A Collision-based Domain Decomposition scheme for Serpent2

Manuel García, Karlsruhe Institute of Technology (KIT)

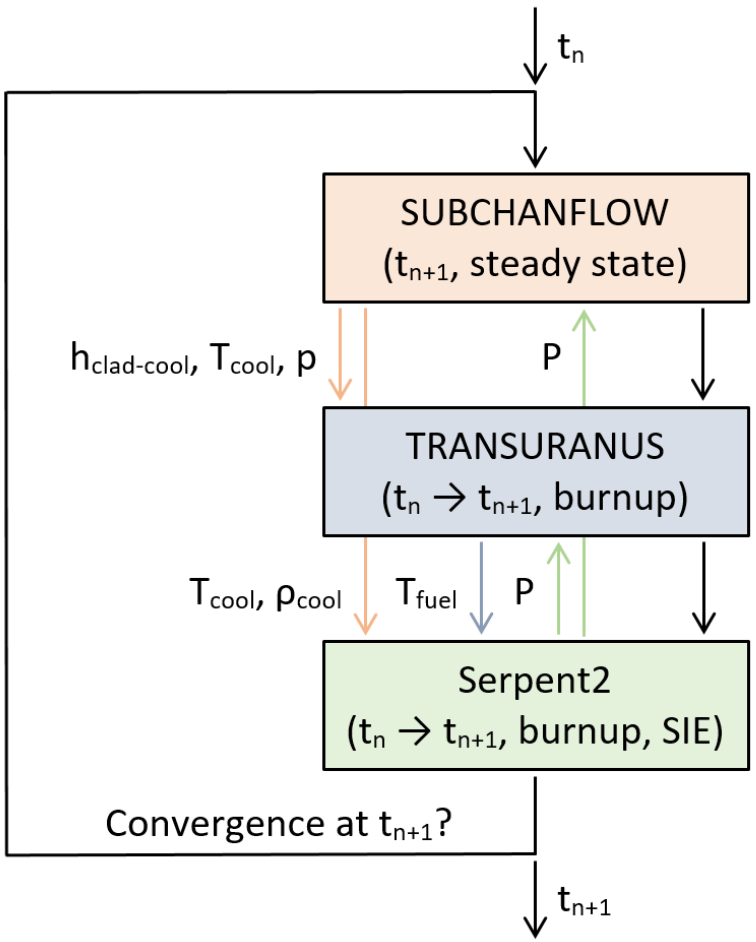
This project is funded by the Horizon 2020- Research and Innovation Framework Programme under the GA Nr.755097

