that all the memory access are aligned. With this data layout, we keep the data very local and can still benefit from the vectorization.

4 HPC System

To perform the scaling test, we use a single node (kasper) and the high-performance computing systems ForHLR II, located at Karlsruhe Institute of Technology (fh2). The single node has two quad-core Intel Xeon processors E5-2623 v3 with Haswell architecture running at a base frequency of 3 GHz (2.7 GHz AVX), and have 4×256 KB of level 2 cache, and 10 MB of shared level 3 cache. The node has 54 GB main memory.

The ForHLR II has 1152 20-way Intel Xeon compute nodes [39]. Each of these nodes contains two deca-core Intel Xeon processors E5-2660 v3 with Haswell architecture running at a base frequency of 2.6 GHz (2.2 GHz AVX), and have 10×256 KB of level 2 cache, and 25 MB of shared level 3 cache. Each node has 64 GB main memory, and an FDR adapter to connect to the InfiniBand 4X EDR interconnect. In total, 256 nodes can be used, which are connected by a quasi fat-tree topology, with a bandwidth ratio of 10:11 between the switches and leaf switches. The leaf switches connect 23 nodes. The implementation of Open MPI in version 3.1 is used.

5 Results and Discussion

In this section, we present and discuss single core performance results as well as scaling experiments run on a high-performance computing system.

The presented performance results are measured in MLCUP/s, which stands for "million lattice cell component updates per second". This unit takes into account that the amount of data depends on the number of lattice cells and the number of moments.

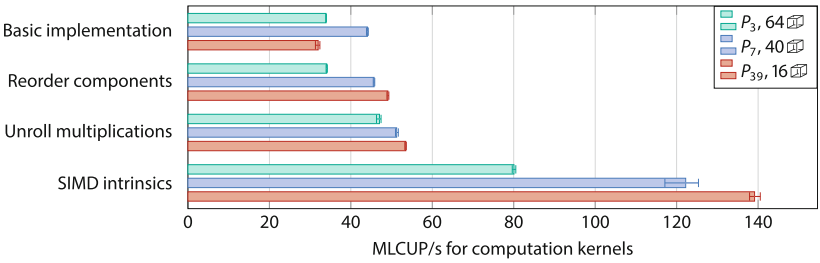

Fig. 2. Performance of the various optimization variants of the calculation sweeps running on a single core. The block size was chosen so that the number of the total components is approximately equal for all number of moments $M_\bullet$. The marks denote the average of three runs. The error bars indicate the minimum and maximum.

5.1 Performance Results

Single Core Performance. The starting point of our HPC implementation was a serial MATLAB code. A primary design goal of StaRMAP is to provide a general-purpose code with several different functions. In this application, we focus on specific cases, but let the number of moments be a parameter. A simple re-implementation in the NAStJA framework yields the same speed as the MATLAB code but has the potential to run in parallel and thus exceed the MATLAB implementation.

Figure 2 shows the performance of the optimization describes in Sect. 3.2. The measurements based on the total calculation sweep time per time-step, i.e., two sweep S^{o} + sweep S^{e}. In all the following simulations, we use cubic blocks, such that a block size of 40 refers to a cubic block with an edge length of 40 lattice cells without the halo. In legends, we write 40 . The speedup from the basic implementation to the reorder components version is small for P_3 and P_7 but significant for P_{39} (+54%). The number of components on each subgrid is small for the first both but large for P_{39}, so the overhead of the loops over all components becomes negligible. Unrolling brings an additional speedup of 38% for P_3, 14% for P_7, and 9% for P_{39}. Vectorization has the smallest effect for P_3 (+70%). For P_7 we gain +138% and +160% for P_{39}.

The combination of all optimizations results in a total speedup of factor 2.36, 2.77, 4.35 for P_3, P_7, P_{39}, respectively. This optimization enables us to simulate sufficiently large domains in a reasonable time to obtain physically meaningful results. Note, these results run with a single thread, so the full L3 cache is used.

