

Exploiting Multilevel Parallelism in Radiation Transfer Methods for Astrophysics Simulations *

Gabriele Jost
ARC-TNE

NASA Ames Research Center
Moffet Field, CA CA 94035-0001, USA
Gabriele.Jost@nasa.gov

Yan-Fei Jiang

Kavli Institute for Theoretical Physics,
University of California,
Santa Barbara, CA 93106, USA
yanfei@kitp.ucsb.edu

Abstract—Radiation is an important factor for determining thermal properties of astrophysical systems, such as star formation or the evolution of galaxies. In this paper we discuss the performance characteristics of an enhanced version of the astrophysics code Athena++. The code employs a direct solver for the radiation transport equation, which is accurate and flexible. It exploits several levels of parallelism for efficient execution on modern multi- and many-core processors. We discuss the radiation transport method and its parallelization. Then we show scalability and performance analysis results to demonstrate the impact of various parallelization strategies on the performance. Results are presented for Intel Xeon and Xeon Phi (KNL) nodes.

Index Terms—Astrophysical Systems, Radiation Transport, Parallelization, Vectorization, Performance Analysis

I. INTRODUCTION

The purpose of this technical report is really twofold: Foremost we want to discuss and report on the performance of the Athena++ code. In addition, we hope that presented methods might help other scientific code developers to optimize their applications for modern multicore processors. Radiation is the key to determine the thermal properties of many astrophysical systems, such as the formation of stars in various environments [1, 2, 3], cosmological structure formation [4], disruption of massive giant molecular clouds [5], and evolution of galaxies with feedback from active galactic nuclei [6, 7], to name but a few. For cases where radiation pressure is the dominant force, for example in the inner region of black hole accretion disks [8], photons control the dynamics of these systems. Therefore, calculating the propagation of photons and the interactions between gas and the radiation field accurately is critical to understand the properties of these systems.

In principle, photons are described by the specific intensity I , which should be evolved in numerical simulations by solving the radiative transfer equation as [9]

$$\frac{\partial I}{\partial t} + \mathbf{n} \cdot \nabla I = \sigma_t (S - I), \quad (1)$$

where σ_t is the total opacity and S is the source function. This equation is very challenging to solve numerically compared with the classical magneto-hydrodynamic equations because of the dimensionality of the problem. The specific intensity is not

only a function of time and space, but also a function of angles and frequency, which significantly increases the computational cost.

For astrophysical systems where the typical dynamic time scale is much longer than light crossing time, it is usually too expensive to solve the above radiative transfer equation directly. This equation is then usually integrated over angles and frequencies to get the frequency independent radiation moment equations, which are used to evolve the radiation energy density and flux. The radiation moment equations are easier to solve implicitly or explicitly with reduced speed of light approximation [10, 11]. However, they need a closure relation, which is typically implemented by specifying the Eddington tensor to relate the radiation pressure to the radiation energy density. When the time variation of the Eddington tensor is small in each hydrodynamic time step, the *time independent* radiative transfer equation can be solved with short characteristics to get a snapshot of the variable Eddington tensor (VET), which is then used to close the radiation moment equations. This approach has been adopted in various numerical codes [12, 13, 14, 15, 16].

When solving the time independent radiative transfer equation is too expensive, various assumptions are made to approximate the Eddington tensor or the radiation flux. The most commonly used method has been flux-limited diffusion (FLD) [17], which does not evolve the radiation flux but instead assumes it can be specified by a diffusive approximation. Another increasingly popular alternative is the M1 method [18, 19], which solves an evolution equation for the radiation flux, but specifies an approximate form for the Eddington tensor. In both methods, the Eddington tensor is specified in terms of the local radiation energy density and flux. The flux limiter in FLD, which parameterizes departures from the diffusive behavior, is also determined by local values of the opacity and radiation energy density. Because numerical algorithms based on these two schemes are easy to implement and have been shown to be accurate in the diffusion limit, they have been widely used for various astrophysical systems [e.g., 20, 21, 22, 23, 24, 25, 11, 26, 27].

However, these assumptions are usually designed to work in one extreme limit, while the dynamics of radiating fluids change dramatically when the radiation energy density, mo-

This work was supported by the NASA Supercomputing Support Services (NS3) contract, NNA07CA29C

mentum, as well as the optical depth across a typical size change [9, 28]. In the regimes where these assumptions are not valid, it is unclear how these assumptions will affect the dynamics of the fluid, especially in multi-dimensions where the angular distribution of the radiation field is important and poorly constrained by these assumptions. For example, [29] shows that when the impact of the radiation forces on the dusty atmosphere of ultra-luminous infrared galaxies is studied, simulations performed with VET lead to different conclusions than simulations performed with FLD. This example demonstrates that in the semi-transparent regime, where diffusion approximation fails, simulations based on this assumption can be misleading. In real astrophysical systems the photosphere is crucial for the observational appearance and cooling, so it is essential to model the transition from optically thick to optically thin regions accurately.

There are several numerical algorithms that can solve equation (1) directly without ad-hoc assumptions. One popular approach is the Monte Carlo Method [30, and references therein], which is able to handle various radiative processes accurately. Several novel approaches have also been developed to reduce the noise and computational cost of this method [31, 32, 33]. The co-author of this report has developed a finite volume approach to solve equation 1 for specific intensities along discrete angles [34]. This algorithm is accurate, flexible and efficient to model various astrophysical systems in 3D, although the simulations are still compute expensive. It is implemented in the publicly available package Athena++¹ and has been successfully used to study the thermal stability in radiation pressure dominated black hole accretion disks [35], structures of black hole accretion disks in super-Eddington regimes [36], stability and structures of accretion disks around supermassive black holes [37], prompt emission in tidal disruption events [38], structures and energy transport at the envelopes of massive stars around iron opacity peaks [39, 40].

