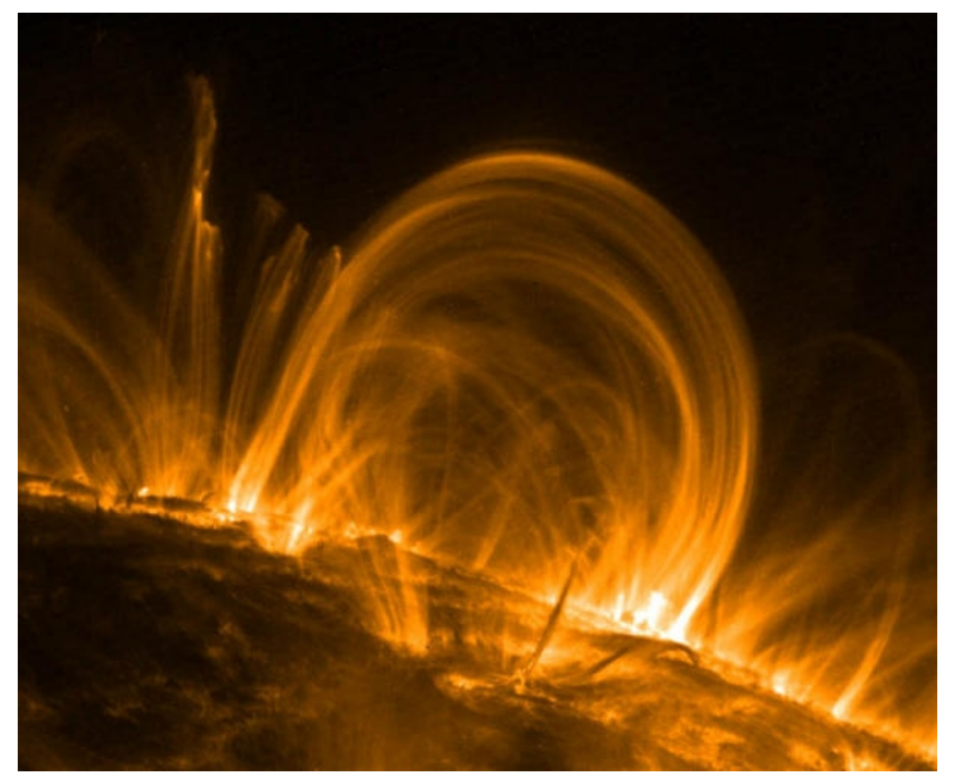
Towards a Coupled Multi-Scale Model Of Magnetic Reconnection

Michal Charemza, Tony Arber
Centre for Fusion, Space and Astrophysics, Physics Department, University of Warwick, Coventry, CV4 7AL

Abstract

Magnetic reconnection in the solar corona is a process that is thought to account for some of the energy release in solar flares. It is governed by changes on both small scales, where the kinetic model of the plasma applies, and very large scales, where the MHD model of the plasma applies. A code that couples the large and small scale models should be able to accurately simulate reconnection, while keeping the run time to feasible levels. However, this is still an computationally intensive task and needs to be run on parallel computers.

Magnetic Reconnection



TRACE image of solar corona using 195Å bandpass.

What is Magnetic Reconnection?

- It is the breaking and reconnecting of magnetic field lines in a plasma.
- Thought to happen in the solar corona
- Releases energy from the magnetic field and converts it into heat and kinetic energies.