Since the relative speedup does not indicate the utilization of a high-performance computing system, we have additionally analyzed the absolute performance of our code. In the following, we will concentrate on the single-node performance of our optimized code.

We show the performance analysis of the calculation sweeps on the single node kasper. First, we use the roofline performance model to categorize our code in the memory- or compute-bound region [43]. We use LIKWID [41] to measure the maximum attainable bandwidth. On kasper we reach a bandwidth of approximately 35 GiB/s, on one fh2 node we gain approximately 50 GiB/s. Since we are using a D3C7 stencil to swipe across the entire domain, four of the seven values to be loaded have already been loaded in the previous cell, so we can assume that only three values need to be loaded. The remaining data values are already in the cache, see Sect. 5.1 for details. The spatial data each holds the entire vector $\boldsymbol{u}$. For the interpolation of the time-independent parameter data, 130 Byte are not located in the cache and have to be loaded for on lattice update. The sweeps have to load 24 Byte per vector component. Remember that we need three sweeps to process one time-step, so an average of 94.5 Byte for P_3 are loaded per lattice component update, 77.6 Byte, 72.2 Byte for P_7, P_{39}, respectively. If we only consider the speak-performance on fh2 of 50 GiB/s $\cdot$ 72.2 Bytes/LCUP = 3 785 MLCUP/s and 2 527 MLCUP/s on kasper. That is far away from what we measured—an indication that we are operating on the compute-bound side. Counting the floating-point operations for one time-step, we get $392 + 40v_\mathrm{e} + 50v_\mathrm{o}$ FLOP, where v_e is the number of even and v_o the number of odd vector components.

So an average of 68.3 FLOP for P_3 are used per lattice component update, 50.5 FLOP, 45.1 FLOP for P_7, P_{39}, respectively. This results in an arithmetic intensity on the lower bound of 0.72 FLOP/Byte to 0.62 FLOP/Byte for P_3, P_{39}, respectively. The Haswell CPU in kasper has an AVX base frequency of 2.7 GHz [17] and can perform 16 floating-point operations with double precision per cycle. This results in 43.2 GFLOP/s per core. The achieved 139.1 MLCUP/s per core corresponds to 6.3 GFLOP/s and so to 15% peak-performance.

Cache Effects. Even if the analysis in the previous section shows that our application is compute-bound, it is worth taking a look at the cache behavior. Running large blocks will result in an excellent parallel scaling because of the computational time increase by $O(n^3)$ and the communication data only by $O(n^2)$.

To discover the cache behavior, we run 20 single jobs in parallel on one node on the fh2, this simulates the 2.5 MiB L3 cache per core. Figure 3 show the performance for different block sizes. For P_7, a maximum block size of 13 fits into the L2 cache, here the largest performance can be seen. At a block size of 35, the performance drops, which can be explained by the fact that with a maximum block size of 40, three layers fit into the L3 cache. For P_3, a block size of 20 still fits into the L2 cache, so here is the peak, up to a block size of 80 the performance remains almost constant after dropping firstly, this is the size

where the three layers fit into the L3 cache. The maximum for P_{39} is at a block size of 5, here the three layers fit into the L3 cache. We have not tested a smaller block size, because of the overhead of loops becomes too big. We will use the marked block sizes for the scaling analysis in the following sections. The block size of 20 was chosen so that all three moment orders can be compared here.

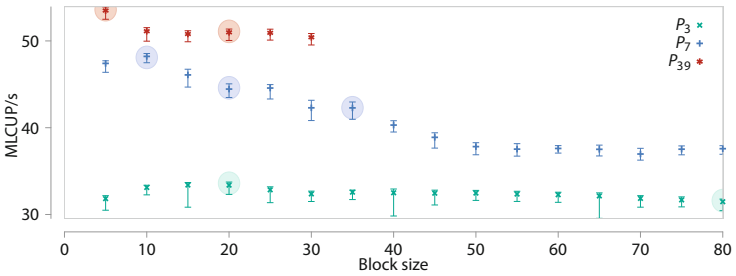

Fig. 3. Performance of the calculation sweeps for different block sizes.