Given the value of this code for our understanding of astrophysical systems, we want to make sure it takes full advantage of vectorization and parallelization available in modern processors. In addition to this, we are experimenting with using multi-threading to exploit yet a third level of parallelism. The rest of the paper is structured as follows. In Section II we describe the challenges that have to be addressed when implementing radiation transport methods. In Section III we discuss the implementation of the algorithms, addressing issues such as communication, vectorization and multi-threading. Performance results and analysis for Xeon and Xeon Phi (KNL) clusters are presented in Section IV and in Section V we draw our conclusion and outline future work.

II. CHALLENGES TO SOLVE THE TIME DEPENDENT RADIATIVE TRANSFER EQUATIONS

Typical MHD simulations need to evolve 8 independent variables in time and space while we typically need to solve 80 specific intensities for the same time and space in order

to resolve the angular distribution of the radiation field accurately. If the computational cost to solve one independent variable is the same for both MHD and radiative transfer parts, the radiative transfer simulations will be slower at least by a factor of 10. In relativistic systems, the CFL condition is similar to MHD. In non-relativistic systems, it is typically smaller by a factor of about 10. The memory requirement is also significantly increased, which can also be a challenge to get the best performance.

There are basically two steps when we solve equation 1. The first step is the left hand side of the equation, which describes the transport of specific intensities. Each specific intensity is independent during this step, which should be an ideal target for vectorization. However, the transport step requires specific intensities from neighboring grid zones, which are usually not continuous in memory particularly in 3D. Getting the best cache performance will be crucial. The second step is the right hand side of equation 1, which describes the interactions between different specific intensities at the same spatial locations. This term couples all the specific intensities but does not require data from neighboring grid zones. Therefore, the two terms have exactly the opposite properties and we want to get the best parallelization performance for both of them.

III. IMPLEMENTATION OF THE ALGORITHM

Athena++ is a general-purpose radiation magneto-hydrodynamic (MHD) code which is widely used to study a wide range of astrophysical problems. It is a grid-based code solving Euler's Equation. It uses a high order Godunov shock capturing scheme, which was developed primarily for studies of the interstellar medium, star formation, and accretion flows.

A. Communication

The algorithm employs Static Mesh Refinement (SMR) and is parallelized using the MPI [42] message passing interface. The SMR criteria is basically the density, more refinement is considered near the disk midplane where most of the turbulence is. To implement the MPI parallelization, the computational volume is organized into a hierarchy of *Grids*, *Domains*, and *Meshes*. Parallelization is achieved via classical domain decomposition. The 3D domain is divided into cubes. The cubes are distributed among the MPI ranks. The algorithm requires the exchange of the 6 cube faces between neighboring ranks. The complete hierarchy, including all the Domains and Grids in the calculation, is called the Mesh. The major data structure is a *mesh-block*. A mesh-block is the typical working unit per MPI rank, however, it is possible to assign multiple mesh blocks to a rank. A mesh block will typically contain 16x16x16 grids. Each grid typically around 100 variables and it is the smallest volume where the conserved variables are stored and updated. Communication in Athena++ can be characterized as fine-grained and task based. If neighbor data is not available, tasks which do not require the data can be performed in the meantime. It employs asynchronous communication implemented via *MPI_Test*, *MPI_Isend* and

¹<http://princetonuniversity.github.io/athena/>

MPI_Irecv. This allows overlapping communication and computation. For further optimization of MPI communication, Athena++ takes advantage of persistent communication. In loops implementing the exchange of boundary values, we do encounter situations, where MPI calls with the same communication arguments are repeatedly executed. In these cases it is advantageous to bind the communication arguments to a persistent communication request once, then, repeatedly using the request to initiate the full message. This is implemented using *MPI_Start*, *MPI_Send_init*, *MPI_Recv_init*. An example is shown in the listing below:

```
// Create the persistent receive request
MPI_Recv_init(hydro_recv_[nb.bufile], rsize,
             MPI_ATHENA_REAL,
             nb.rank, tag, MPI_COMM_WORLD,
             &req_hydro_recv_[nb.bufile]);
....
// Start the receive of the message
void BoundaryValues::StartReceiving(bool field)
{
  MeshBlock *pmb=pmy_block_;
  for(int n=0;n<pmb->nneighbor;n++) {
    NeighborBlock& nb = pmb->nneighbor[n];
    if(nb.rank!=Globals::my_rank) {
      if (field) { // normal case
        MPI_Start(&req_hydro_recv_[nb.bufile]);
        if (MAGNETIC_FIELDS_ENABLED)
          MPI_Start(&req_field_recv_[nb.bufile]);
      } } }
  return;
}
```

The MPI scalability of the Athena++ is discussed in Section IV-B.

B. Computation

Modern processors provide instructions that can operate on multiple data and return multiple results. They are referred to as SIMD (Single Instruction Multiple Data). In SIMD mode, multiple elements are packed into wide registers to be processed all at once. The width of the registers has been increasing with more advanced processor types. If the code is not vectorized then the full capability of the floating point unit is not used. Compilers will try to generate SIMD instructions whenever possible, but it is a challenging task due to compile-time unknown factors and code complexities. Compiler vendors have provided vendor specific pragmas that allow the user to provide hints to the compiler to ease the challenge of exploiting SIMD instructions. Using compiler pragmas was necessary to increase vectorization in Athena++. Our goal was to increase the vectorization ratio, but not depend on vendor specific pragmas. Instead we used the OpenMP [41] shared memory parallelization API. OpenMP 4 provides support for instruction level parallelism via the *simd* construct. Rather than hints, these pragmas are commands to the compiler and will override all compiler analysis. It is therefore very important to clearly understand data dependences within the code and also declare variables as private to a vectorized loop if necessary. Athena++ is implemented employing C++ object oriented programming methods, which supports modularity. However, the code is also targeted to exploit SIMD Vectorization employing stride 1 access as much as possible in time consuming loops. Athena++ data structures are organized as SOAs (Structures

