Performance optimization of the GReX code

AstroLab Team: Fabio Baruffa, Salvatore Cielo and Luigi Iapichino
Participants: Elias Roland Most and Ludwig Jens Papenfort
Institut für Theoretische Physik, Goethe Universität, Frankfurt am Main

Scientific goals

Neutron star mergers are amongst the most violent events in the universe, where two compact objects of the size of Manhattan weighing more than our Sun collide under the influence of strong gravity. The gravitational waves (GW) emitted by such an event have recently been observed by the LIGO network for the first time. This detection, GW170817, was accompanied by an electromagnetic (EM) kilonova afterglow, AT2017gfo, resulting from the decay of heavy elements formed in the ejected material undergoing r-process nucleosynthesis. It has also established a firm connection to short gamma-ray bursts, enabling a multimessenger view on neutron star mergers for the first time. All these EM counterparts are ultimately enabled by the presence of strong magnetic fields in the merger remnant that either drive mass outflows by an induced shear viscosity or facilitate jet launching when a black hole is formed. Despite its importance, the magnetic field evolution in the merger is only poorly understood: At the time of merger a shear layer forms between the two stars in which a strong magnetic field amplification by small scale turbulence is taking place, see Fig. 1. The main computational obstacle is that very high numerical resolutions are required to properly resolve the turbulence, which as previous studies indicate, may well be $< 1\,\mathrm{m}$. Simulations to model these systems in 3D by using magnetohydrodynamics coupled to fully dynamical space-time evolution are computationally extremely demanding and the requirements to reach these scales are twofold. First, an efficient adaptive mesh refinement technique needs to be employed such that the shear layer can be adequately resolved in a computationally efficient fashion while keeping dynamically less important regions at low resolution. Second, solving the system of equations in this hierarchical way needs to be scalable to large core counts. Ultimately, in order to further push the resolution to new frontiers and gain important insights about the small scale turbulence present in this process, exascale machines will be the only means of making significant progress.

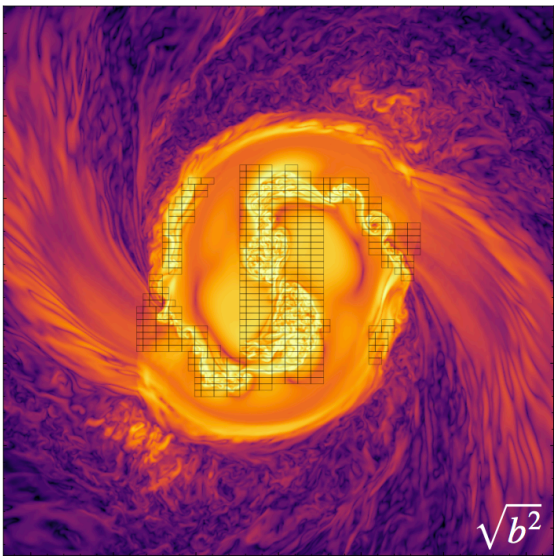

Figure 1: Magnetic field amplification due to the turbulent Kelvin-Helmholtz instability in a binary neutron star merger simulated with `GReX`. Adaptive mesh refinement efficiently captures the shear layer at extreme resolutions (35 m).

Code overview and performance

In order to model this process we plan to solve the equations of ideal general-relativistic magnetohydrodynamics in dynamical space-times using the `GReX` code. `GReX` is a new code written in C++17 and is built on the massively parallel AMReX framework (MPI+OpenMP) and has been successfully run on up to **32,000** cores of the SuperMUC-NG system. `GReX` uses a high-order conservative finite volume shock capturing algorithm to solve the equations of general-relativistic magnetohydrodynamics coupled with a third order strongly stability preserving Runge-Kutta timestepping scheme. `GReX` also uses the Z4c scheme to evolve the dynamical space-time required to capture the merger process in neutron star mergers. The equations are solved using a fourth-order accurate centered finite-difference scheme. The AMReX framework, which is the U.S. Exascale Compute Project co-design center for mesh based codes, provides Berger-Rigoutsis patch-based adaptive mesh refinement that enables us