Scaling Results. To examine the parallel scalability of our application, we consider weak scaling for different block sizes. During one run, each process gets a block of the same size. So we gain accurate scaling data that does not depend on any cache effects described in Sect. 5.1. First, we look at one node of the fh2, and then at the performance across multiple nodes, with each node running 20 processes at the 20 cores. We use up to 256 nodes, which are 5 120 cores. Figure 4(a) shows P_7-runs with different block sizes, where the MPI processes distributed equally over the two sockets. All three block sizes show similar, well-scaling behavior. Moreover, the whole node does not reach the bandwidth limit of 3 785 MLCUP/s, which confirms that the application is on the compute-bound side.

Before conducting scaling experiments, we evaluate the various parts of the application. Therefore, we show the amount of used calculation and communication time in Fig. 4(b). The calculation time for one time-step consists of the time used by two sweeps S^{o} and one sweep S^{e}. The communication time sums up the time used for the three halo exchanges. A high communication effort of about 50% is necessary. This proportion rarely changes for different vector lengths.

Figure 5 shows the parallel scalability of the application for different vector lengths and block sizes. The results of runs with one node are used as the basis for the efficiency calculations. In (a) three regimes are identifiable, $P_3, 80$ ⧈ and $P_{39}, 20$ ⧈ are more expensive and take a long time. $P_7, 35$ ⧈ is in the middle, and the remainder takes only a short average time per time-step. As expected, this is also reflected in the efficiency in (b). The expensive tasks scale slightly better with approximately 80% efficiency on 256 nodes, 5120 cores. The shorter tasks still have approximately 60% efficiency. From one to two nodes, there is a drop in some jobs; the required inter-node MPI communication can explain this.

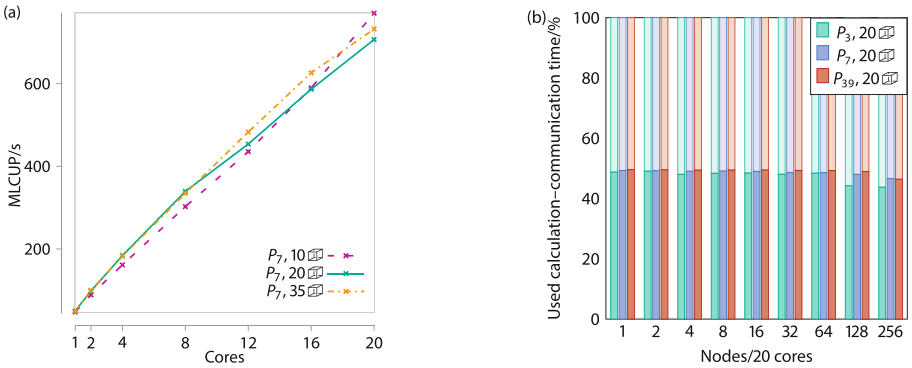

Fig. 4. (a) Single Node scaling on fh2. (b) Calculation time (dark) versus communication time (light).

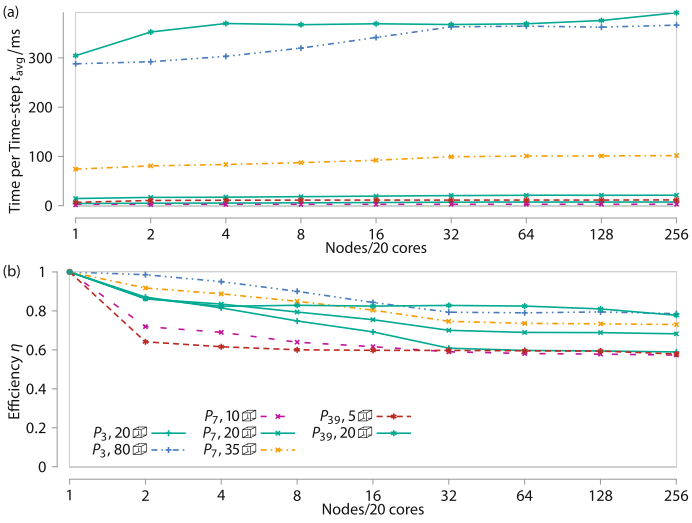

Fig. 5. MPI scaling (a) average time per time-step and (b) efficiency on fh2 for up to 5 120 cores.