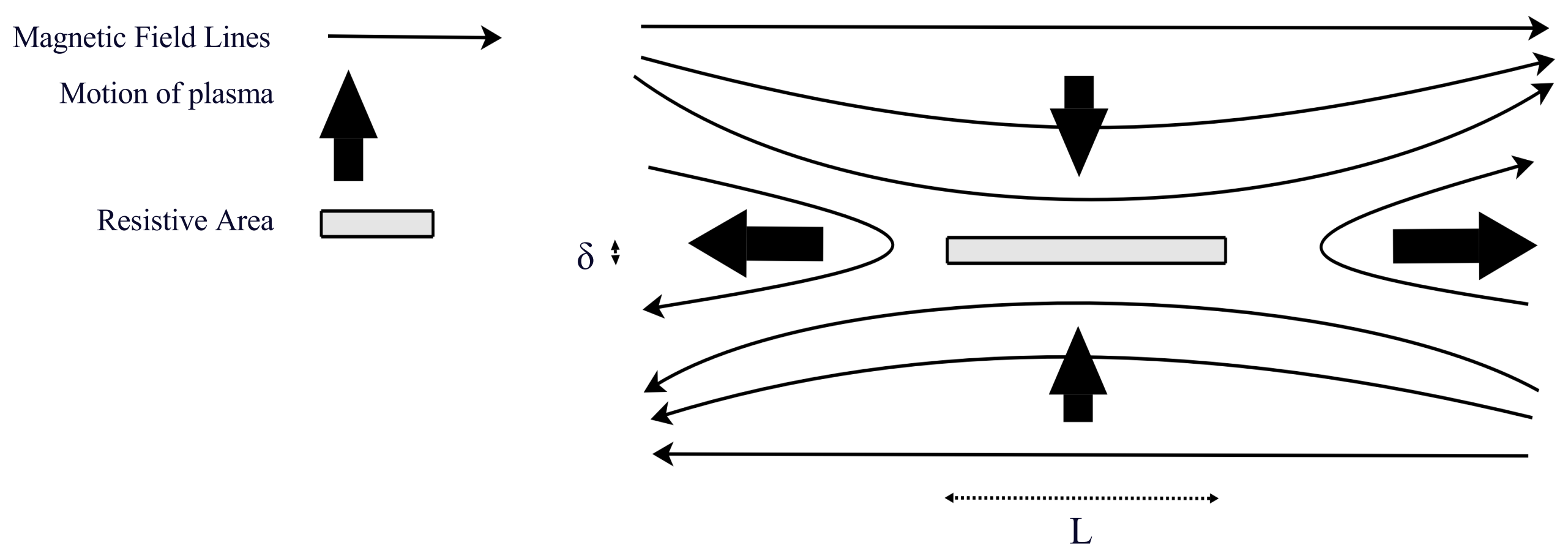
MHD

- A single fluid description of a plasma, describing its motion, density, temperature and magnetic field.

Requirements for MHD Reconnection

- **Resistivity:** According to Alfven's Theorem, for a standard MHD plasma magnetic field lines cannot move through the plasma. In a resistive MHD model they can, so one must use a model that allows resistivity to model reconnection.

Scales of Magnetic Reconnection



A basic view of magnetic reconnection. Note that outside the resistive area magnetic field lines are fixed to the plasma.

- For a solar flare L ~ 10Mm.
- For a solar flare, depending on the model of reconnection, δ ~ 5m
- This is over **6 orders of magnitude difference**: this is why it is difficult to simulate reconnection using coronal values.

Kinetic Model

- A kinetic model of a plasma models the distribution functions of the electrons and ions: typically using the Vlasov equation(s).
- Kinetic effects important on lengths λ_i ~ 10m: the ion inertial depth.
- In solar corona $\lambda_i > \delta$, so kinetic processes affect reconnection.
- Very computationally intensive to simulate: thus can only be used in a small area.

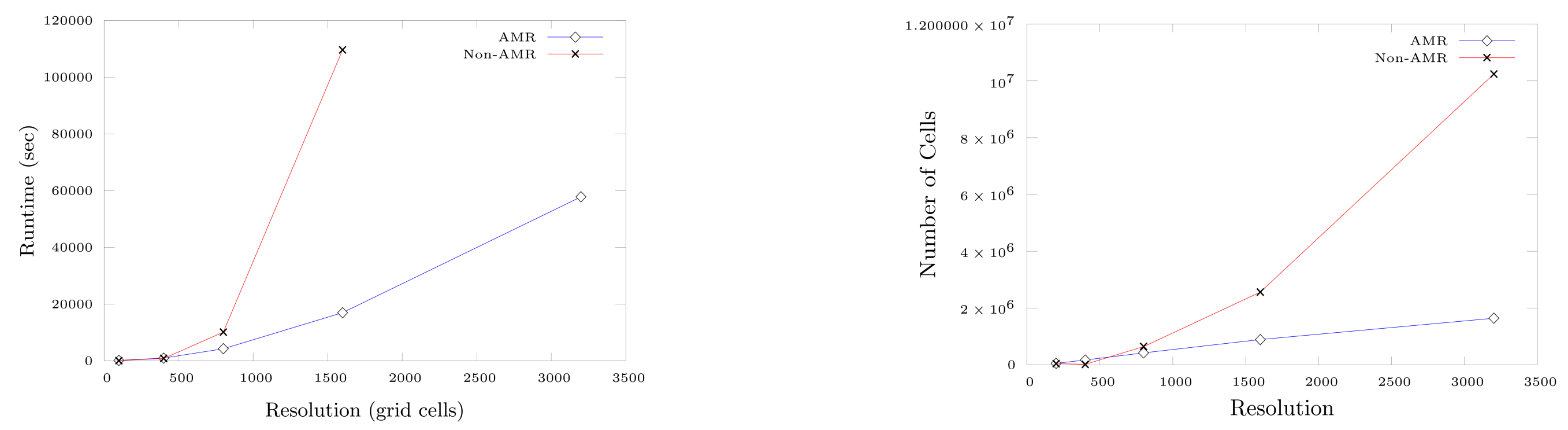
Adaptive Mesh Refinement

What is a Adaptive Mesh Refinement?

