

Parallel Mesh-Adaptive Fluid Simulations
Using a
POSIX-Threads / MPI Hybrid

Jürgen Dreher

University of St. Andrews



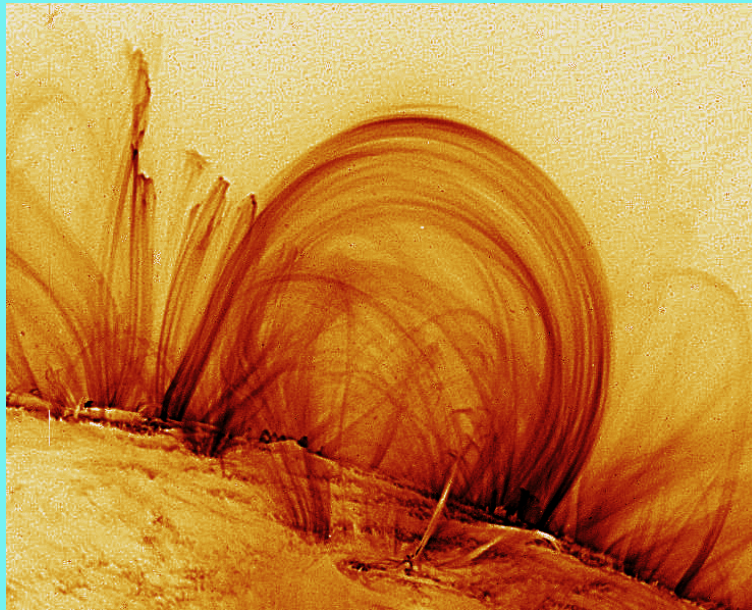
Parallel mesh-adaptive fluid simulations using a POSIX-threads / MPI hybrid

- Why do we need computers in St. Andrews ?
- Adaptive Mesh Refinement: How does it work ?
- General considerations
- Multi-thread implementation
- MPI-distribution and hybrid parallelism, load balancing
- Outlook



Investigations into Solar and magnetospheric plasma physics

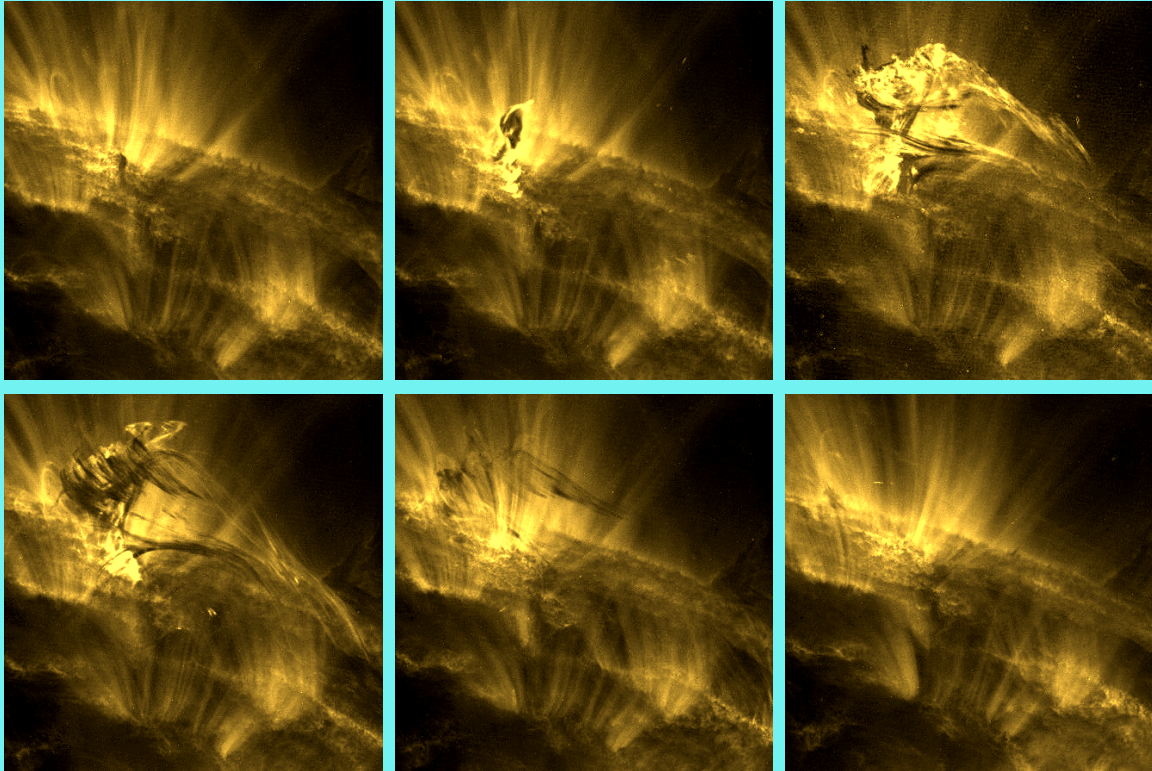
Example: Magnetic loops in the Solar corona (TRACE satellite)



