Outflows from the Magellanic Clouds: MHD Simulations and Multi-Wavelength Synthetic Observations

Scaling And Performance

Computational Method

We use the FLASH v4.2 magnetohydrodynamics (MHD) code to carry out our simulations. To solve the ideal MHD equations, we use the unsplit staggered mesh MHD solver (Lee and Deane 2009, Lee 2013), which is based on a finite-volume, high-order Godunov scheme and uses a constrained transport scheme to ensure divergence-free magnetic fields. FLASH has been well-tested and optimized for use on massively parallel architectures, making it a popular MHD code in the astrophysics community. As discussed in our main proposal document, we use a modified version of FLASH that includes an additional cosmic ray module, which evolves cosmic rays as a second fluid and includes a fluid description of the kinetic cosmic ray streaming process. In practice, the cosmic ray fluid is defined as a mass scalar in FLASH, and it obeys a relativistic equation of state, as well as a separate evolution equation that depends on whether streaming is desired or not. Because, in the streaming picture, the direction of flow along field lines is always directed down the cosmic ray pressure gradient, numerical issues arise near extrema in cosmic ray pressure, where the gradient changes sign. To counteract this we use a regularization method, in which one chooses an appropriate scale length, L, for the system that then defines a characteristic cosmic ray pressure gradient, P_{CR}/L. The streaming speed of cosmic rays, which ideally is v_A, is approximated as

$$v_s = v_A \tanh\left(\frac{|\hat{\mathbf{b}} \cdot \nabla \mathrm{P_{CR}}|}{\mathrm{P_{CR}/L}}\right) \tag{1}$$

For a cosmic ray pressure gradient ∇P_{CR} much greater than P_{CR}/L, $v_s = v_A$, while smaller cosmic ray pressure gradients will lead to $v_s < v_A$. One would like L to be as large as possible (so $v_s \approx v_A$ for a wide range of cosmic ray pressure gradients in the system); however, the proper simulation timestep constraint for this regularization method is $dt < dx^2/(v_s L)$, which is second order in cell width. This timestep constraint can be written as

$$dt = 9.26 \times 10^{10} \left(\frac{f_{cfl}}{0.8}\right)\left(\frac{N}{4}\right)\left(\frac{dx}{100\mathrm{pc}}\right)^2\left(\frac{v_s}{400\mathrm{km/s}}\right)^{-1}\left(\frac{L}{10\mathrm{kpc}}\right)^{-1} s \tag{2}$$

where N is the number of subcycles over the streaming term (with 4 being the largest number since we employ 4 guard cells) and f_{cfl} is the Courant number.

Aside from this restrictive timestep criterion for streaming, the inclusion of cosmic rays in FLASH only minimally increases the memory required per processor (because we keep track of one more variable) and slightly increases the computation time to update each cell due to subcycling over the streaming term. It does not affect the code scalability.

Scaling And Timing

To characterize the scalability of our FLASH 4.2 simulations on Stampede2, we start with a simple study using 512 blocks, each with 8^3 cells, where we successively increase the domain size and

number of blocks by a factor f and accordingly increase the number of processors by the same factor f. This makes sure the work load per processor is roughly the same in each test. We do not increase the number of particles according to the number of blocks, however. In our real simulations, we will have only a few thousand particles at any given time, and in these tests, we generate up to a few thousand particles. Overall, these tests allows us to get a sense of weak scaling for our problem, though the exact times will not correspond to any of our planned simulations. The timing results for the total of 100 time steps are given in Table 1 below.