From 32 nodes, the efficiency of all sizes is almost constant. This is because a maximum of 23 nodes is connected to one switch, i.e., the jobs must communicate via an additional switch layer. For runs on two to 16 nodes, the job scheduler can distribute the job to nodes connected to one switch but does not have to.

5.2 Simulation Results

With the parallelizability and scalability of the methodology and implementation established, we now showcase its applicability in a representative test example. We consider a cube geometry that resembles radiation transport (albeit with

simplified physics) in a nuclear reactor vessel, consisting of a reactor core with fuel rods, each 1 cm (5 grid-points) thick, surrounded by water (inner box in Fig. 6, and concrete (outer box). The non-dimensional material parameters are: source $q_0 = 2$, absorption $\Sigma_a^w = 10, \Sigma_a^c = 5$, scattering $\Sigma_s = 1$. The spatial resolution of the rod geometry and surrounding has a grid size of 500 , which we compute on up to 2000 cores via moment resolutions P_3, P_7, P_{19}, P_{29}, and P_{39}, depicted in Fig. 6 right. As one can see by comparing $P_N, N \geq 19$, the P_{19} simulation is well-resolved.

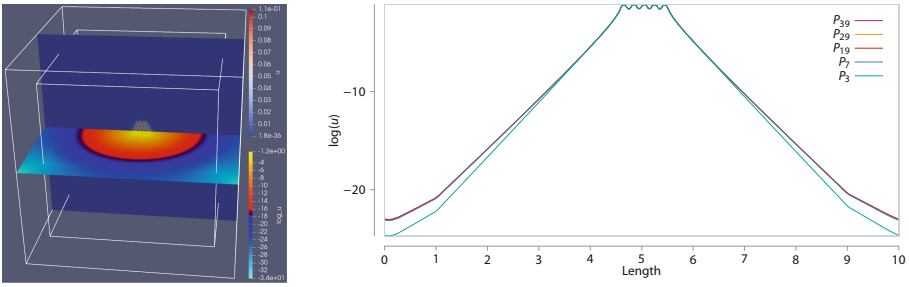

Fig. 6. Left: Rod geometry surrounded by water and concrete. The vertical slice shows u and the plane $\log_{10}(u)$. Right: Plot of the intensity $\log_{10}(u)$ over the section.

6 Conclusion

We have developed and evaluated a massively parallel simulation code for radiation transport based on a moment model, which runs efficiently on current HPC systems. With this code, we show that large domain sizes are now available. Therefore, an HPC implementation is of crucial importance. Starting from the reference implementation of StaRMAP in MATLAB, we have developed a new, highly optimized implementation that can efficiently run on modern HPC systems. We have applied optimizations at various levels to the highly complex stencil code, including explicit SIMD vectorization. Systematic performance engineering at the node-level resulted in a speedup factor of 4.35 compared to the original code and 15% of peak performance at the node-level. Besides, we have shown excellent scaling results for our code.

Acknowledgments. This work was performed on the supercomputer ForHLR funded by the Ministry of Science, Research and the Arts Baden-Württemberg and by the Federal Ministry of Education and Research. B. Seibold wishes to acknowledge support by the National Science Foundation through grant DMS–1719640.