of Arrays) which facilitates traversing computational loops with stride 1. The loop below shows a typical loop from the radiation transfer code. In most cases the compiler was able to vectorize the loops, without the need for pragmas. In the code below, the compiler assumed false dependences between the *prad* data structure and the temporary array *temp_i2* because of potential overlap of the data structures. We had to employ the OpenMP *simd* pragma to enforce vectorization. Note that variables *adv_coef* and *vdotn* are private to the loop, since they are declared within the loop.

```
#pragma omp simd
for(int n=0; n<nang; ++n){
  ....
  Real vdotn =
  vx*prad->mu(0,k,j,i,n)
  +vy*prad->mu(1,k,j,i,n)
  + vz*prad->mu(2,k,j,i,n);
  vdotn *= invcrat;
  coef_tmp [n] = slorzsq * (1.0-vdotn);
  Real adv_coef = tau_fact *
  vdotn * (3.0 + vdotn * vdotn)
  ir(k,j,i,n+iffr*nang) * (1.0 - adv_coef);
  temp_i2(k,j,i,n+iffr*nang) = adv_coef;
  ...
}
```

The code snippet in the listing below shows an Athena++ loop calculating a global sum. It can be vectorized by declaring *er0* as a *reduction* variable.

```
Real er0 = 0.0;

for(int ifr=0; ifr<nfreq; ++ifr){
  #pragma omp simd reduction (+:er0)
  for(int n=0; n<nang; ++n){
    Real ir_weight = lab_ir[n+iffr*prad->nang];
    ...
    er0 += ir_weight;
    ..
  }
  er0 *= prad->wfreq(ifr);
}
```

Employing these techniques makes the code highly vectorized. The performance impact shall be discussed in Section IV-D.

C. Multithreading

Currently, Athena++ simulations are using very small mesh blocks in order to be able to use more than 10K cores to reduce the simulation time. This is not ideal as it increases the face to volume ratio. In this report we focus on strong scaling. In the past we conducted weak scaling experiments. The current weak scaling efficiency is about 90-95% with 10K cores. Using OpenMP, larger mesh blocks could be used to increase the weak scaling efficiency to 99%, for example. The current production runs require the use of about 8x8x16 cells per MPI rank in order to reduce wall clock time, because there is a lot of work per cell. Therefore the number of cells inside the mesh block is not significantly larger than the number of cells on the six surfaces. If we can increase the block size to 64x64x64 per MPI rank, then the amount of communication between MPI surfaces will be really tiny compared with the calculations inside each mesh block. We have started to experiment with inserting OpenMP pragmas in computationally intensive subroutines. The development of the

OpenMP parallelization is still at an early stage but we shall briefly outline the general approach. We focused on a subset of the most time consuming routines. The loops in these routines traverse 5 dimensional arrays. We place OpenMP *parallel* and *for* constructs on the outermost loops, in order to provide sufficient work for the individual threads. In addition we place a *simd* construct on the inner loop to exploit instruction level parallelism. This is demonstrated in the listing below. Note that we declare the arrays *x1area*, *vol*, and *dflx* within the parallel region. They will therefore be private to the individual threads.

```
#pragma omp parallel
{
  tid=omp_get_thread_num();
  AthenaArray<Real> x1area, vol, dflx;
  x1area.InitWithShallowSlice(x1face_area_, 2, tid, 1);
  vol.InitWithShallowSlice(cell_volume_, 2, tid, 1);
  dflx.InitWithShallowSlice(flx_, 3, tid, 1);
#pragma omp for schedule(static)
  for (int k=ks; k<=ke; ++k) {
    for (int j=js; j<=je; ++j) {
      // calculate x1-flux divergence
      pmb->pcoord->Face1Area(k, j, is, ie+1, x1area);
      for(int i=is; i<=ie; ++i){
#pragma omp simd
        for(int n=0; n<prad->n_fre_ang; ++n){
          dflx(i, n) = (x1area(i+1) *x1flux(k, j, i+1, n)
            - x1area(i)*x1flux(k, j, i, n)); }

```

The impact of OpenMP parallelization on performance shall be discussed in Section IV-C.

IV. RADIATION TRANSFER METHOD PERFORMANCE

In this section we shall present timings and discuss the performance.

A. Evaluation Environment

We ran our experiments on 2 different systems, both of them are located at the NASA Ames Research Center at Moffett Field, CA. Our Xeon based studies ran on the NASA Ames Research Center Pleiades cluster employing Xeon Broadwell nodes. Each Pleiades Broadwell rack contains 72 nodes; each node contains two 14-core E5-2680v4 (2.4 GHz) processors and 128 GB of memory, providing approximately 4.6 GB per core. The Broadwell nodes are connected to the Pleiades InfiniBand network (ib0 and ib1) via four-lane Fourteen Data Rate (4X FDR) devices and switches for internode communication. The other system is a cluster of Xeon Phi nodes, also located at NASA Ames Research Center. It has 20 nodes of Intel(R) Xeon Phi(TM) CPU 7230 running at 1.3GHz. Each node has 64 CPU cores run as 256 Hyper-Threaded logical cores. Each physical core has two 512-bit VPU. There are 16GB of high bandwidth MCDRAM and 192GB DDR4 memory on each node. While we were writing this report, Skylake nodes were added to the NASA Ames Research Center Electra cluster. We have added some preliminary results for this system as well. Electra now includes 1,152 Skylake nodes, which are partitioned into eight physical racks. Each node contains two 20-core Xeon Gold 6148 sockets (2.4 GHz) and 192 GB of memory. The Skylake processors include the Advanced Vector Extensions 512 (AVX-512).