Observed: Transversal oscillations, longitudinal waves



Eruptions, large scale instabilities





Computations needed, e.g. for time-dependant simulations in Magneto-Hydrodynamics

$$\begin{aligned}\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) &= 0 \\ \rho \left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} \right) &= \mathbf{j} \times \mathbf{B} - \nabla p \\ \mathbf{E} + \mathbf{v} \times \mathbf{B} &= 0 \\ \frac{\partial p}{\partial t} + \mathbf{v} \cdot \nabla p &= -\gamma p \nabla \cdot \mathbf{v} \\ \frac{\partial \mathbf{B}}{\partial t} &= -\nabla \times \mathbf{E} \\ \nabla \times \mathbf{B} &= \mu_0 \mathbf{j} \\ \nabla \cdot \mathbf{B} &= 0\end{aligned}$$

Initial value problem:

- specify initial configuration
- follow temporal evolution in numerical simulation



- ⇒ test idealized theories for complex systems/geometries

- ⇒ examine fundamental processes

- ⇒ derive implications for key questions, e.g.
 - heating of Solar corona
 - acceleration of Solar wind
 - acceleration of particles in auroral zones
 - ...



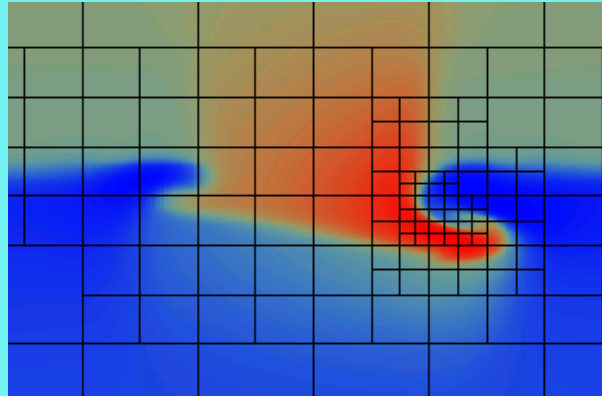
Multi-scale phenomena are often difficult to investigate in simulation

- small scales require fine numerical grid
- fine grid in entire domain too expensive

Examples:


- steepening of MHD waves, shock formation
- “magnetic reconnection”: strongly localized effects change large scale magnetic topology

⇒ Use adaptive mesh refinement to cover multiple scales with limited computational costs



Basic idea:

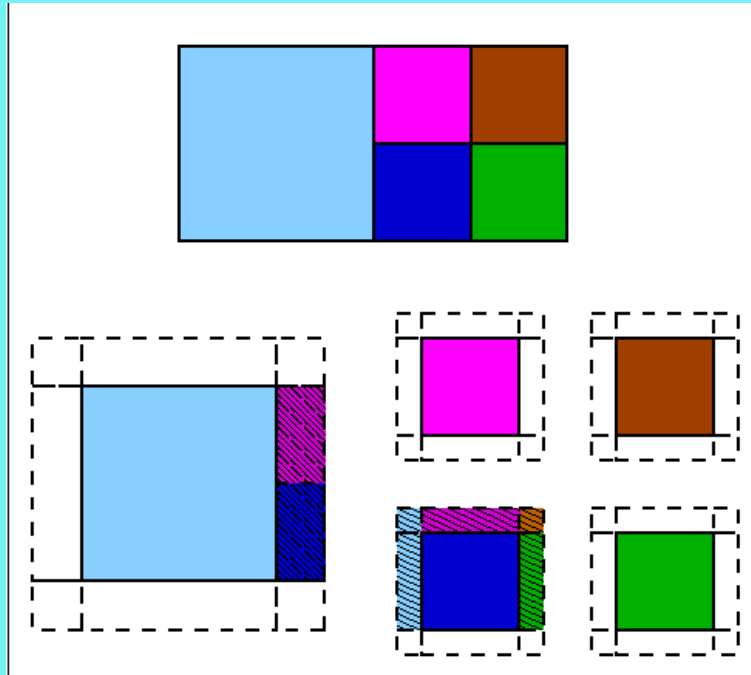
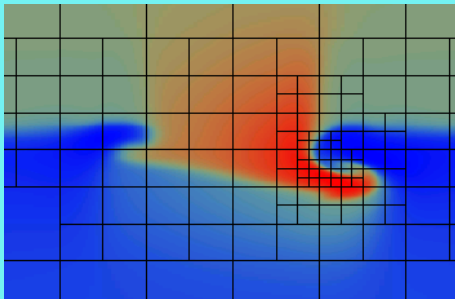
- use high spatial resolution where necessary
- create hierarchy of grids with different resolutions
(here: quad-tree / oct-tree by recursive bisection of grids)
- adjust grid coverage dynamically to solution

Example:  Magnetic reconnection in 2D, colour: electrical current density



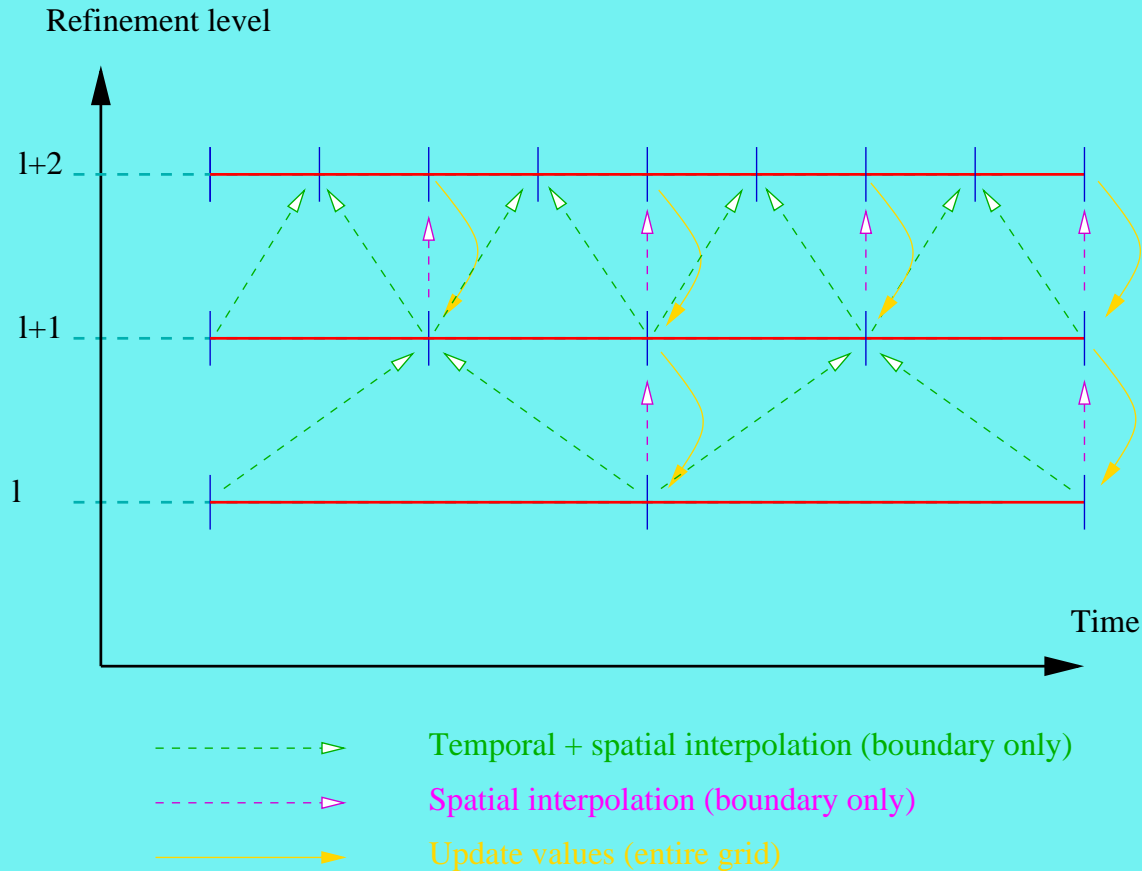
Grids are not entirely autonomous, they must exchange e.g. boundary data
⇒ inter-grid communication necessary