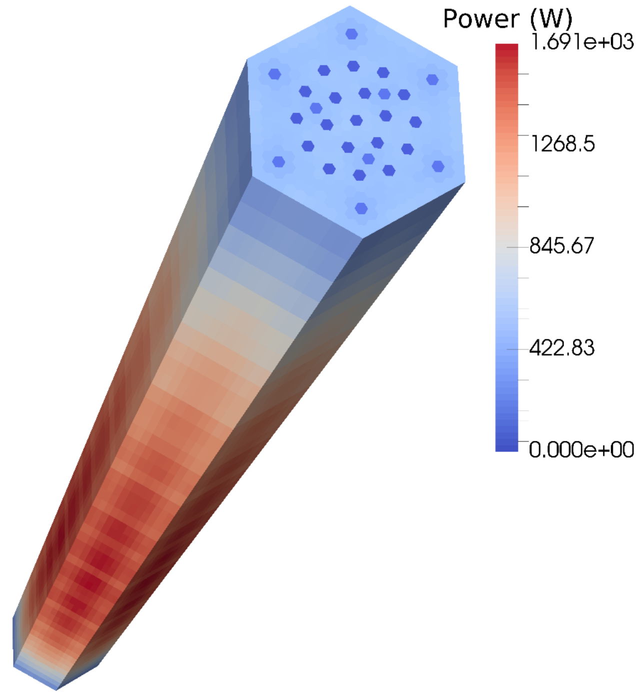
Motivation: the McSAFE EU project

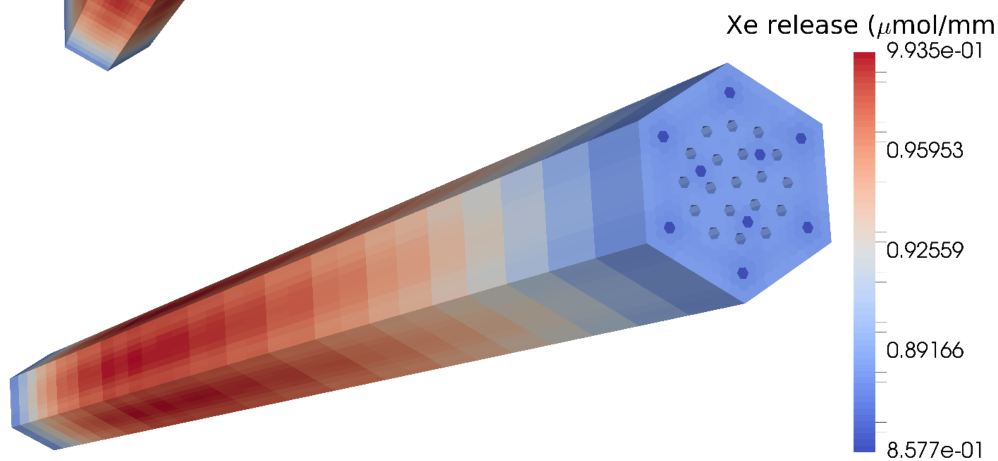
- Participants:
  - 9 research institutions: KIT, VTT, HZDR, JRC, CEA, NRI, KTH, AMEC, DNC.
  - 3 industry partners: EKK, CEZ, EdF.
- High-fidelity multiphysics for safety analysis of LWRs:
  - Monte Carlo neutron transport: Serpent2, Tripoli4, MCNP, MONK.
  - Subchannel thermalhydraulics: SUBCHANFLOW (SCF).
  - Fuel-performance analysis: TRANSURANUS (TU).
- Main developments:
  - Implementation of a Serpent2-SCF(-TU) coupling for steady-state, burnup and transient problems.
  - Optimization of steady-state and transient capabilities for HPC.
  - Optimization for massive (full-core) depletion problems.
- Validation with plant data:
  - PWR-Konvoi.
  - VVER-1000.

Current capabilities: pin-by-pin depletion

- Fully coupled pin-by-pin depletion:

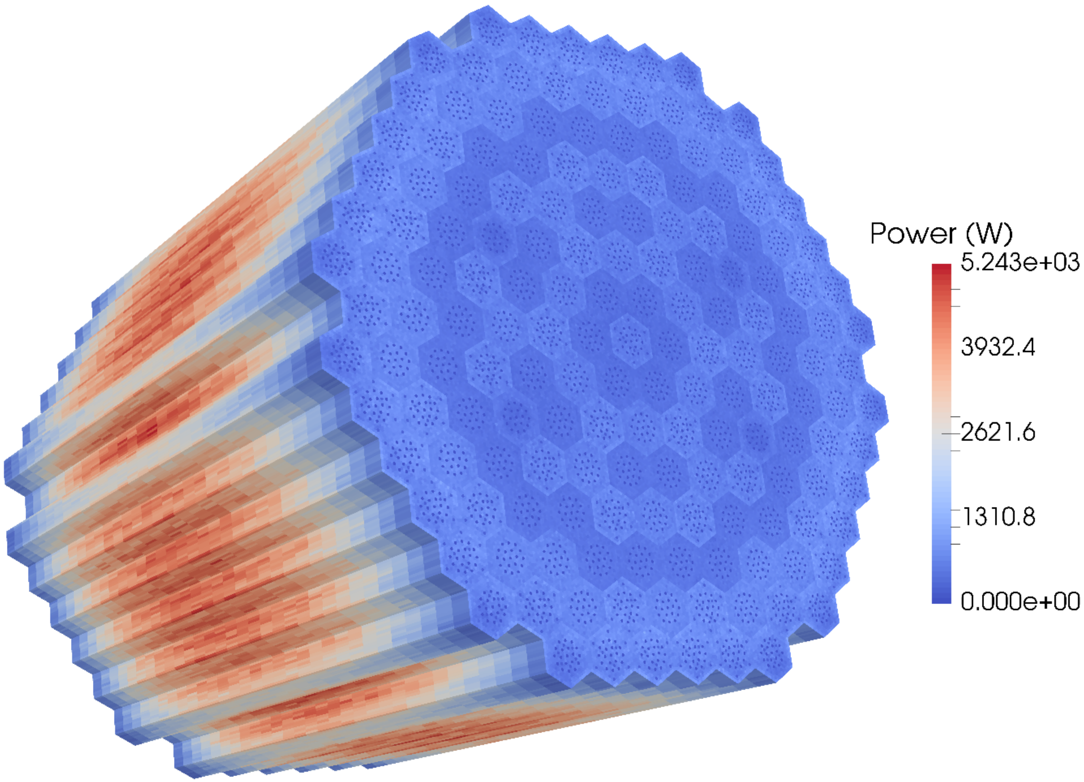


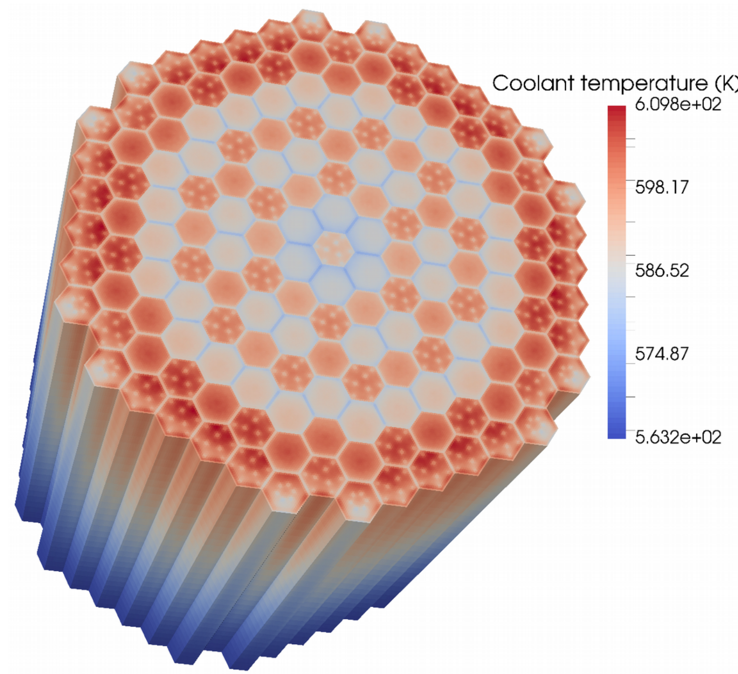




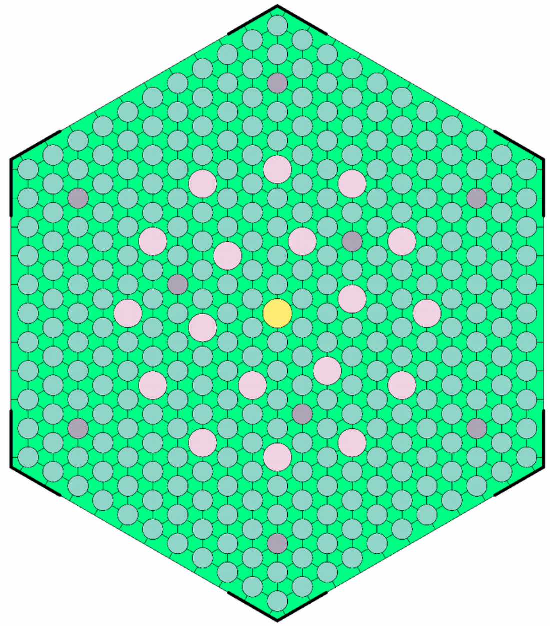
Current capabilities: full-core steady state