For our study we used the Intel (R) compiler, 2018.0.128 and SGI MPI version 2.17 which is part of SGI's Message

Passing Toolkit (MPT) . On the KNL cluster we used Intel MPI version 2017 update 2.

For the Xeon Broadwell nodes we used the flags:

```
icc -O3 -qopenmp-simd
```

For the Xeon Phi we used the same compiler version with the flags:

```
icc -xmic-avx512 -O3 -qopenmp-simd
```

B. MPI Scalability and Computational Performance on Xeon Broadwell Nodes

For our tests we ran experiments on 3 different sets of input data, which are simulating accretion disks around black holes as studied in [36, 40]. The 3 data sets differ in the spatial resolution we use. Higher spatial resolution requires more MPI ranks and the actual MPI ranks we use are

- AngularDisk448 is case suitable for up to 448 ranks
- AngularDisk1792 is suitable for up to 1792 ranks
- AngularDisk6196 is suitable for up to 6196 ranks

In all our experiments we use Xeon Broadwell nodes, placing 28 ranks per node. Using fewer ranks per node did not yield a performance gain. Hyper-threading was not beneficial, but rather detrimental. Scaling charts are shown in Figure 1.

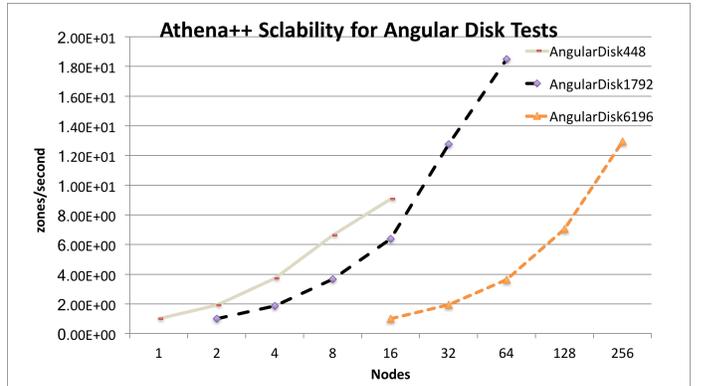


Fig. 1. Athena++ scalability for different problem sizes on Xeon Broadwell nodes. The vertical axis reports zones/second, therefore higher means better.

We note that the application scales, almost to maximum number of ranks for all 3 test cases. The chart in Figure 2 shows computation and communication time on a logarithmic scale for an increasing number of nodes. Both, computation and communication decrease up to 32 nodes. Almost all of the overall time is spent in computation. Only for 64 nodes, the maximum possible number for this test case, does the communication cost increase significantly and starts to impact the overall time significantly.

We note that even for the maximum number of ranks most of the time is spent in computation. Most of the communication is performed through point-to-point (p2p) communication, which can be overlapped with computation. This behavior was the same for all 3 test cases.

The chart in Figure 3 shows the average memory used by the application per node. We see that the usage is decreasing

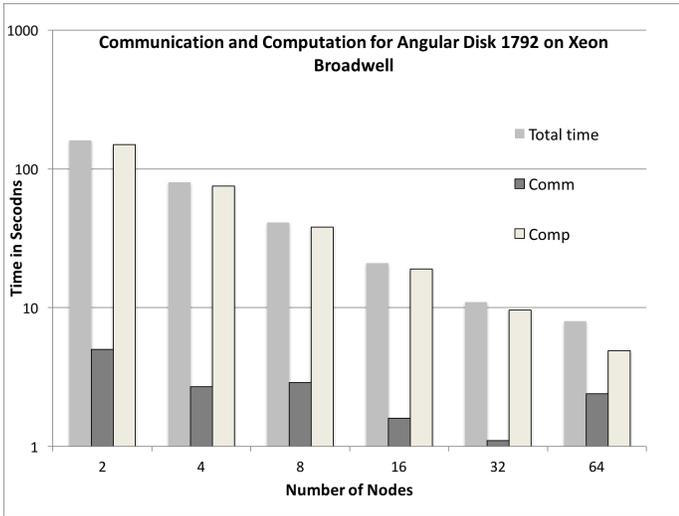


Fig. 2. Computation and MPI communication Angular Disk 1792 test case. 28 MPI ranks are placed on each compute node.

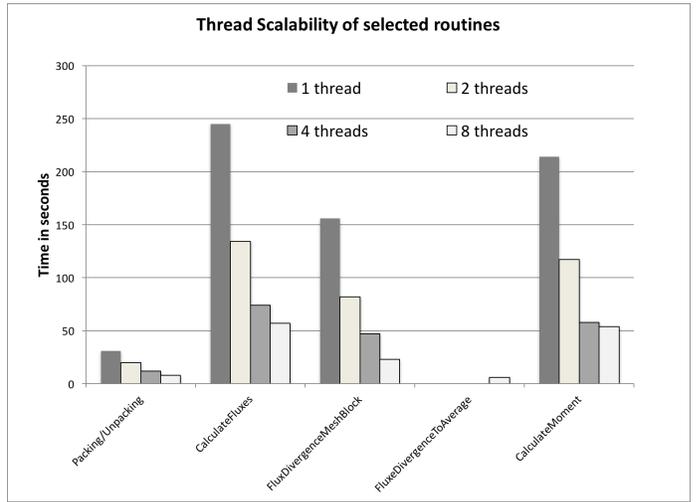


Fig. 4. Routine based thread scalability for a linear wave test case.

