Parallelising Computational Microstructure Simulations for Metallic Materials with OpenMP

Ralph Altenfeld

Markus Apel, Dieter an Mey, Bernd Boettger,
Stefan Benke, Christian Bischof

IWOMP 2011 / Chicago

Agenda

- **The MICRESS application**
  - motivated by material science
- **Phase-field equation solver**
  - Problem description
    - Physics and numerics
    - Data structure
  - Parallelisation solutions
    - First-come-first-serve
    - Graph
  - Performance
  - OpenMP task dependencies
- **Summary and outlook**

Motivation …

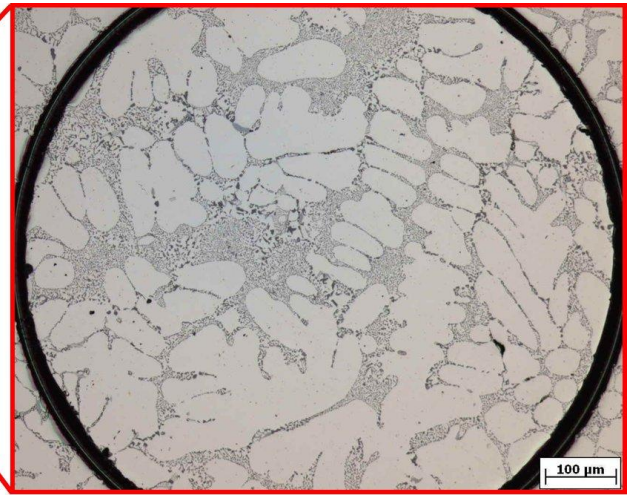
aluminium casting

~ 1 m

automotive component

~ 10 cm

material microstructure



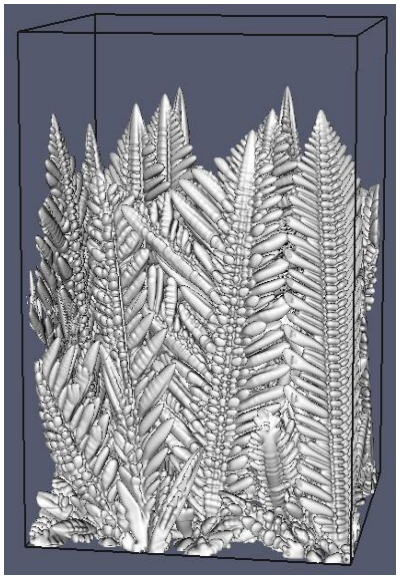
~ 1µm … 1 mm

- For engineering applications material properties are of major interest
- Process determines microstructure determines properties

Phase-field software: MICRESS®

technical alloys

- multicomponent
- multiphase
- polycrystal



applications

- solidification (dendritic, peritectic, eutectic)
- solid state reactions (grain growth, phase transformations, recrystallisation)