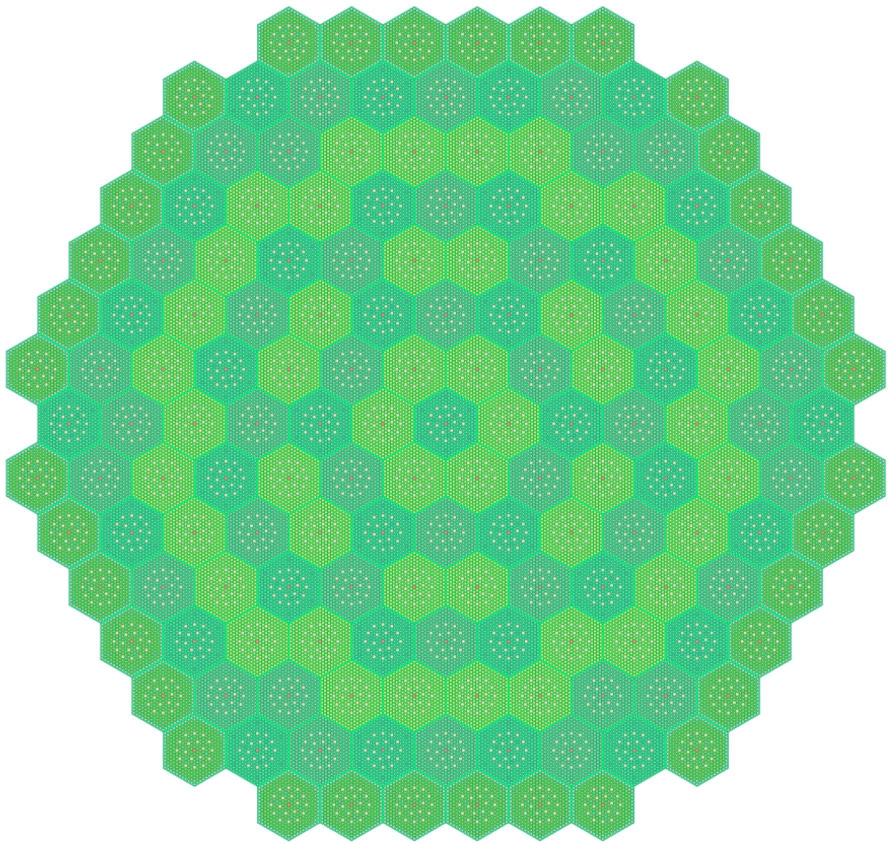
- Full-core steady-state (no burnup) calculations:





Next step: full-core pin-by-pin depletion





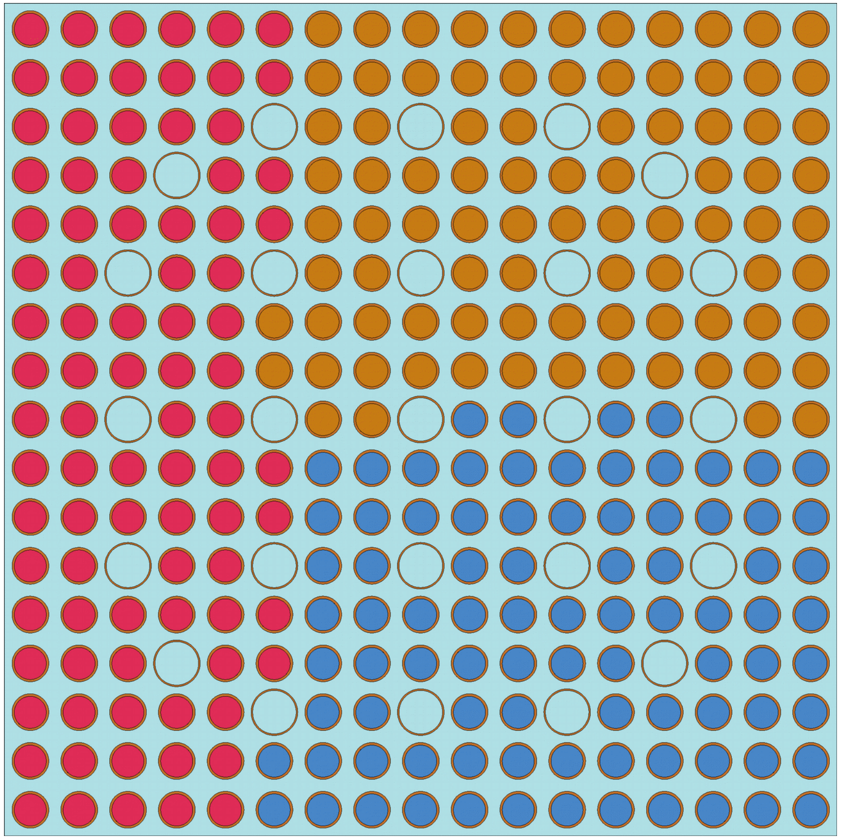
- Main challenges:
  - Calculation time (~10^9 particles) → optimizations in Serpent2, SCF and TU.
  - Memory demand (~TB) → domain decomposition for Serpent2.