“Ghost cell” strip around grids





Inter-grid communication between refinement levels within one time step:





Adaptive Mesh Refinement

Jürgen Dreher

Project (since Nov. 2001):

Cooperation between Ruhr-Universität Bochum (R. Grauer) and St. Andrews (J. Dreher)

Funded by EU research network “PLATON” (coordinator: Thomas Neukirch, St. Andrews)

- develop flexible AMR framework
(wrt. numerical schemes, physical models, computational platforms)
- suitable for shared and distributed memory platforms
- implementation in C++, evtl. offer Fortran interfaces

Several AMR codes already in use, e.g. Flashcode (U. Chicago), SAMRAI (Los Alamos), commercial CFD products...



Major subtasks:

- advance single grid (90+ % CPU time)
- inter-grid communication (copy, interpolation)
- diagnostics, regridding, output, [load distribution]

Grid are quasi-autonomous

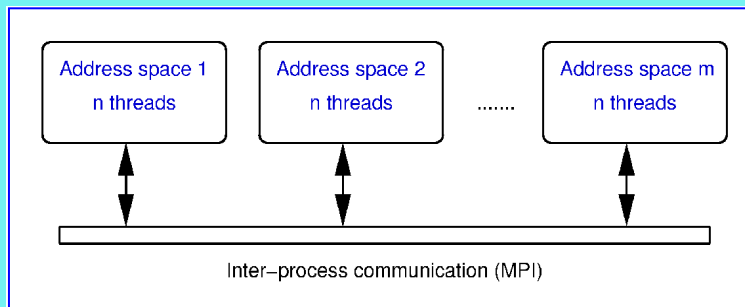
Basis parallelization strategy: **treat individual grid as unit of work**

- + compact in memory, good cache utilization
(5+ % speed-up through domain decomposition alone!)
- + advanced (=moderately expensive) numerical schemes
good ratio computation / communication
- + coarse grained parallelism
 - ! many grids necessary for parallel performance
 - communication pattern unpredictable
 - explicit marshalling/de-marshalling for MPI distribution



Hybrid concept for parallelization:

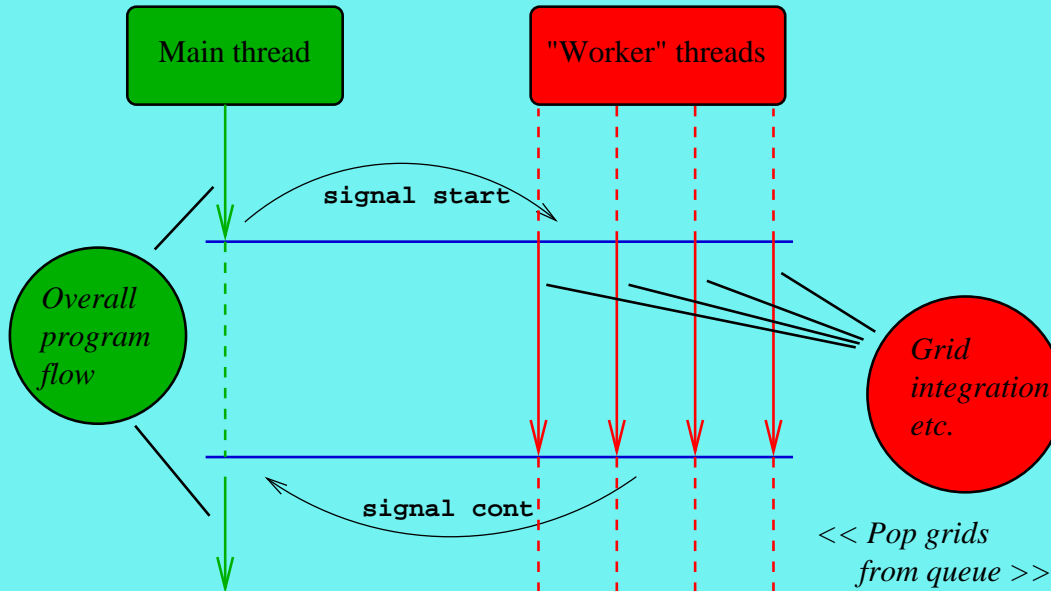
- Use shared memory architectures efficiently with multi-threading
- Use MPI between distributed memory nodes (e.g. cluster), explicit load balancing
- Combine both in hybrid mechanism
⇒ *Reduce network communication and number of computing nodes*





Multi-thread parallelization with standard POSIX-MT (mutex, condition variables, ...)