N_{cpu}	$t_{MHD}(s)$	$t_{particles}(s)$	$t_{IO}(s)$	t_{total}	ϵ_{MHD}	$\epsilon_{particles}$	ϵ_{IO}	ϵ_{total}
16	111.4	18.2	2.3	130.8	1	1	1	1
32	133.1	22.2	2.5	156.5	0.83	0.82	0.92	0.83
64	140.7	22.1	3.2	164.9	0.79	0.82	0.72	0.79
128	140.1	20.8	3.7	161.7	0.8	0.88	0.63	0.81
256	139.9	20.5	6.7	161.4	0.79	0.89	0.34	0.81
512	141.1	20.6	10.6	162.9	0.79	0.88	0.22	0.80
1024	143.3	20.8	17.6	165.1	0.78	0.87	0.13	0.79

Table 1: Weak scaling table. We find good overall weak scaling relative to the 16 CPU run, although the I/O, which we define as the time to write one checkpoint file, scales poorly. This is not an issue, however, because we will only write ≈ 100 checkpoint files per simulation, which will take up a negligible amount of the total runtime.

We now perform tests more akin to the real simulations we plan to run with AMR. We use two domains: a 60 kpc x 60 kpc x 60 kpc domain when ram pressure stripping is included and a 40 kpc x 40 kpc x 40 kpc domain when ram pressure is turned off. In the ram pressure case, our domain must be larger to track the disk's stripped gas that forms a filament many tens of kpcs long. Our target resolution for each simulation type (ram pressure or no ram pressure) is ≈ 39.5 pc (not including our single proposed run up to a resolution of 19.75 pc) with a base resolution of 312 pc. For simulations without stripping, this is accomplished by using a base grid of 16^3 PARAMESH blocks (128^3 cells). This grid is then refined up to n additional levels based on density, with every block containing a density of 10^{-26}g/cm^3 being further refined into smaller blocks until the maximum of n refinement levels is achieved.

In Figure 1, we present a scaling test of our $(40\text{kpc})^3$ LMC model without cosmic ray streaming. We separate tests with and without cosmic ray streaming because of the increase in workload and in evolution time per timestep due to subcycling over the streaming term. With streaming, an AMR level of 3 with 1024 processors took 277.750 seconds to take 200 timesteps, while the same test with streaming took 371.896 seconds. Strong scaling efficiencies with streaming (not shown) are very similar to those without streaming. The difference in run-time is reflected in our timing estimates of eqns. 3-6 in the General Timing Estimation section below. In each test, we include our full suite of physical processes, including the formation of star cluster particles, feedback from these particles onto neighboring cells, and radiative cooling with subcycling included. We additionally refine the grid once during this time period, which we find takes a negligible amount of time, and output two checkpoint files, one plotfile (reduced version of the checkpoint files), one particle file, and a data output file. Our tests model the first 200 timesteps of our LMC galaxy evolution, with the exception of the $n = 4$ test, which is our most time-consuming simulation and only runs for 50 timesteps. The LMC initial condition, including magnetic field configuration, is outlined in the Research Usage Plan of the main document.

For each refinement level, we start near the minimum number of cores required for a given

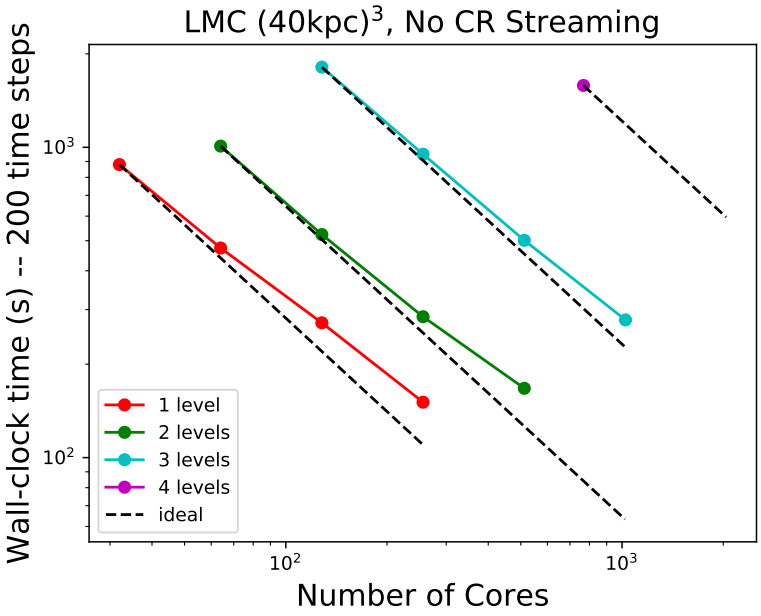

Figure 1: Wall-clock times for different AMR level simulations. The dashed line is the ideal speed-up relative to wall-clock time for a given AMR level with the lowest number of cores.