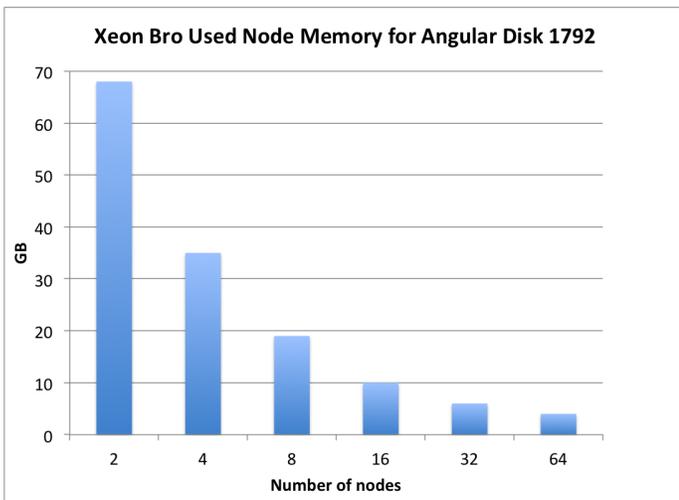


Fig. 3. Athena++ used node memory with an increasing number of nodes for angular disk case 1792. There are 28 MPI ranks on each node. Each nodes provides 128 GB of DRAM.

when increasing the number of ranks. This indicates that there is very little replicated data within Athena.

C. Thread Scalability on Xeon Nodes

We did some preliminary performance studies employing multithreading via OpenMP. Our thread scalability experiments were conducted for a linear wave, which is very similar to the previous disk simulation in the sense that they are both using the same modules in the code. The only difference is that the linear wave is testing the propagation of a radiation modified sound wave in a uniform background medium. We used 64 nodes, placing 1 MPI rank per node. Other placement schemes are possible, but we wanted to allow for a maximum number of 28 threads. The performance profile in Athena is flat. Below is an excerpt of a performance profile for one of the most time consuming ranks for a linear wave calculation:

```

Elap: 61.81
User: 57.13
Sys: 4.48
Total: 66.64 Symbol
      8.79 RadIntegrator::CalculateFluxes
      8.61 Radiation::CalculateMoment
      5.93 RadIntegrator::FluxDivergence
      5.19 RadIntegrator::AddSourceTerms
      4.74 BufferUtility::Unpack4DDData
      4.45 Radiation::CalculateComMoment
      4.14 BufferUtility::Pack4DDData
      1.80 RadIntegrator::SecondOrderFluxX1
      1.63 RadIntegrator::AbsorptionScattering
....

```

We focus for now on a few of the most time consuming routines. Figure 4 shows the timings for the routines that we had parallelized as described in Section III-C. We see a good scalability up to about 4 threads. Using 8 threads still yields performance improvement for some routines, but we noticed a significant load imbalance among the threads. Our future work will include increasing thread scalability.

D. Comparing Xeon to KNL

As mentioned earlier, our KNL system is endowed with 16 GB of MCDRAM per node. If not mentioned otherwise, we have configured all of the MCDRAM as cache for our experiments. We ran the 3 test angular disk cases on 8 KNL nodes.

The chart in Figure 5 shows the performance on Xeon and KNL for test case 1792.

We employed hyper-threading on KNL, placing 224 threads per node. Hyper-threading turned out be beneficial on KNL. This is due to excellent MPI scalability of Athena++. Being able to run with a higher number of ranks outweighs any negative impact of resource sharing due to hyper-threading. For our test cases we noticed a performance advantage of KNL over Xeon Broadwell, which is increasing when increasing number of nodes.

All of the computationally intensive routines are fully vectorized, either by the compiler directly or by inserting *omp*

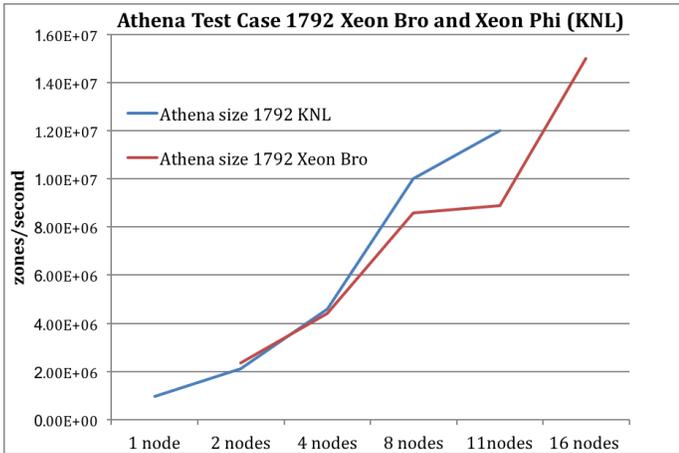


Fig. 5. Athena++ Performance on Xeon and Xeon Phi. Higher rates indicate better performance. All of the 16 GB MCDRAM are configured as cache.

simd pragmas as described in Section III-B. The KNL processor has an *avx512* instructions set. The vector register width is 64 bytes, which is double the size of the *avx* instruction set of the Xeon processor. Vectorization is therefore very important for the KNL system. We compared the performance running with and without vectorization in Figure 6

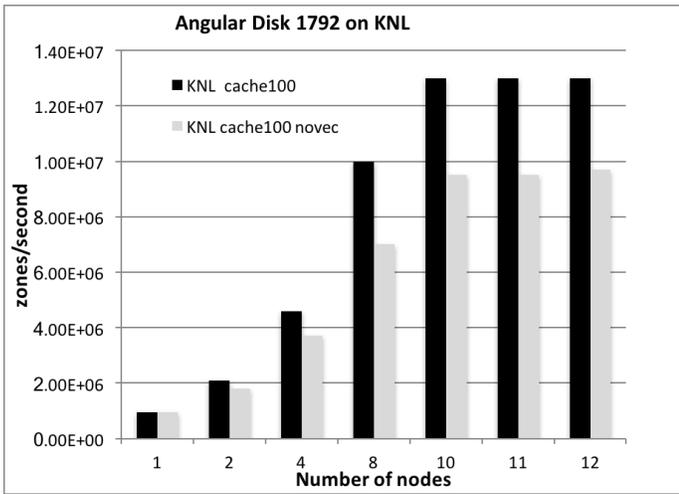


Fig. 6. Athena++ Performance with and without vectorization on KNL. For runs on 10, 11 and 12 nodes, there are fewer than 28 ranks per node. Higher rates indicate better performance.