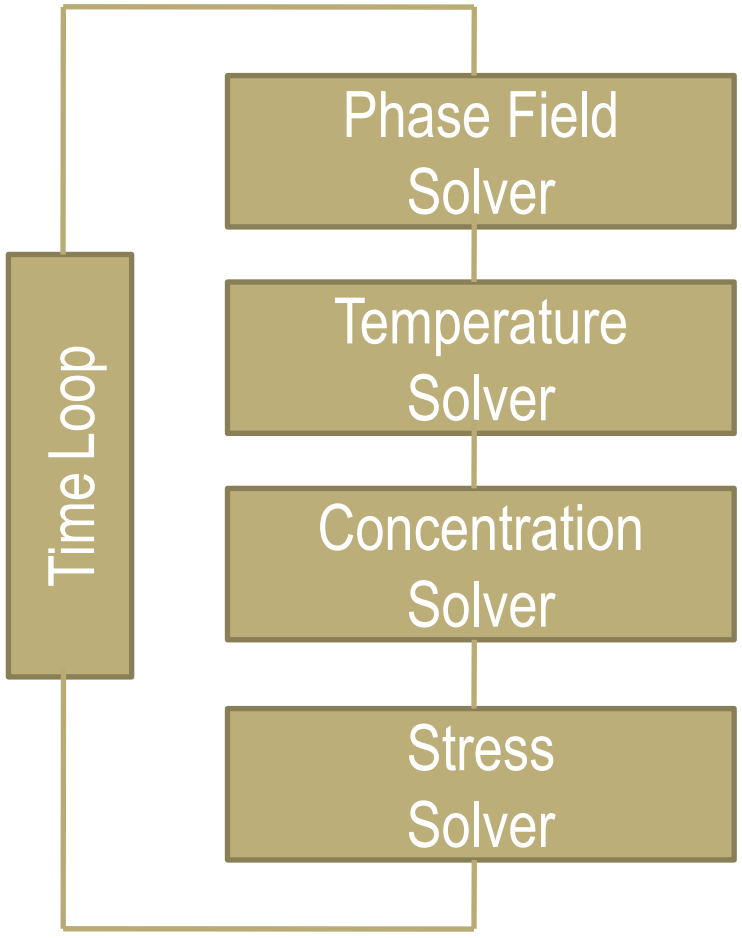
micress®

www.micress.de

special features:

- physically identified, measurable input parameters
- profound treatment of alloy thermodynamics
- computational efficiency at technical length scales in 2D / 3D

MICRESS parallelisation project



- Preceeding diploma thesis
  - Stress solver
    - Bi-Conjugate Gradient stabilized (BiCGstab)
    - Speedup of 5 using 8 threads (Intel Nehalem EP, 2 sockets, 8 cores)
- Interdisciplinary project
  - Computer Science / HPC
    Center for Communication and Computation of the RWTH Aachen
  - Material and process development
    Access e.V. (private, non profit research institute associated with the RWTH-Aachen)
  - Focus:
    - Testcase 'Carbonisation' stressing concentration and phase field solver

Grain growth: phase-field equation

Multiphase-field equation:

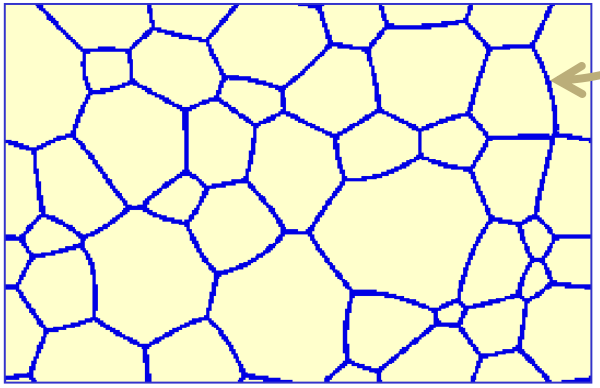
$$\dot{\phi}_\alpha = \sum_\beta^n \mu^*_{\alpha\beta}\left[\sigma^*_{\alpha\beta}\left(\phi_\alpha \nabla^2\phi_\beta - \phi_\beta \nabla^2\phi_\alpha + \frac{\pi^2}{2\eta^2_{\alpha\beta}}(\phi_\alpha - \phi_\beta) + \sum_{\alpha\neq\beta\neq\gamma}^n J_{\alpha\beta\gamma}\right) + \frac{\pi}{\eta_{\alpha\beta}}\sqrt{\phi_\alpha\phi_\beta}\,\Delta G_{\alpha\beta}\right]$$

dual interface terms "double_obstacle"

higher interface terms "multi_obstacle"

chemical driving force

Grain structure

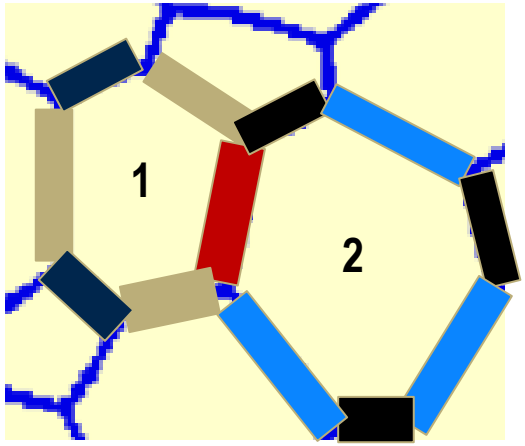


Reference:

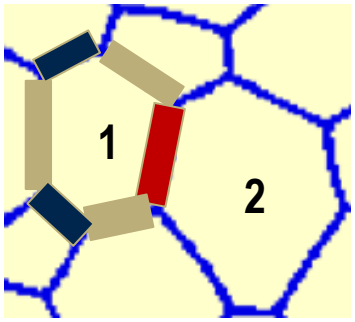
- *Steinbach et al; Physica D 1996*
- *Tiaden et al. Physica D 1998*
- *Steinbach, Pezzolla; Physica D 1999*
- *Eiken, Böttger, Steinbach; Phys. Rev. E 2006*

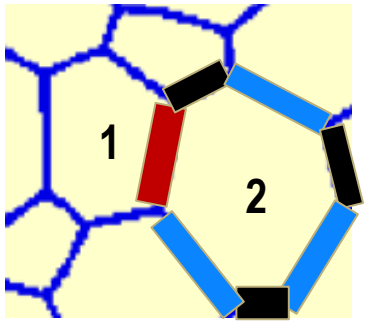
Phase field data structure

Grain structure



Auxiliary grids





Grain interfaces copy for calculations

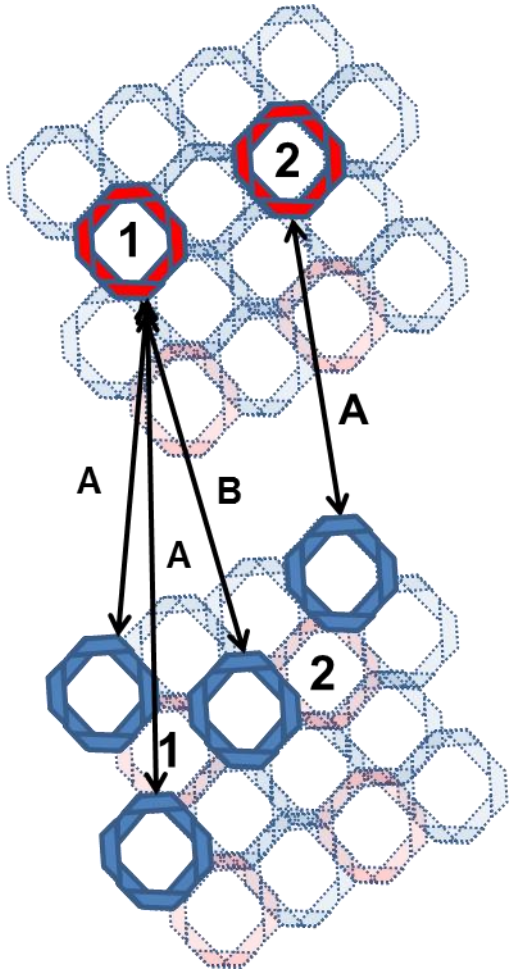
Phase field interface list

Grain 1 Grain 2

- Pairwise solution for neighboured grains on auxiliary grids
- Compute time depends on interface length