problem size, which is dictated by our choice of the maximum number of blocks per processor. We have experimented with this MAXBLOCKS array parameter and find that setting it to 400 gives a good trade-off between the minimum number of cores required and the memory per processor. For our LMC simulations, this translates to ≈ 3.5 Gb per processor, which fits within the 192 Gb RAM per node on the Stampede2 Skylake nodes if we use under ≈ 54 processors per node. We choose to use 32 processors per node in order to safely avoid any memory issues.

In Figure 1, we see that, for a given number of refinement levels (given number of blocks), our simulations show excellent strong scaling only slightly worse than ideal.

In Table 2, we tabulate these wall-clock times and efficiencies for the 3 AMR level run, and we also give these numbers for the refinement time. Overall, we find excellent scaling greater than 80% for the total evolution time and greater than 90% for the MHD updates. Particles scale more poorly due to communication overhead; when feedback occurs in the cells surrounding a particle, all processors communicate with each other to see if the particle is near a block edge, and hence, if neighboring cells lie in a different block (on a different processor). Since we generally only have a few thousand particles present at any given time, and since the particles unit takes up only a small fraction of the total runtime, we are not concerned with this inefficiency for the purposes of this study. We also find that refinement does not scale well with increasing cores, but it also takes up a very small amount of our total time due to the infrequency of refinement in our simulations (we find that every 100 timesteps is sufficient).

Optimizations

We utilize FLASH's AMR capabilities such that the grid refines and derefines based on a density threshold. This minimizes the memory requirement and runtime of our simulations, while allowing us to adaptively follow, in higher resolution, stripped and outflowing material tens of kpc from the galaxy center. This would not be possible using a uniform grid or static mesh refinement centered

N_{cpu}	$t_{MHD}(s)$	$t_{particles}(s)$	$t_{refinement}(s)$	t_{total}	ϵ_{MHD}	$\epsilon_{particles}$	$\epsilon_{refinement}$	ϵ_{total}
128	1534.703	242.378	7.753	1813	1	1	1	1
256	792.664	139.627	5.934	950.276	0.968	0.867	0.653	0.954
512	405.713	84.157	5.325	501.469	0.945	0.720	0.363	0.904
1024	207.856	61.219	4.953	277.750	0.922	0.494	0.195	0.815

Table 2: Strong scaling table for the level 3 AMR run, which is at our target resolution. For a given simulation size, the total runtime scales well at an efficiency of over 80% for the total evolution time and over 90% for the MHD update time, which dominates the total runtime. Particles and refinement scale more poorly, but they take up a small portion of the total runtime, with the particles unit taking up at worst $\approx 22\%$ of the runtime using 1024 cores.

on our disk. While the grid refinement/derefinement doesn't seem to be a large time sink in our simulations, we limit the refinement to occur every 100 timesteps, which for a timestep of a few $\times 10^{11}$ seconds, is approximately every Myr, or about a 1000 times per simulation.

The I/O is done using FLASH's parallel HDF5 unit, which gives a significant speed-up over serial I/O. The time to write a checkpoint file of size 6.6 GB containing 47552 blocks (for $n = 3$ levels of refinement) on the Stampede2 /work partition is 14.976 seconds using 1024 processors, which is of minimal strain on our total computation time since we plan to write checkpoint files only $\approx 50 - 100$ times per simulation, each time re-writing previous files and keeping only the most recent ≈ 3 checkpoint files in order to reduce storage costs. Plot files, which contain only the necessary variables for plotting, are of order 1.3 GB and take only 2 seconds to write to the /work partition using 1024 cores. We plan to write these twice as often as checkpoint files for a total of 100 to 200 plot files per simulation. Allowing for grid refinement, we can expect $\approx 150 - 300$ GB of data per simulation at our target resolution of 3 AMR levels.