We do observe a significant performance increase using fully vectorized code. Performance profiling indicated, however, that the execution is very memory bound. We therefore collected profiling statistics to better understand the impact of memory and cache size on the performance. We have collected statistics on per node memory usage on KNL. The chart in Figure 7 shows behavior of speed-up and per-node memory usage. We plotted the ratio of the total memory usage over the size of available MCDRAM. We observe that the speed-up increases with increasing core count. Correspondingly, there is an increase of the percentage of application memory that

fits into MCDRAM. Going up to 8 nodes, the application benefits from both aspects. The number of MPI ranks does not increase from 8 nodes to 11 nodes, as it is restricted to 1792. Nevertheless, spreading ranks across more nodes further decreases the memory requirements per node. For runs on 8 nodes and more, all data fits within the 16 GB MCDRAM.

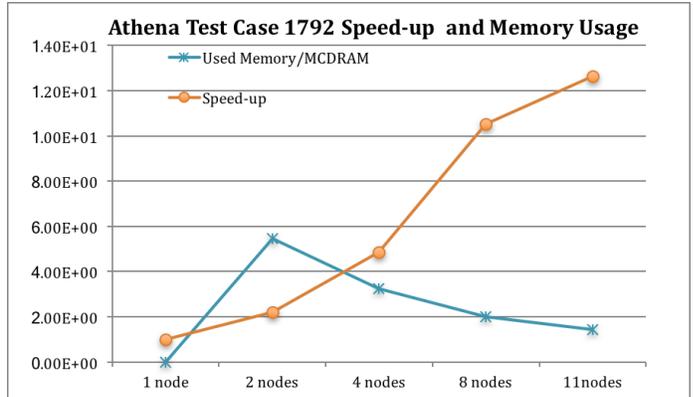


Fig. 7. Speed-up and Memory Usage on KNL for Angular Disk Case 1792

We ran our tests in different cache configuration modes to further study the impact of high bandwidth memory. The chart in Figure 8 shows the performance when configuring 25%, 50% and 100% of the MCDRAM as cache. The performance on KNL improves with increased availability of MCDRAM. This is a further indication that the application is memory bound.

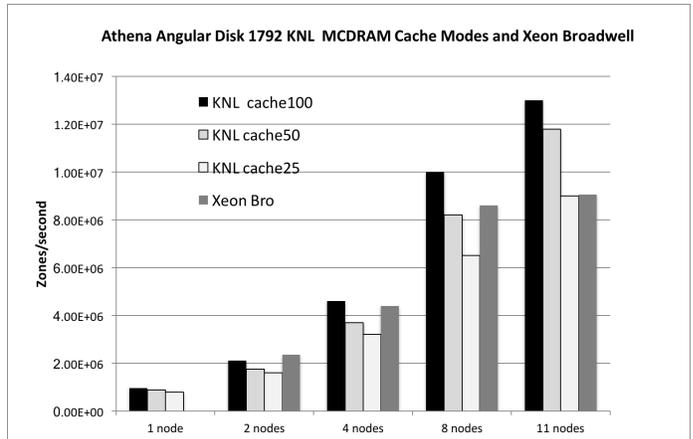


Fig. 8. Performance for Angular Disk 1792 on KNL in different cache modes.

E. Preliminary performance analysis on Intel Skylake nodes

A subset of the *avx512* instructions is also available for Intel Skylake nodes. While KNL and Skylake share a large set of the instructions, the sets are not identical. For example, KNL has instructions for Exponential, Reciprocal, and Prefetch instructions. Skylake includes instructions for Vector Length extensions, Byte, Word, Double and Quadword instructions and others. In addition to *xmm* and *ymm* registers, *zmm*

registers are available to support a 512 bit SIMD length. On Skylake, we set the

```
-xcore-avx512
```

compiler flag to enable the generation of avx512 instructions. By default, the Intel compiler will not be aggressive about using the zmm registers. Starting with Intel v 18.0 compilers, a new flag

```
-qopt-zmm-usage=low|high
```

is added to enable user control about the aggressiveness of the compiler. We compiled using 3 different sets of compiler flags

```
icc -xcore-avx2 -O3 -qopenmp-simd
```

```
icc -xcore-avx512 -O3 -qopenmp-simd
```

```
icc -xcore-avx512 -qopt-zmm-usage=high -O3 -qopenmp-simd
```

We ran a linear wave test case using 512, 1024 and 2048 ranks on Skylake nodes. We placed 40 ranks on each Skylake node, using 13, 26 and 52 nodes respectively. The chart in Figure 9 shows the performance. It also includes the performance for runs on 19, 37 and 74 Broadwell nodes. By default, the compiler did generate avx2 instructions, even when the `-xcore-avx512` flag was set. Increasing the usage of zmm registers yields a slight overall performance degradation. Performance analysis revealed that the most time consuming basic blocks did in fact benefit from the increased SIMD length. However, we noticed an increase of time spent in system library calls to memory copy operations. We are currently investigating the cause of this. When comparing Skylake to Broadwell performance, we observe that 52 Skylake nodes outperform 74 Broadwell nodes by about 10%. As we have found that increased vector length did not help the overall performance, we assume at this point that the increase is due entirely to increased memory bandwidth on the Skylake nodes. We are currently conducting further studies on this.