Going parallel

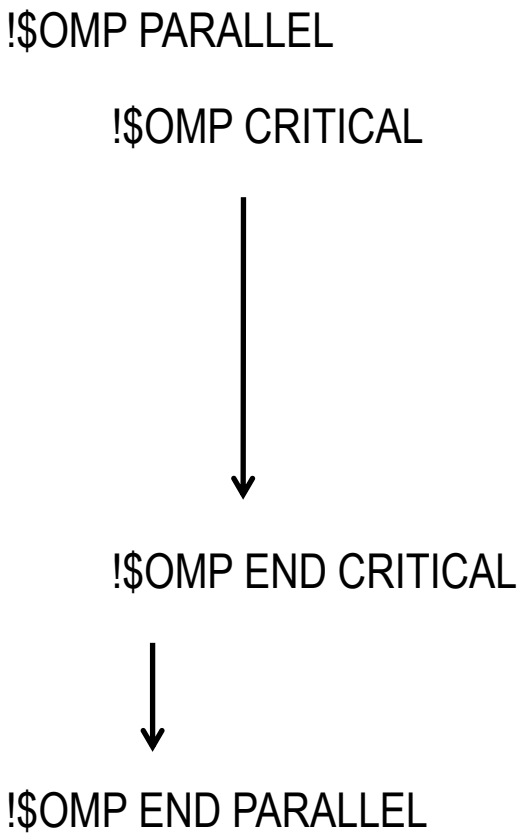
- Here
  - Memory footprint like serial version, i.e. two shared whole simulation area grids
    - Grids also used by other solvers
    - No complex, errorprone code changes
  - Threads compete for room on auxiliary grids
  - Prerequisite for parallel computation
    - Boundary boxes do not overlap
- Problem
  - Find a interface schedule for most efficient use of auxiliary grids



First-come-first-serve solution

One parallel region

- Scheduler as an OpenMP critical region
  - Threads report on finished work
  - Threads look for additional work
    - A shared placement list for each auxiliary grid
    - Check overlap with new grain pair
- Phase field interface compution for a grain pair



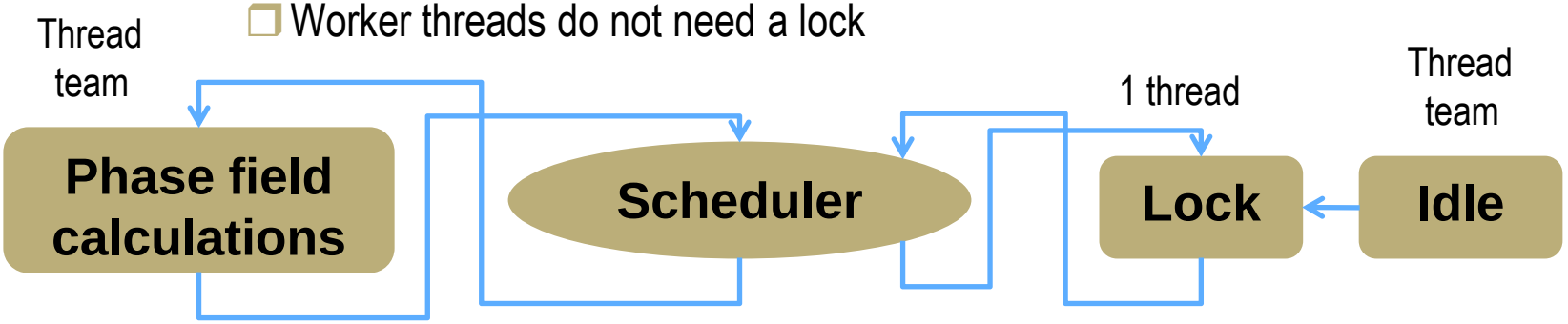
Pros and cons

Pros

- No preprocessing
- No load imbalances (similar to a dynamic OpenMP schedule)