- AMR is a technique of solving partial differential equations numerically, but using resolutions that are not constant in space and time. Typically a higher resolution is used where there is a high change in the solution variables.

Advantages

- **Faster** than comparable non-AMR code.
- **Less Memory Used**: Lower memory usage than comparable non-AMR code.



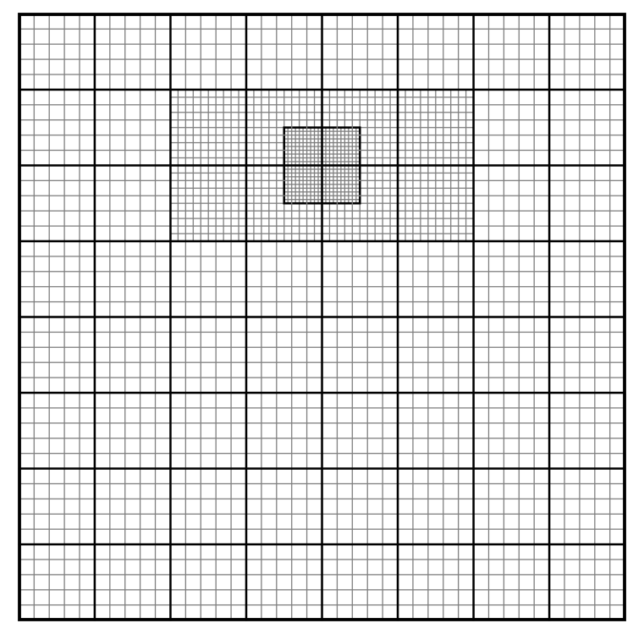
How runtime and maximum memory usage vary for the MHD Orszag-Tang problem when run on the `Larma` code with AMR, and when run using a comparable non-AMR run [2].

Problems with AMR Codes

- **Complex code**: Thus it takes time to write and maintain.
- **Computational overhead**: An internal structure of the refinement levels must be maintained. Also, solution variables must be passed between refinement levels, which requires interpolation.

AMR MHD Code (`Larma` [2])