The timestep constraint to resolve cosmic ray streaming is the most computationally expensive aspect of our simulations, but we make every effort to speed this up by capping the Alfven speed used in the cosmic ray flux and thermal gas heating terms (see eqns. 6 and 7 of the Main Document) and by subcycling four times, which quadruples our timestep. We similarly subcycle our radiative cooling routine in order to properly capture the cooling time while maintaining a large simulation timestep.

General Timing Estimation

To estimate the computation time for a simulation, $t_{\rm sim}$, we take the product of the domain update time per step (for a given number of cores and blocks) times the total number of steps, $N_{\rm steps}$, times the total number of blocks, $N_{\rm blocks}$. This is further divided by some efficiency, ϵ, which we find from our scaling studies to be $> 75\%$, therefore allowing us to interpolate between our timing values for the desired number of blocks and processors. For the domain update time, we take our timing study results from each of the 4 cases: ($(40\text{kpc})^3$ box and $(60\text{kpc})^3$ box with and without streaming). For example, for our $(40\text{kpc})^3$ box without streaming, we take the value we get from our 32 core, 1 AMR level scaling study test (which has 5632 total blocks), which is 4.398 s per timestep.

For the $(40\text{kpc})^3$ box without streaming:

$$t_{\rm sim} = 4.398 \left(\frac{32 \quad \text{cores}}{5632 \quad \text{blocks}} \right) \times N_{\rm steps} \times \frac{N_{\rm blocks}}{N_{\rm cpu} \times \epsilon} \quad \text{seconds} \tag{3}$$

For the $(40\text{kpc})^3$ box *with* streaming:

$$t_{\text{sim}} = 3.089 \left(\frac{64 \quad \text{cores}}{5632 \quad \text{blocks}} \right) \times N_{\text{steps}} \times \frac{N_{\text{blocks}}}{N_{\text{cpu}} \times \epsilon} \quad \text{seconds} \tag{4}$$

For the $(60\text{kpc})^3$ box without streaming:

$$t_{\text{sim}} = 4.026 \left(\frac{128 \quad \text{cores}}{15360 \quad \text{blocks}} \right) \times N_{\text{steps}} \times \frac{N_{\text{blocks}}}{N_{\text{cpu}} \times \epsilon} \quad \text{seconds} \tag{5}$$

For the $(60\text{kpc})^3$ box *with* streaming:

$$t_{\text{sim}} = 3.087 \left(\frac{256 \quad \text{cores}}{22528 \quad \text{blocks}} \right) \times N_{\text{steps}} \times \frac{N_{\text{blocks}}}{N_{\text{cpu}} \times \epsilon} \quad \text{seconds} \tag{6}$$

$N_{\text{steps}} \approx 1\text{Gyr}/\text{t}_{\text{step}}$ can be calculated from our average timestep t_{step}, which is either the streaming timestep given in eqn. 2 or can be estimated without streaming given our CFL number, cell width, and fastest propagation speed in our domain.

The number of blocks, N_{blocks}, will dynamically change throughout the simulation. One can write this in terms of a covering fraction, C_k, for each AMR level $k > 0$, i.e. what fraction the domain is covered by cells at a certain refinement level relative to the base $k = 0$ level:

$$N_{\text{blocks}} = n_{\text{base}}^3 + (n_{\text{base}} * 2^1)^3 C_1 + (n_{\text{base}} * 2^2)^3 C_2 + ...(n_{\text{base}} * 2^k)^3 C_k) \tag{7}$$

where $n_{\text{base}} = 16$ for the $(40\text{kpc})^3$ box, and $n_{\text{base}} = 24$ for the $(60\text{kpc})^3$ box for a base resolution of 312 pc at $k = 0$.

This covering fraction depends on the amount of gas expelled from the LMC or SMC by outflows and ram pressure stripping, and estimates for each of our planned simulations are given in the Justification of Resources section of the Main Document.