References

1. Anile, A.M., Pennisi, S., Sammartino, M.: A thermodynamical approach to Eddington factors. J. Math. Phys. **32**, 544–550 (1991)
2. Berghoff, M., Kondov, I., Hötzer, J.: Massively parallel stencil code solver with autonomous adaptive block distribution. IEEE Trans. Parallel Distrib. Syst. **29**, 2282–2296 (2018)
3. Berghoff, M., Frank, M., Seibold, B.: StaRMAP - A NAStJA Application (2020). https://doi.org/10.5281/zenodo.3741415
4. Berghoff, M., Kondov, I.: Non-collective scalable global network based on local communications. In: 2018 IEEE/ACM 9th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (scalA), pp. 25–32. IEEE (2018)
5. Berghoff, M., Rosenbauer, J., Pfisterer, N.: The NAStJA Framework (2020). https://doi.org/10.5281/zenodo.3740079
6. Berghoff, M., Rosenbauer, J., Schug, A.: Massively parallel large-scale multi-model simulation of tumor development (2019)
7. Brunner, T.A., Holloway, J.P.: Two-dimensional time dependent Riemann solvers for neutron transport. J. Comput. Phys. **210**(1), 386–399 (2005)
8. Case, K.M., Zweifel, P.F.: Linear Transport Theory. Addison-Wesley, Boston (1967)
9. Davison, B.: Neutron Transport Theory. Oxford University Press, Oxford (1958)
10. Evans, T.M., Stafford, A.S., Slaybaugh, R.N., Clarno, K.T.: Denovo: a new three-dimensional parallel discrete ordinates code in SCALE. Nuclear Technol. **171**(2), 171–200 (2010). https://doi.org/10.13182/NT171-171
11. Frank, M., Herty, M., Schäfer, M.: Optimal treatment planning in radiotherapy based on Boltzmann transport calculations. Math. Mod. Meth. Appl. Sci. **18**, 573–592 (2008)
12. Frank, M., Küpper, K., Seibold, B.: StaRMAP – a second order staggered grid method for radiative transfer: application in radiotherapy. In: Sundar, S. (ed.) Advances in PDE Modeling and Computation, pp. 69–79. Ane Books Pvt. Ltd. (2014)
13. Frank, M., Seibold, B.: Optimal prediction for radiative transfer: a new perspective on moment closure. Kinet. Relat. Models **4**(3), 717–733 (2011). https://doi.org/10.3934/krm.2011.4.717
14. Garrett, C.K., Hauck, C., Hill, J.: Optimization and large scale computation of an entropy-based moment closure. J. Comput. Phys. **302**, 573–590 (2015). https://doi.org/10.1016/j.jcp.2015.09.008
15. Guerdane, M., Berghoff, M.: Crystal-melt interface mobility in bcc Fe: linking molecular dynamics to phase-field and phase-field crystal modeling. Phys. Rev. B **97**(14), 144105 (2018)
16. Hauck, C.D., McClarren, R.G.: Positive P_N closures. SIAM J. Sci. Comput. **32**(5), 2603–2626 (2010)
17. Intel Corporation: Intel Xeon Processor E5 v3 product family: specification update. Technical report 330785–011, Intel Corporation (2017)
18. Kershaw, D.S.: Flux limiting nature's own way. Technical report UCRL-78378, Lawrence Livermore National Laboratory (1976)
19. Küpper, K.: Models, Numerical Methods, and Uncertainty Quantification for Radiation Therapy. Dissertation, Department of Mathematics, RWTH Aachen University (2016)