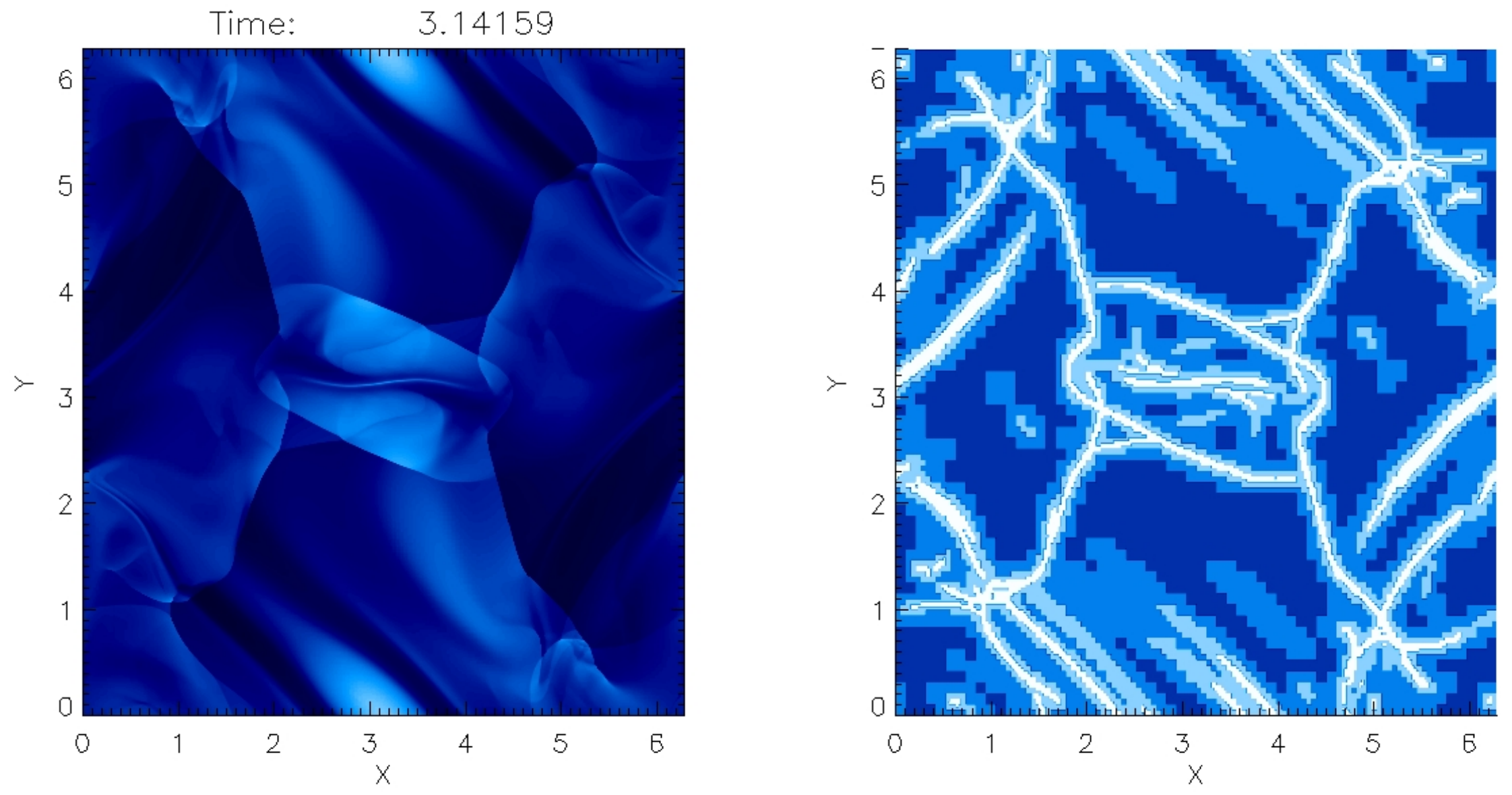
- Based on the 2D MHD solver `Lare2D` [1].
- **Block structured**: applies rectangular "patches" where the solution variables are in twice the resolution as the courser level. These patches can then be refined further with additional patches, and so on up to a set limit of refinement.
- **Data Transfer Between Patches:** Throughout the code solution variables are copied between patches. This has implications for a parallel version of the code as this requires communication between processors.



Example of how `Larma` places patches on a 2D grid

Why is AMR useful to this project?

- The finest patches will be solved using a parallel kinetic Vlasov solver when simulating magnetic reconnection.



Test run of Orszag-Tang Problem using `Larma`. The left hand side shows a density plot, and the right hand side shows how level of refinement varies in space: lighter shades show higher resolution areas.

Parallel AMR

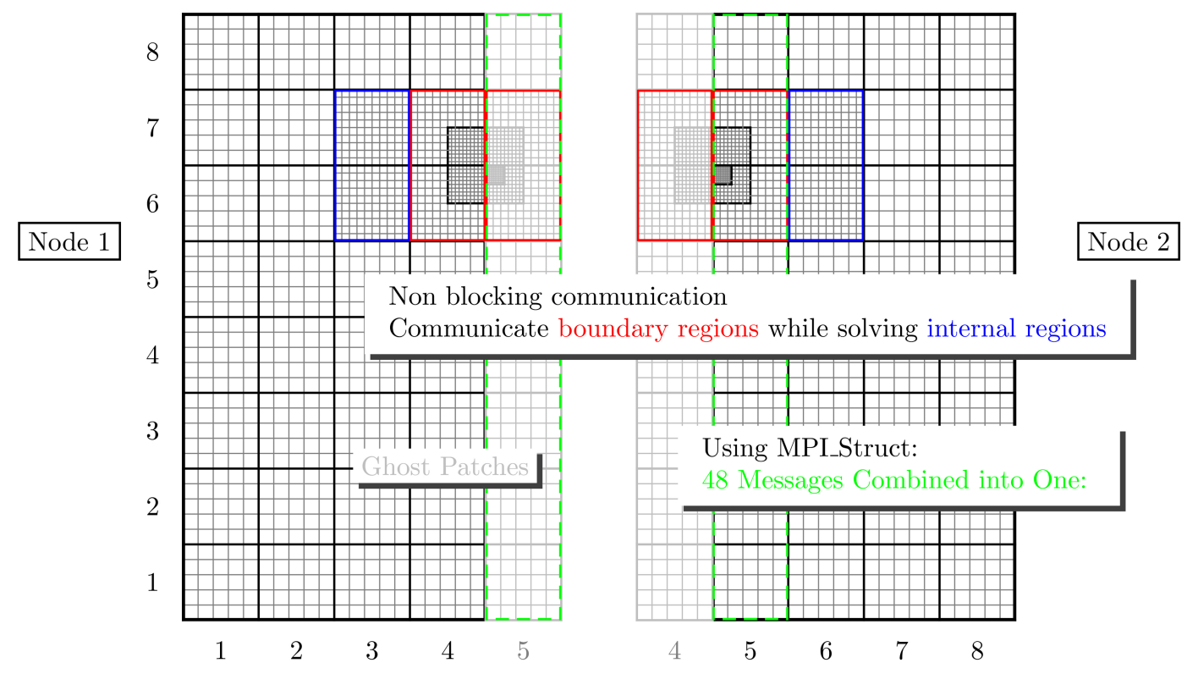
What is a Parallel Code?