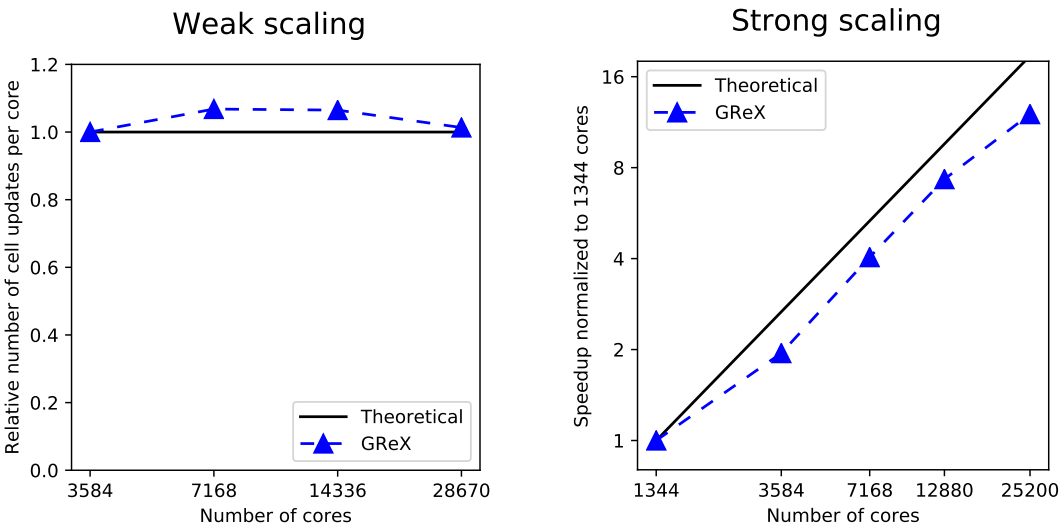

Figure 2: Scaling performed on FRONTERA for the `GReX` code. (*Left*) Weak scaling showing the relative number of cell updates per core. (*Right*) Strong scaling of a production run on the normal queue of FRONTERA.

to resolve regions of high space-time curvature, i.e. close to the neutron star, and also the strong gradients near its surface in a computationally efficient fashion. Application codes based on AMReX have been successfully used on $> 100,000$ cores. To specifically leverage the architectural advantages of recent Intel processors all compute kernels of `GReX` are vectorized using **AVX-512** intrinsics and result in an overall **AVX-512** vector register use across the whole code of $\simeq$ **95**% on a Skylake-SP node on SuperMUC-NG.
In order to assess the scaling of the code we have performed a set of weak and strong scaling test for a medium sized AMR problem on the FRONTERA system, which is very similar to SuperMUC-NG. The results are shown in Fig. 2 and demonstrate good scaling up to the equivalent of 1 island on SuperMUC-NG.

Goals for the AstroLab project

Since the single node performance is already optimized to a very high degree, our goal is to achieve extreme scalability of the code in order to enable even higher numerical resolutions for accurately capturing magnetic turbulence in neutron star merger problems. We would like to continue to benchmark the production setup for a neutron star merger with magnetic turbulence on an extreme scale of 3-4 islands of SuperMUC-NG at LRZ. In particular, we would like to investigate if, given an optimal base grid setup for 1-2 islands (38-76 thousand cores), we can still achieve good performance on such an extreme core count, even when the grid is dynamically adapted as in Fig. 1. Additionally, such scaling entails a demanding I/O test: while so far we have not had any problems with the I/O, which has been used in the context of other AMReX simulations $> 100,000$ cores, we would like to assess that this also works well for our simulations.

Achievements within the AstroLab project

Single node OpenMP scalability

As a preliminary study, Salvatore Cielo and the AstroLab team have investigated the single-node scalability of the GReX code using a realistic simulation setup, scaled down to a single node. By performing different splits between the number of ranks and the number of threads it was found that the code scales well from 1 to 48 cores in both pure MPI and MPI+OpenMP setups. This is shown in Fig. 3. Only hyperthreading was found to not yield any significant improvement in runtime and we chose to not use in the remainder of the tests.

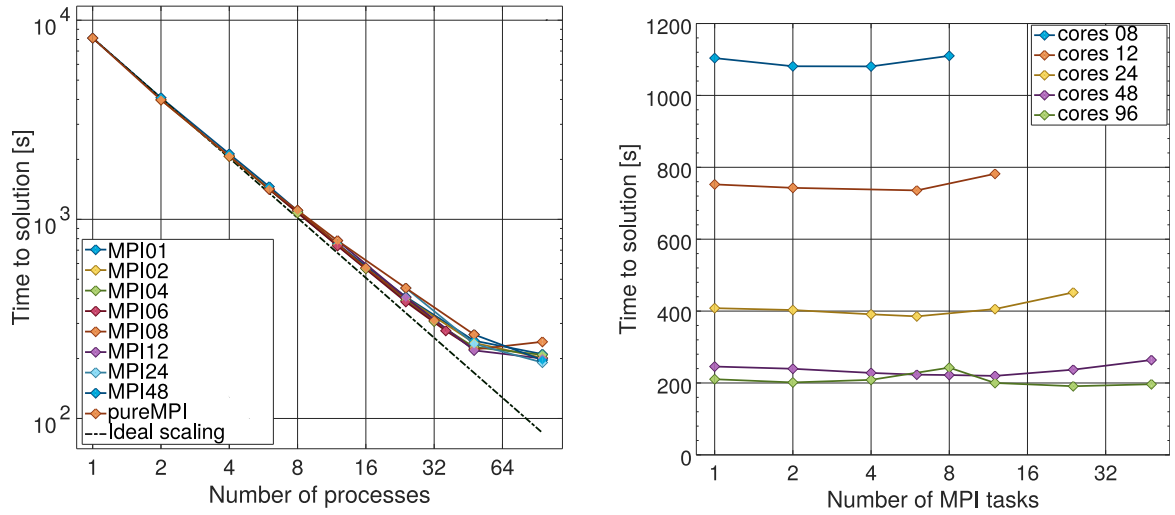

Figure 3: Intranode performance of a typical neutron star merger simulation with the GReX code. (Left) Run time as a function of number of MPI ranks, where the different curves denote different number of OpenMP threads. (Right) Same simulations but for fixed core count. The varying number of MPI ranks corresponds to a different OpenMP thread count. Overall, the code scales well until the physical limit of cores. Hyperthreading (green line) does not yield any performance improvements.