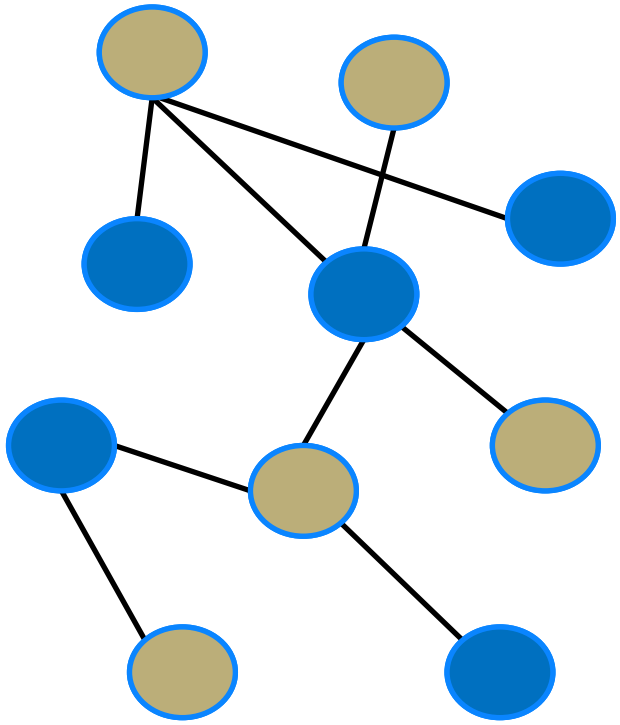
Cons

- Schedule depends on the grain ordering
- Performance suffers from critical region when using too much threads
  - Enhanced solution
    - Protection of critical section entrance with a lock
    - Only one of the idle threads enters, others wait for the lock
    - Worker threads do not need a lock



Graph solution …

- Create/update a graph of all tasks as a preprocessor step
- Task graph definition
  - Node = Task
    - Interface between two neighboured grains
  - Edge = dependence between tasks
    - Grains overlap on auxiliary grids (boundary boxes)
- Find a compute schedule
  - Colour the task graph
  - Tasks of the same colour can be computed in parallel