20. Larsen, E.W.: Tutorial: the nature of transport calculations used in radiation oncology. Transp. Theory Stat. Phys. **26**, 739 (1997)
21. Larsen, E.W., Miften, M.M., Fraass, B.A., Bruinvis, I.A.D.: Electron dose calculations using the method of moments. Med. Phys. **24**, 111–125 (1997)
22. Larsen, E.W., Morel, J.E., McGhee, J.M.: Asymptotic derivation of the multigroup P_1 and simplified P_N equations with anisotropic scattering. Nucl. Sci. Eng. **123**, 328–342 (1996)
23. Levermore, C.D.: Relating Eddington factors to flux limiters. J. Quant. Spectrosc. Radiat. Transfer **31**, 149–160 (1984)
24. Marshak, A., Davis, A.: 3D Radiative Transfer in Cloudy Atmospheres. Springer, Heidelberg (2005). https://doi.org/10.1007/3-540-28519-9
25. McClarren, R.G., Evans, T.M., Lowrie, R.B., Densmore, J.D.: Semi-implicit time integration for P_N thermal radiative transfer. J. Comput. Phys. **227**(16), 7561–7586 (2008)
26. McClarren, R.G., Holloway, J.P., Brunner, T.A.: On solutions to the P_n equations for thermal radiative transfer. J. Comput. Phys. **227**(3), 2864–2885 (2008)
27. Messer, O.B., D'Azevedo, E., Hill, J., Joubert, W., Berrill, M., Zimmer, C.: MiniApps derived from production HPC applications using multiple programing models. Int. J. High Perform. Comput. Appl. **32**(4), 582–593 (2018). https://doi.org/10.1177/1094342016668241
28. Mihalas, D., Weibel-Mihalas, B.: Foundations of Radiation Hydrodynamics. Dover (1999)
29. Modest, M.F.: Radiative Heat Transfer, 2nd edn. Academic Press (1993)
30. Morel, J.E., Wareing, T.A., Lowrie, R.B., Parsons, D.K.: Analysis of ray-effect mitigation techniques. Nuclear Sci. Eng. **144**, 1–22 (2003)
31. Müller, I., Ruggeri, T.: Rational Extended Thermodynamics, 2nd edn. Springer, New York (1993). https://doi.org/10.1007/978-1-4612-2210-1
32. Murray, R.L.: Nuclear Reactor Physics. Prentice Hall (1957)
33. Olbrant, E., Larsen, E.W., Frank, M., Seibold, B.: Asymptotic derivation and numerical investigation of time-dependent simplified P_N equations. J. Comput. Phys. **238**, 315–336 (2013)
34. Olson, G.L.: Second-order time evolution of P_N equations for radiation transport. J. Comput. Phys. **228**(8), 3072–3083 (2009)
35. Pomraning, G.C.: The Equations of Radiation Hydrodynamics. Pergamon Press (1973)
36. Seibold, B., Frank, M.: StaRMAP code. http://www.math.temple.edu/~seibold/research/starmap
37. Seibold, B., Frank, M.: Optimal prediction for moment models: crescendo diffusion and reordered equations. Contin. Mech. Thermodyn. **21**(6), 511–527 (2009). https://doi.org/10.1007/s00161-009-0111-7
38. Seibold, B., Frank, M.: StaRMAP - a second order staggered grid method for spherical harmonics moment equations of radiative transfer. ACM Trans. Math. Softw. **41**(1), 4:1–4:28 (2014)
39. Steinbuch Centre for Computing: Forschungshochleistungsrechner ForHLR II. https://www.scc.kit.edu/dienste/forhlr2.php
40. Su, B.: Variable Eddington factors and flux limiters in radiative transfer. Nucl. Sci. Eng. **137**, 281–297 (2001)
41. Treibig, J., Hager, G., Wellein, G.: LIKWID: a lightweight performance-oriented tool suite for x86 multicore environments. In: Proceedings of PSTI 2010, the First International Workshop on Parallel Software Tools and Tool Infrastructures, San Diego, CA (2010)

42. Turpault, R., Frank, M., Dubroca, B., Klar, A.: Multigroup half space moment approximations to the radiative heat transfer equations. J. Comput. Phys. **198**, 363–371 (2004)
43. Williams, S., Waterman, A., Patterson, D.: Roofline: an insightful visual performance model for multicore architectures. Commun. ACM **52**(4), 65–76 (2009)
44. Zeldovich, Y., Raizer, Y.P.: Physics of Shock Waves and High Temperature Hydrodynamic Phenomena. Academic Press (1966)