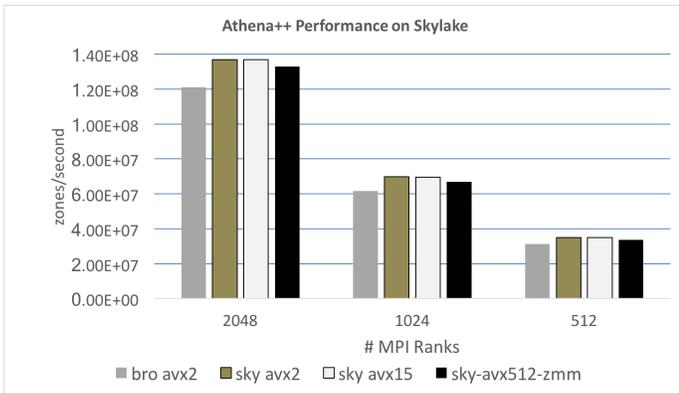


Fig. 9. Performance impact of avx512 instructions on Skylake for a linear wave test case. The aggressive use of zmm registers degrades the overall performance.

V. CONCLUSIONS AND FUTURE WORK

We have conducted an in-depth performance study of the radiation transport method in the Athena++ Astrophysics

code. We looked into MPI communication, multithreading and vectorization. We have experimented on different hardware architectures, such as Intel Xeon Broadwell and Intel Xeon Phi (KNL). Some preliminary results have also been added for Intel Xeon Skylake nodes. We found that Athena++ lends itself to exploiting parallelism on all levels of parallelism on modern multi- and many-core processors. Using asynchronous communication and persistent communication requests, as well as minimizing the amount of replicated data and MPI buffer space lead to very good MPI scalability. Traversing data structures with stride 1 in the innermost loops allows exploiting SIMD parallelism, which improves core compute performance on modern processors. This is supported by arranging complex data structures as SOA (Structures Of Arrays). In addition to this, Athena++ lends itself to thread parallelism, which may further decrease the amount of communication. In our future work we are planning to address aspects such as extending OpenMP parallelization in Athena++ and improving memory access patterns to overcome memory bandwidth limitations. This could be achieved, for example, by re-arranging data structures and by using streaming store operations. In addition, we are planning to experiment using GPU accelerators.

REFERENCES

- [1] T. A. Thompson, E. Quataert, and N. Murray. Radiation Pressure-supported Starburst Disks and Active Galactic Nucleus Fueling. *ApJ*, 630:167–185, September 2005.
- [2] C. F. McKee and E. C. Ostriker. Theory of Star Formation. *ARA&A*, 45:565–687, September 2007.
- [3] M. R. Krumholz, R. I. Klein, C. F. McKee, S. S. R. Offner, and A. J. Cunningham. The Formation of Massive Star Systems by Accretion. *Science*, 323:754–, February 2009.
- [4] R. Barkana and A. Loeb. In the beginning: the first sources of light and the reionization of the universe. *Phys. Rep.*, 349:125–238, July 2001.
- [5] N. Murray, E. Quataert, and T. A. Thompson. The Disruption of Giant Molecular Clouds by Radiation Pressure—the Efficiency of Star Formation in Galaxies. *ApJ*, 709:191–209, January 2010.
- [6] L. Ciotti and J. P. Ostriker. Radiative Feedback from Massive Black Holes in Elliptical Galaxies: AGN Flaring and Central Starburst Fueled by Recycled Gas. *ApJ*, 665:1038–1056, August 2007.
- [7] D. Proga. Dynamics of Accretion Flows Irradiated by a Quasar. *ApJ*, 661:693–702, June 2007.
- [8] N. I. Shakura and R. A. Sunyaev. Black holes in binary systems. Observational appearance. *A&A*, 24:337–355, 1973.
- [9] D. Mihalas and B. W. Mihalas. *Foundations of radiation hydrodynamics*. 1984.
- [10] N. Y. Gnedin and T. Abel. Multi-dimensional cosmological radiative transfer with a Variable Eddington Tensor formalism. *New Astronomy*, 6:437–455, October 2001.
- [11] M. A. Skinner and E. C. Ostriker. A Two-moment Radiation Hydrodynamics Module in Athena Using a