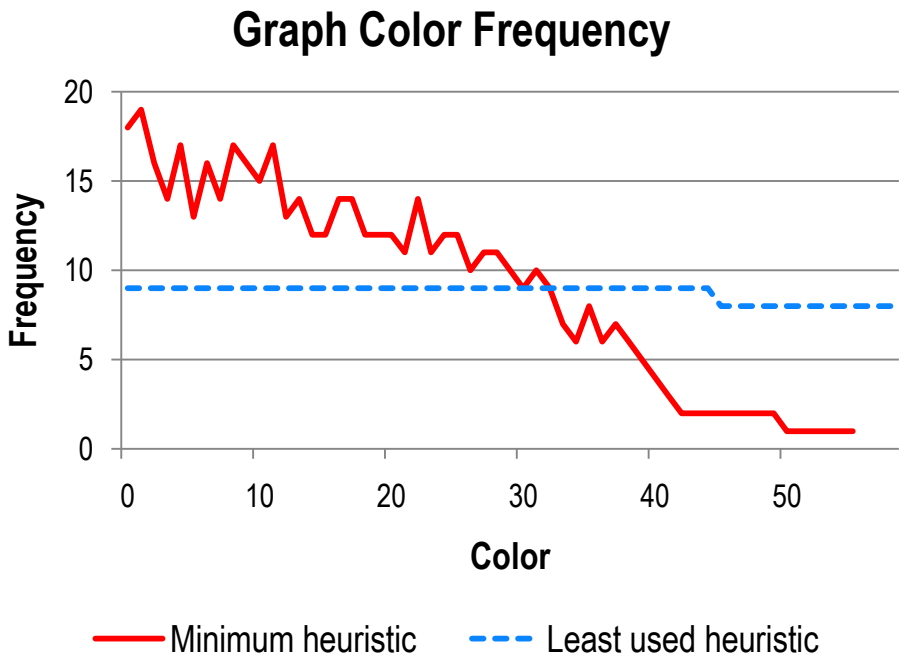


Dependency graph

… Graph solution …

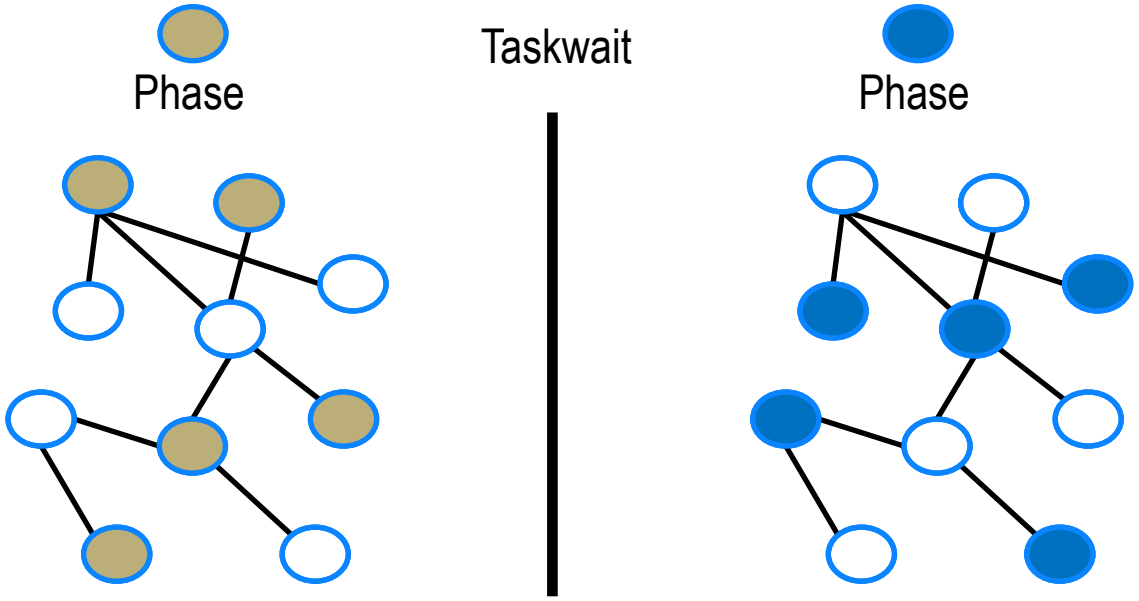
- **Optimal colouring** (i.e. minimal number of colours) is NP-hard
- **Heuristic**
  - Breadth-first traversing of the graph
  - Starting with the node with highest degree
  - Choose a free colour among the already used colours

- **Different ways of colouring**
  - First free colour
    - More frequent colours in the beginning
  - Least used free colour
    - More balanced