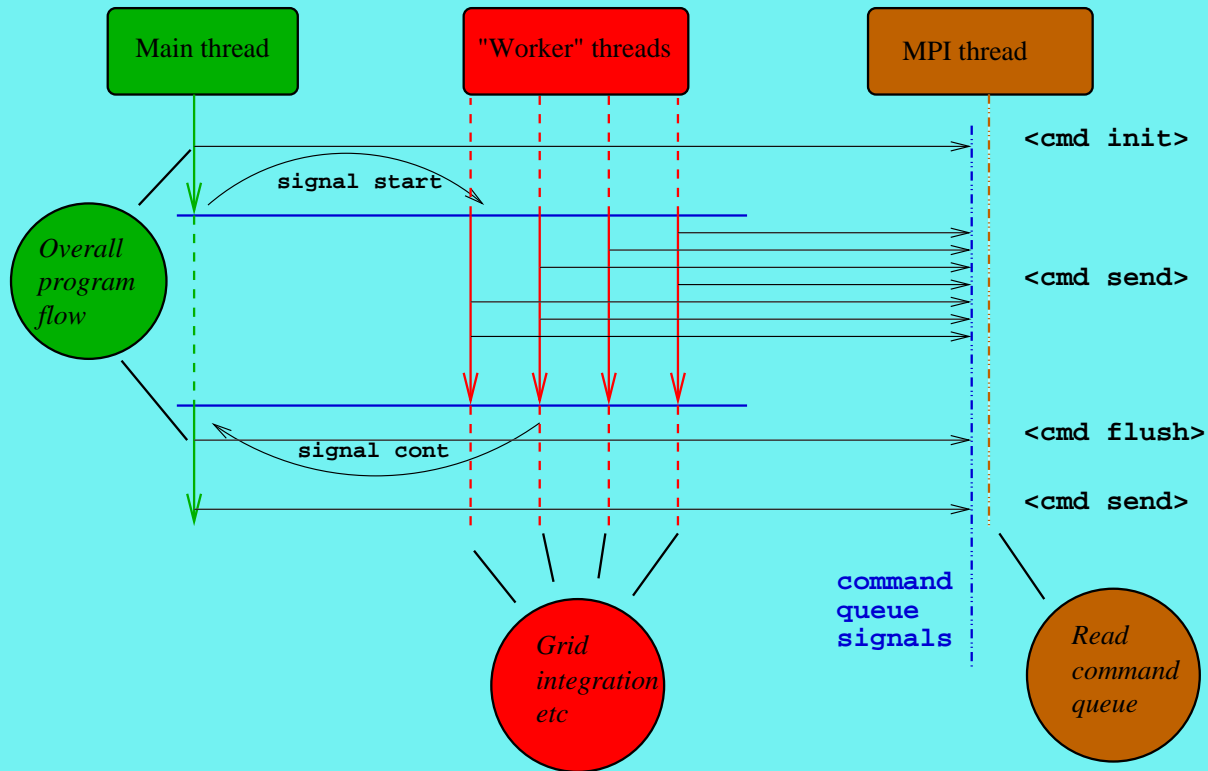
Parallel sections: single grid integration, data copy on same level
All other tasks sequential



Typical speed-up (wall-clock time): Factor 3.2+ with 4 CPU's on DEC Alpha, Linux PC