- Time-explicit Godunov Method. *ApJS*, 206:21, June 2013.
- [12] J. M. Stone, D. Mihalas, and M. L. Norman. ZEUS-2D: A radiation magnetohydrodynamics code for astrophysical flows in two space dimensions. III - The radiation hydrodynamic algorithms and tests. *ApJS*, 80:819–845, June 1992.
- [13] J. C. Hayes and M. L. Norman. Beyond Flux-limited Diffusion: Parallel Algorithms for Multidimensional Radiation Hydrodynamics. *ApJS*, 147:197–220, July 2003.
- [14] I. Hubeny and A. Burrows. A New Algorithm for Two-Dimensional Transport for Astrophysical Simulations. I. General Formulation and Tests for the One-Dimensional Spherical Case. *ApJ*, 659:1458–1487, April 2007.
- [15] Y.-F. Jiang, J. M. Stone, and S. W. Davis. A Godunov Method for Multidimensional Radiation Magnetohydrodynamics Based on a Variable Eddington Tensor. *ApJS*, 199:14, March 2012.
- [16] S. W. Davis, J. M. Stone, and Y.-F. Jiang. A Radiation Transfer Solver for Athena Using Short Characteristics. *ApJS*, 199:9, March 2012.
- [17] C. D. Levermore and G. C. Pomraning. A flux-limited diffusion theory. *ApJ*, 248:321–334, August 1981.
- [18] B. Dubroca and J. Feugeas. Etude théorique et numérique d’une hiérarchie de modèles aux moments pour le transfert radiatif. *Academie des Sciences Paris Comptes Rendus Serie Sciences Mathematiques*, 329:915–920, November 1999.
- [19] J. A. Pons, J. M. Ibáñez, and J. A. Miralles. Hyperbolic character of the angular moment equations of radiative transfer and numerical methods. *MNRAS*, 317:550–562, September 2000.
- [20] N. J. Turner and J. M. Stone. A Module for Radiation Hydrodynamic Calculations with ZEUS-2D Using Flux-limited Diffusion. *ApJS*, 135:95–107, July 2001.
- [21] M. R. Krumholz, R. I. Klein, C. F. McKee, and J. Bolstad. Equations and Algorithms for Mixed-frame Flux-limited Diffusion Radiation Hydrodynamics. *ApJ*, 667:626–643, September 2007.
- [22] W. Zhang, L. Howell, A. Almgren, A. Burrows, and J. Bell. CASTRO: A New Compressible Astrophysical Solver. II. Gray Radiation Hydrodynamics. *ApJS*, 196:20, October 2011.
- [23] B. van der Holst, G. Tóth, I. V. Sokolov, K. G. Powell, J. P. Holloway, E. S. Myra, Q. Stout, M. L. Adams, J. E. Morel, S. Karni, B. Fryxell, and R. P. Drake. CRASH: A Block-adaptive-mesh Code for Radiative Shock Hydrodynamics-Implementation and Verification. *ApJS*, 194:23, June 2011.
- [24] B. Commerçon, R. Teyssier, E. Audit, P. Hennebelle, and G. Chabrier. Radiation hydrodynamics with adaptive mesh refinement and application to prestellar core collapse. I. Methods. *A&A*, 529:A35+, May 2011.
- [25] M. González, E. Audit, and P. Huynh. HERACLES: a three-dimensional radiation hydrodynamics code. *A&A*, 464:429–435, March 2007.
- [26] A. Sądowski, R. Narayan, A. Tchekhovskoy, and Y. Zhu. Semi-implicit scheme for treating radiation under M1 closure in general relativistic conservative fluid dynamics codes. *MNRAS*, 429:3533–3550, March 2013.
- [27] J. C. McKinney, A. Tchekhovskoy, A. Sądowski, and R. Narayan. Three-dimensional general relativistic radiation magnetohydrodynamical simulation of super-Eddington accretion, using a new code HARMRAD with M1 closure. *MNRAS*, 441:3177–3208, July 2014.
- [28] J. I. Castor. *Radiation Hydrodynamics*. November 2004.
- [29] S. W. Davis, Y.-F. Jiang, J. M. Stone, and N. Murray. Radiation Feedback in ULIRGS: Are Photons Movers and Shakers? *arXiv:1403.1874*, March 2014.
- [30] B. A. Whitney. Monte Carlo radiative transfer. *Bulletin of the Astronomical Society of India*, 39:101–127, March 2011.
- [31] J. D. Densmore, T. J. Urbatsch, T. M. Evans, and M. W. Buksas. A hybrid transport-diffusion method for Monte Carlo radiative-transfer simulations. *Journal of Computational Physics*, 222:485–503, March 2007.
- [32] J. Steinacker, M. Baes, and K. D. Gordon. Three-Dimensional Dust Radiative Transfer*. *ARA&A*, 51:63–104, August 2013.
- [33] N. Roth and D. Kasen. Monte Carlo Radiation Hydrodynamics with Implicit Methods. *ArXiv e-prints*, April 2014.
- [34] Y.-F. Jiang, J. M. Stone, and S. W. Davis. An Algorithm for Radiation Magnetohydrodynamics Based on Solving the Time-dependent Transfer Equation. *ApJS*, 213:7, July 2014.
- [35] Y.-F. Jiang, J. M. Stone, and S. W. Davis. On the Thermal Stability of Radiation-dominated Accretion Disks. *ApJ*, 778:65, November 2013.
- [36] Y.-F. Jiang, J. M. Stone, and S. W. Davis. A Global Three-dimensional Radiation Magneto-hydrodynamic Simulation of Super-Eddington Accretion Disks. *ApJ*, 796:106, December 2014.
- [37] Y.-F. Jiang, S. W. Davis, and J. M. Stone. Iron Opacity Bump Changes the Stability and Structure of Accretion Disks in Active Galactic Nuclei. *ApJ*, 827:10, August 2016.
- [38] Y.-F. Jiang, J. Guillochon, and A. Loeb. Prompt Radiation and Mass Outflows from the Stream-Stream Collisions of Tidal Disruption Events. *ApJ*, 830:125, October 2016.
- [39] Y.-F. Jiang, M. Cantiello, L. Bildsten, E. Quataert, and O. Blaes. Local Radiation Hydrodynamic Simulations of Massive Star Envelopes at the Iron Opacity Peak. *ApJ*, 813:74, November 2015.
- [40] Y.-F. Jiang, M. Cantiello, L. Bildsten, E. Quataert, and O. Blaes. The Effects of Magnetic Fields on the Dynamics of Radiation Pressure-dominated Massive Star Envelopes. *ApJ*, 843:68, July 2017.
- [41] OpenMP 4.5 Specifications <http://www.openmp.org/wp-content/uploads/openmp-4.5.pdf>.
- [42] W. Gropp, T. Hoefler, R. Thakur, E. Lusk

Using Advanced MPI <http://www.openmp.org/wp-content/uploads/openmp-4.5.pdf>.