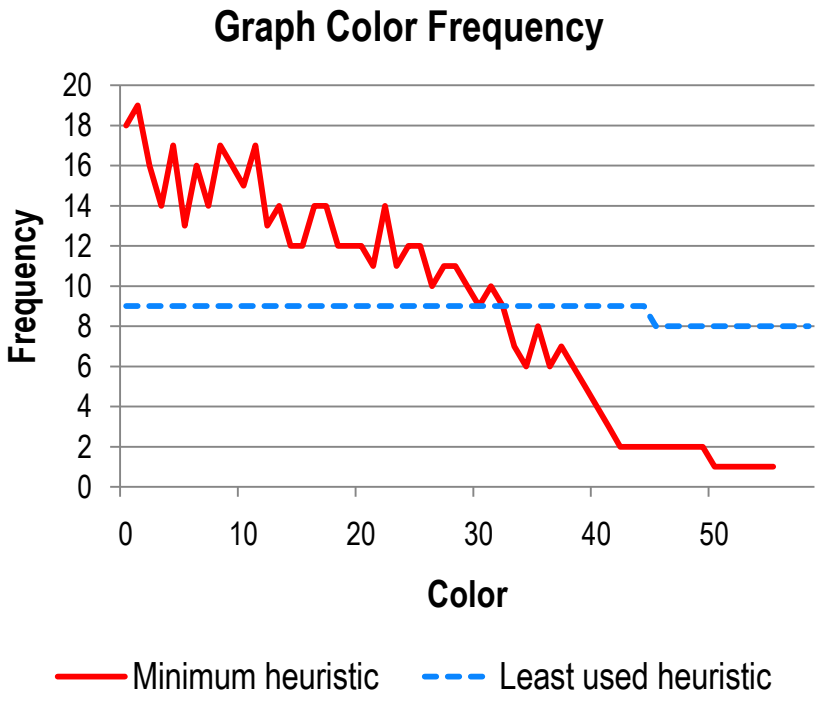
… Graph solution …

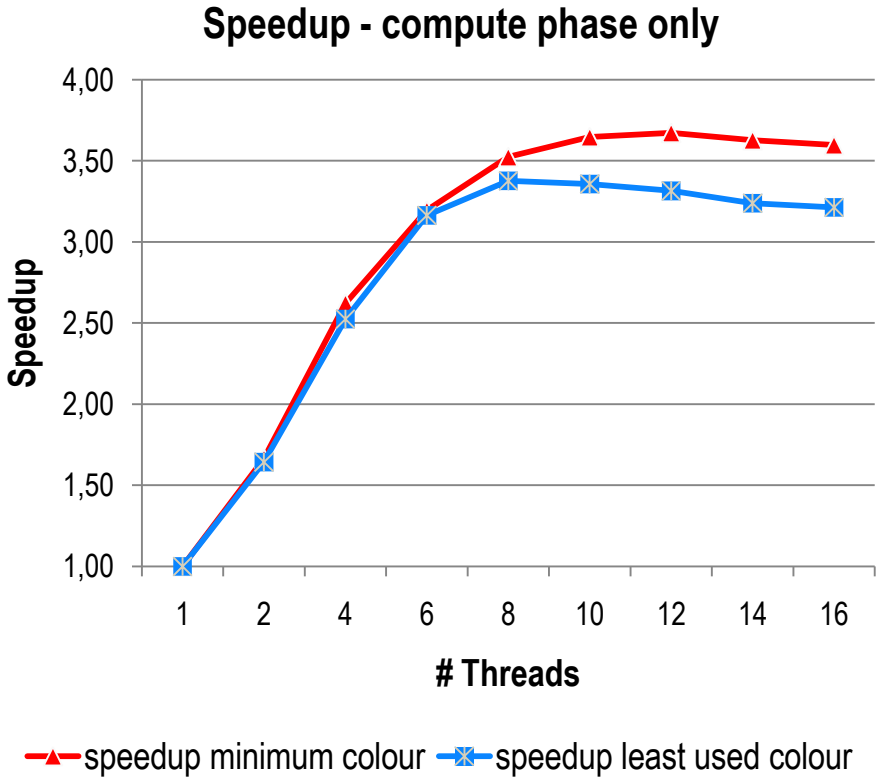
- **Compute phase**
  - Master Thread schedules OpenMP tasks
  - For each used colour in the graph
    - If (graph node == colour)
      - Start a OpenMP task for the according grain pair (interface)
    - OpenMP taskwait



Which colouring is best ?