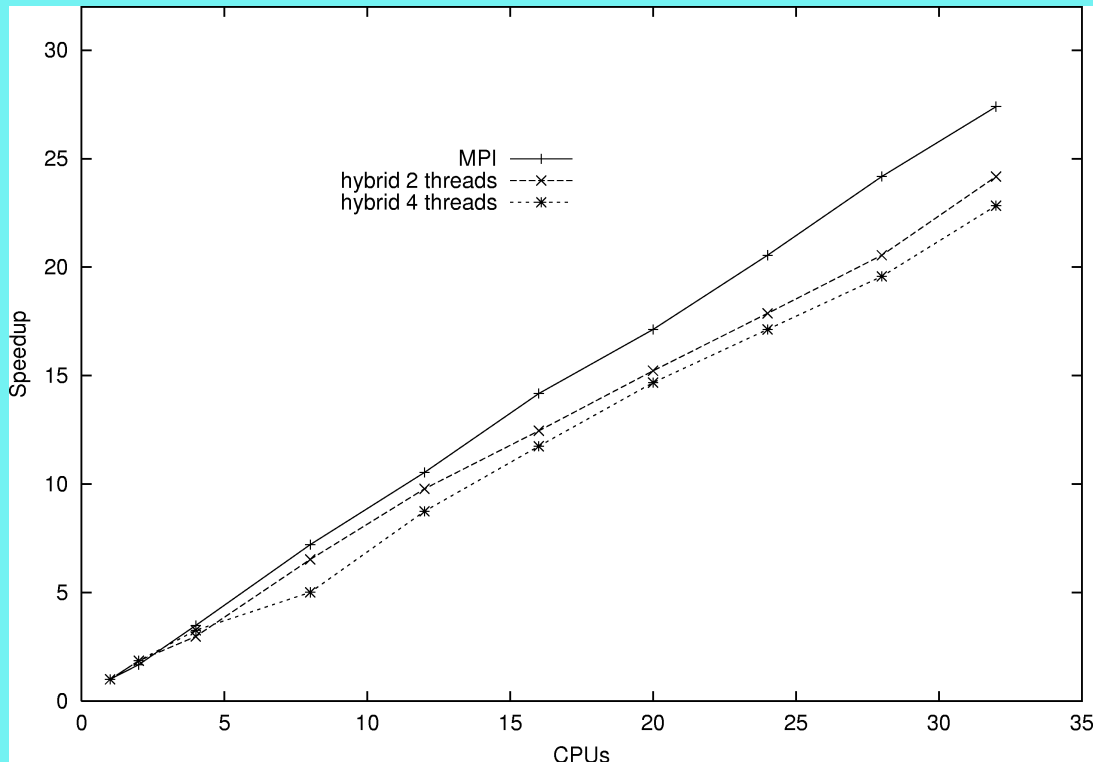


Hybrid parallelization: *Current free MPI implementations not thread safe !*





Scaling on Jülich Linux PC cluster *Zampano* (8 x 4 Pentium III 500 MHz, Myrinet)

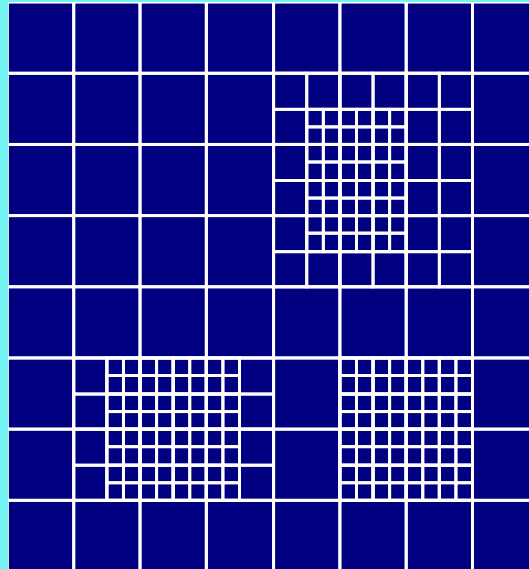


Memory bandwidth bottleneck on PC's (?):

Distributed 4×1 process via Myrinet faster than 1×4 procs via shared memory MPI!