On-site workshop

In February 2020 we have met at LRZ for a one-day on-site workshop to discuss the status of the GReX optimization project. Bottlenecks found up to this point were the logical results of load balancing and communication when using a large number of AMR grids with many cells. After a discussion and a detailed analysis of single node with reports from the Intel® Application Performance Snapshot™(APS) and trace data from Intel Trace Analyzer and Collector™(ITAC), it became clear that one of the major issues is the domain decomposition. This is done via a set of spatially overlapping patches of different sizes. These patches need to be distributed equally among the ranks, but depending on the physical problem this may not always be the case. This highlights the need for a fine tuned physical problem setup when using AMR.

Impact of local tiling

One option to alleviate the decomposition issues is to use local tiling, provided by the AMReX framework. In the tiling approach each patch is split again into subpatches that can be distributed among the threads of the same MPI rank. An investigation into different tile size shows (Table 1) that a tile size of 16 cells may perform as well as without tiling on single-node runs, but degrades the performance significantly

	Max grid size	Tile_size (along y and z)	Time of single timestep	Difference wrt no tiling
Single node	32	32	225.6 s	+2.4%
	32	16	238.1 s	+8.1%
	16	16	237.9 s	+8.0%
256 nodes	32	32	1257 s	+73%
	16	16	912 s	+26%

Table 1: Impact of tile size on the run time for a single time step (average of three). All runs feature 12 OpenMP threads for 4 MPI ranks per node. Tiles izes of 32 or 16 perform almost as well as no tiling for single node runs, but the communication overhead degrades the performance significantly for larger runs, though smaller tiles may help code balancing (as 16 outperforms 32 on 256 nodes). Tile sizes smaller than 16 are never convenient.

(about 25% on total runtime) on larger runs (256 nodes of SuperMUC-NG and above). The reason for this is likely found in the communication overhead at the boundary of each newly created tile. Thus, while tiling may have significant impact in other contexts, we prefer to focus any further optimization effort on alleviating the pressure on the MPI communication.

Improved ghost zone exchange

In order to reduce some of the MPI overhead found in the APS analysis, the ghost zone exchange was rewritten to reduce the amount of communication when using subcycling in time. The overhead stemmed from the fact that for an AMR level l, the next higher level $l+1$ requires the filling of ghost zones at the level boundary. Since level l evolves with time step Δt_l and level $l+1$ with $\Delta t_l/2$, we need to interpolate coarse grid values both in space and in time. Previously, this interpolation was always done on the rank holding the coarse level and only the interpolated ghost zone values where communicated. Within the new approach, we instead communicate the spatially interpolated coarse grid values needed to compute *all* fine grid ghost zones. This way only one communication step is necessary and the level boundary filling can then happen locally (and without extra communication) on the fine level. We found that this leads to an improvement of 20% on our local workstation.

Final scaling

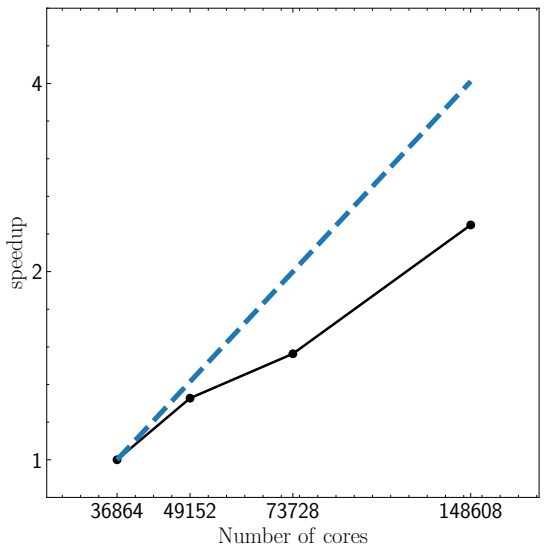

Figure 4: Strong scaling of GReX on SuperMUC-NG. The change in runtimes is reported via the effective speed-up. The dashed line corresponds to a perfect ideal scaling.

As the final part of the project, the scaling performance of the optimized code was assessed on SuperMUC-NG as part of the Extreme-Scaling-Workshop. In accordance with the FRONTERA scaling results, it was found that the code scales reasonably well to about $50,000$ cores, but then degrades when going to $\simeq 70,000$ cores. When further doubling the number of cores the scalability again improves, allowing for the code to successfully run on $\simeq 150,000$ cores. See also Fig. 4 for further details. While the reason for the loss of optimal scaling is not yet understood, it might likely be associated with a different load-balancing at intermediate core counts for the specific problem setup. We plan to investigate this in more detail in the future.