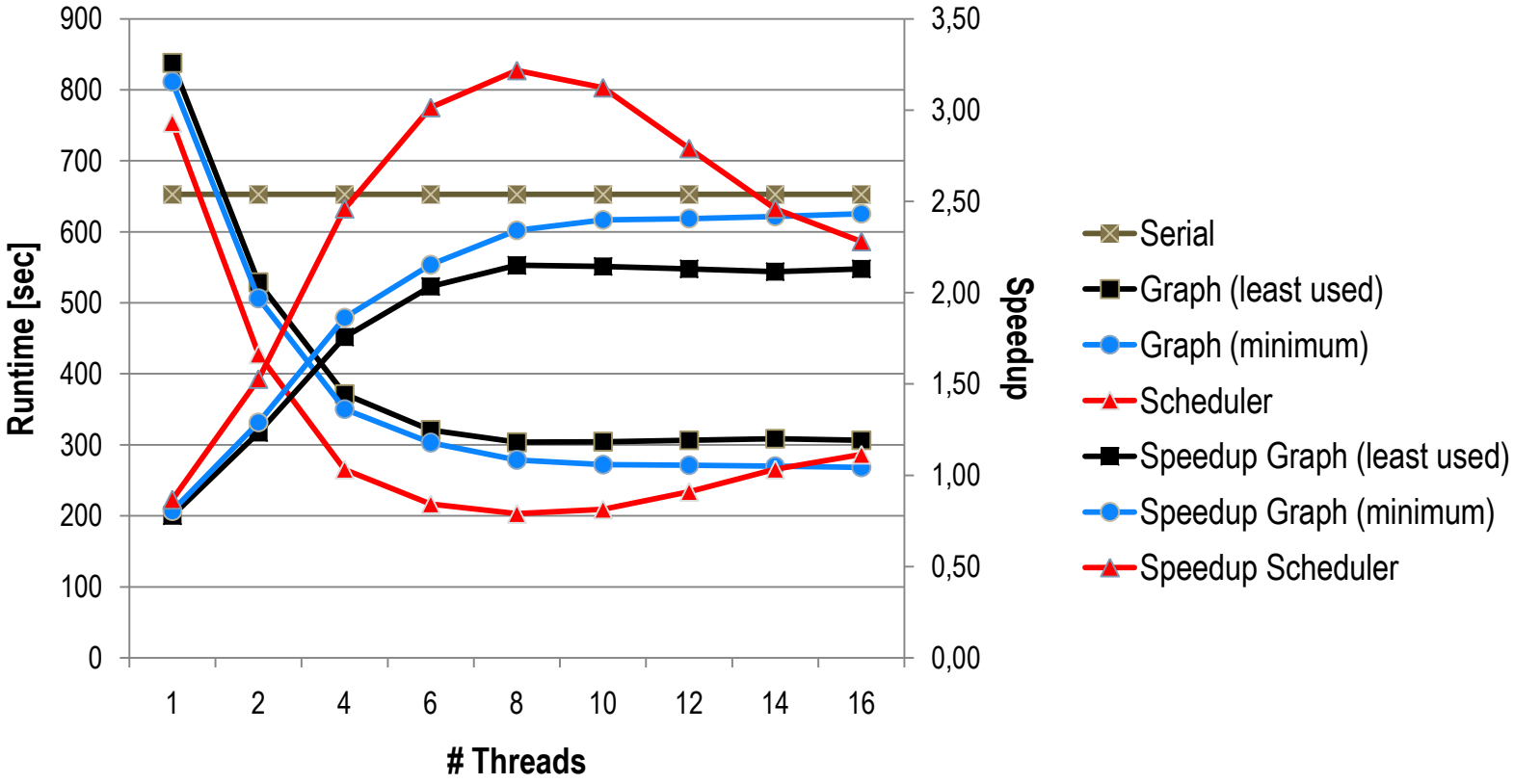
- Tasks runtime depend on interface length:
  - More task per colour results in a more balanced workload
- More tasks in a computation colour phase
  - More threads can do work





Performance of phase field solver (inclusive graph creation)

- First-come-first-serve (scheduler)
  - No preprocessing overhead
  - Best for this problem size
- Graph solutions
  - Less overhead in computation phase
  - Graph creation/update overhead dominates (still sequential)



OpenMP Task Dependencies

- **OpenMP enhancement**
  - Possibility to define and submit dependent OpenMP tasks
- **Submit task while graph traversing**
  - Report finished status of tasks
    - A designated flag for each submitted task
  - Dependencies
    - A list of above flags for a new task
  - Kind of specialisation of data clauses A. Duran proposed 2008
- **Pros**
  - No graph colouring necessary
  - Benefit from efficient scheduling algorithms of the OpenMP runtime system
- **Cons**
  - OpenMP runtime system doesn‘t know the kind of problem

Summary and outlook

- **Phase-field solver problem**
  - First-come-first-serve solution
    - Speedup: about 3.2
    - Critical section is a bottleneck when using more threads
  - OpenMP task solution
    - Speedup: about 2.5
    - Efficient handling of higher thread numbers
    - Performance suffers from sequential graph creation/update
- **Improvements**
  - Parallelisation of graph creation / update
  - More sophisticated handling of auxiliary grids
    - Blending in only interfaces, not whole grains
    - Use of smaller private auxiliary grids
  - OpenMP task dependencies in a new standard

Thank you for your attention!
Questions?