- A code designed to run using many processors at the same time.
- **Linear Scaling**: Ideally with N processors the code is N times faster than when run on 1.
- **Super-Linear Scaling**: Due to cache effects a parallel code run on N processors can be even more than N times faster than when run on 1.

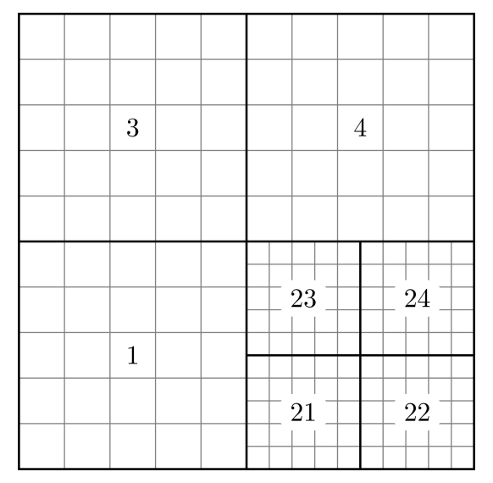
Problems with Parallel Codes

- **Complex Code**: thus it takes a long time to design, write, debug, and maintain.
- **Communication Time**: The processors (usually) need to communicate with each other at runtime.
- **Load Balancing**: To most efficiently use the processors, each should have approximately the same amount of work to do. This can be difficult to achieve due to code complexity.

Parallel AMR Code (`Parlarma`)


Node 1
Node 2
Non blocking communication
Communicate boundary regions while solving internal regions
Ghost Patches
Using MPI_Struct:
48 Messages Combined into One:
1
2
3
4
5
6
7
8

- Based on MHD AMR `Larma` Code [2].
- **Domain Decomposition:** The global domain is split between nodes into equal sizes.
- **Ghost zone**: Beyond the boundary of each node's domain a strip of "ghost patches" and "ghost cells" is maintained with the current solution.
  - **AMR Coordinates:** Each process must communicate when/where/at what refinement level to place patches near it's boundary,
  - **MPI Structures:** To reduce communication time, the solution along the boundary is combined into one message. For a patch of 10x10 cells, for a single message on mhdcluster the latency is ~ 1/3 of the total time for a message to be sent (using [3] for timings).
- **Non-blocking communication** (Not yet implemented): Communication of boundary areas can run at the same time as solving of internal areas.



Example of how AMR coordinates change in patches

Current Problems with `Parlarma`

- **Does not scale well**: Efficiency on a run using 64 processors can be as low as 20%. This is very far from linear scaling.
- **Not load balanced**: On a test run of the Orszag-Tang problem on 64 processors on a 200x200 grid, over 38% of the run time was lost due to load imbalance.

Further Work

- Implement non-blocking communication for `Parlarma`
- Implement a load balancing algorithm for `Parlarma`.
- **Couple to parallel Vlasov code**: The finest patches in `Parlarma` will be solved using a kinetic Vlasov model of a plasma.

References

[1] T. D. Arber, A. W. Longbottom, C. L. Gerrard and A. M. Milne. 'A staggered grid, Lagrangian-Eulerian remap code for 3D MHD simulations' Journal of Computational Physics, 171 (2001) 151-181.

[2] M. Haynes, 'Block Adaptive Mesh Refinement for a Lagrangian Remap Code', PhD. Thesis, University of Warwick, 2005

[3] SKaMPI MPI Benchmarker http://liinwww.ira.uka.de/~skampi/

centre for fusion, space and astrophysics

THE UNIVERSITY OF WARWICK

EPSRC
Engineering and Physical Sciences
Research Council