Load distribution:

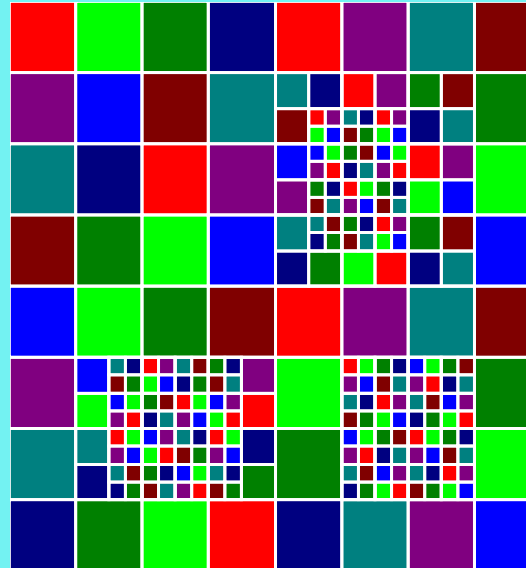


- workload must be balanced among nodes
- ... on each level separately
- fast distribution algorithm
- limit inter-node communication



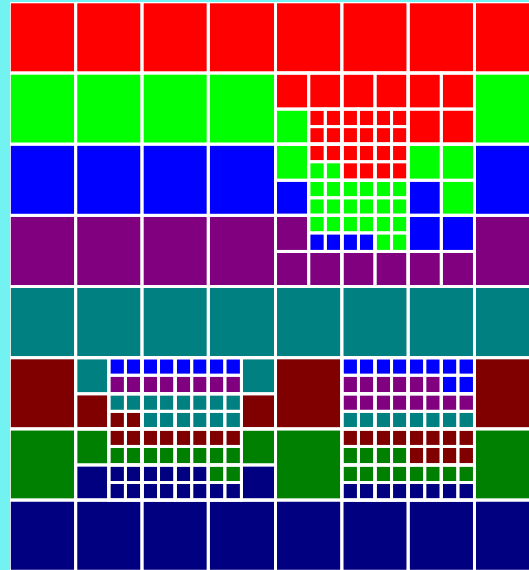
Example: Round-robin

- + load balanced
- + simple to compute
- much communication between neighbours





Better: Row-by-row ordering
Conserves neighbourhood horizontally

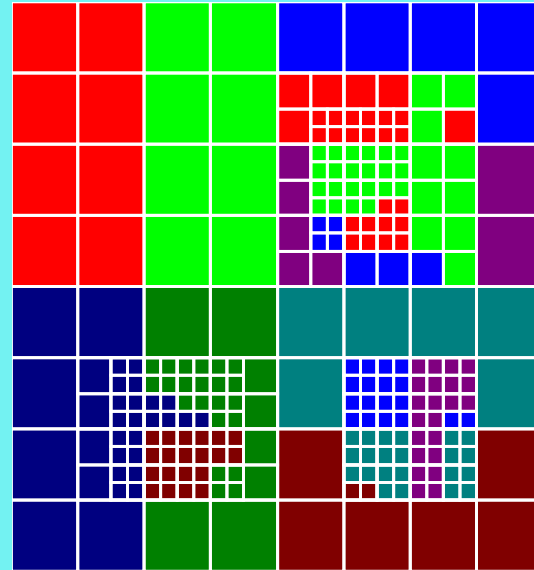
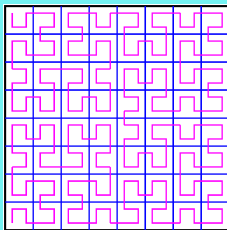
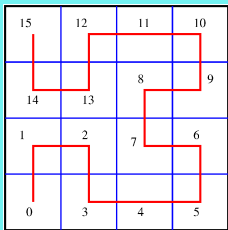




Even better:

Space filling curves tend to preserve neighbourhood in all directions

- not perfect, but reasonably good clustering
- small surface-to volume ratio of clusters





After every grid refinement:

- compute new load distribution
- migrate grids between processors (with threshold for allowed imbalance)

Example: Rayleigh-Taylor instability





- Scaling properties on larger shared memory systems ?
(Soon: IBM Regatta in Jülich, 32 CPUs per node)
- Tuning in MT version possible, more parallel regions, but involved
- When does MT pay off compared to pure MPI ?
- Other strategies, OpenMP vs. POSIX MT?
- Also important: Application to physical problems, aspects like numerics etc.



- Scaling properties on larger shared memory systems ?
(Soon: IBM Regatta in Jülich, 32 CPUs per node)
- Tuning in MT version possible, more parallel regions, but involved
- When does MT pay off compared to pure MPI ?
- Other strategies, OpenMP vs. POSIX MT?
- Also important: Application to physical problems, aspects like numerics etc.

Thank you!