Parallelization in Monte Carlo transport

- Particle-based parallelism:
  - Particle histories are independent and can be calculated simultaneously.
  - Natural algorithm, embarrasingly parallel (not really).
  - Domain replication across MPI tasks, shared memory in OpenMP.
  - Typically very good speedup, but no memory scalability.
- Data decomposition:
  - Distribute material and cross-section data across MPI tasks.
  - Fetch remote data during tracking.
- Domain decomposition:
  - Geometry divided in domains somehow.
  - Particle histories are still independent, so particle-based parallelism is used.
  - Tracking algorithm more complex due to particles changing domains (if no approximations are introduced).
  - Domain decomposition across MPI tasks, shared memory in OpenMP.
  - Potentially poorer speedup, but memory scalability.
  - The decomposition of the geometry (CSG) can be quite challenging.
  - A few implementations done lately (OpenMC, RMC).

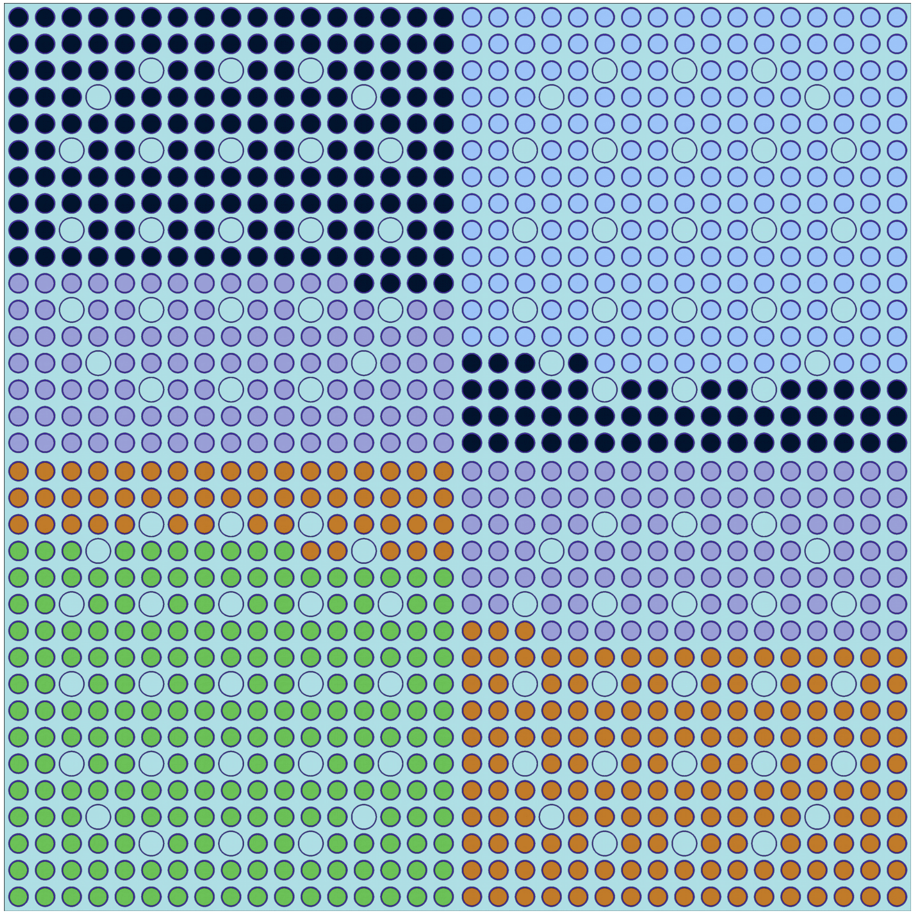
Collision-based Domain Decomposition

- Basic idea:
  - Divide burnable materials in domains (~ data decomposition).
  - Replicate non-burnable materials in all domains.
  - Each MPI task represents a domain.
  - No explicit partition of the geometry.
- Particle tracking:
  - Neutrons created in each domain in local fissile materials.
  - Particles tracked until absorbed or leaked, or until a collision in a non-local material occurs.
  - Particles buffered and sent across domains.
  - A transport cycle is completed when all particle histories in the whole system are finished.



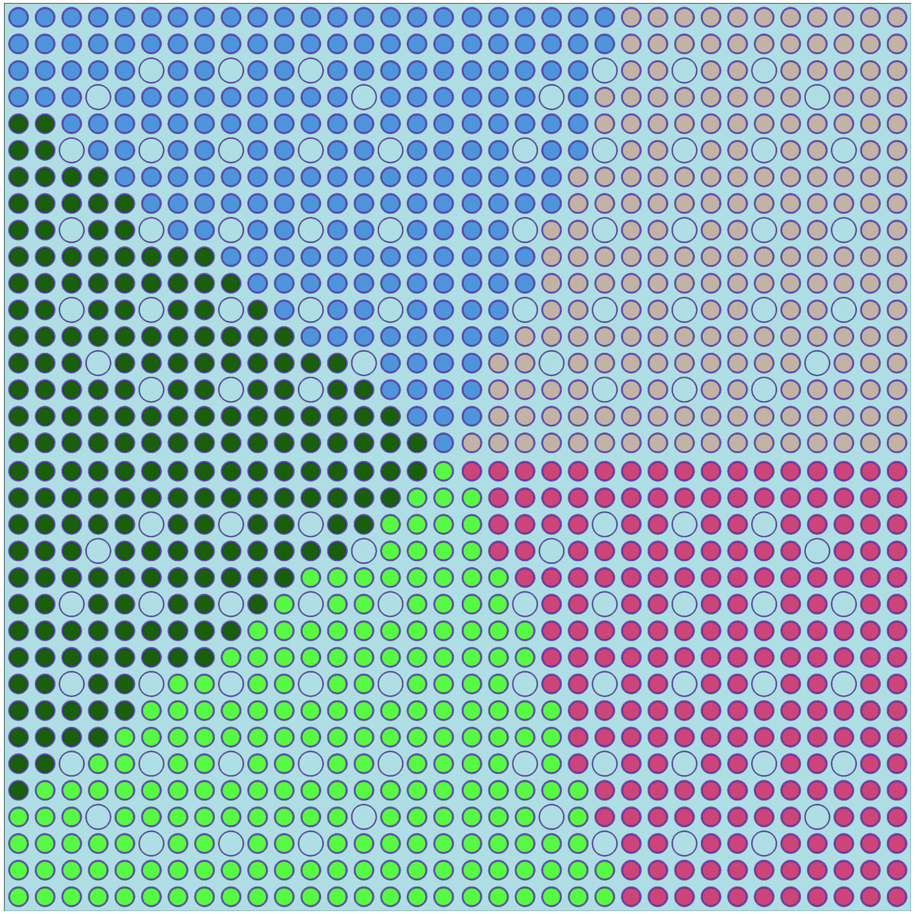
Material division

- Implemented options:
  - By material index (set dd 1).



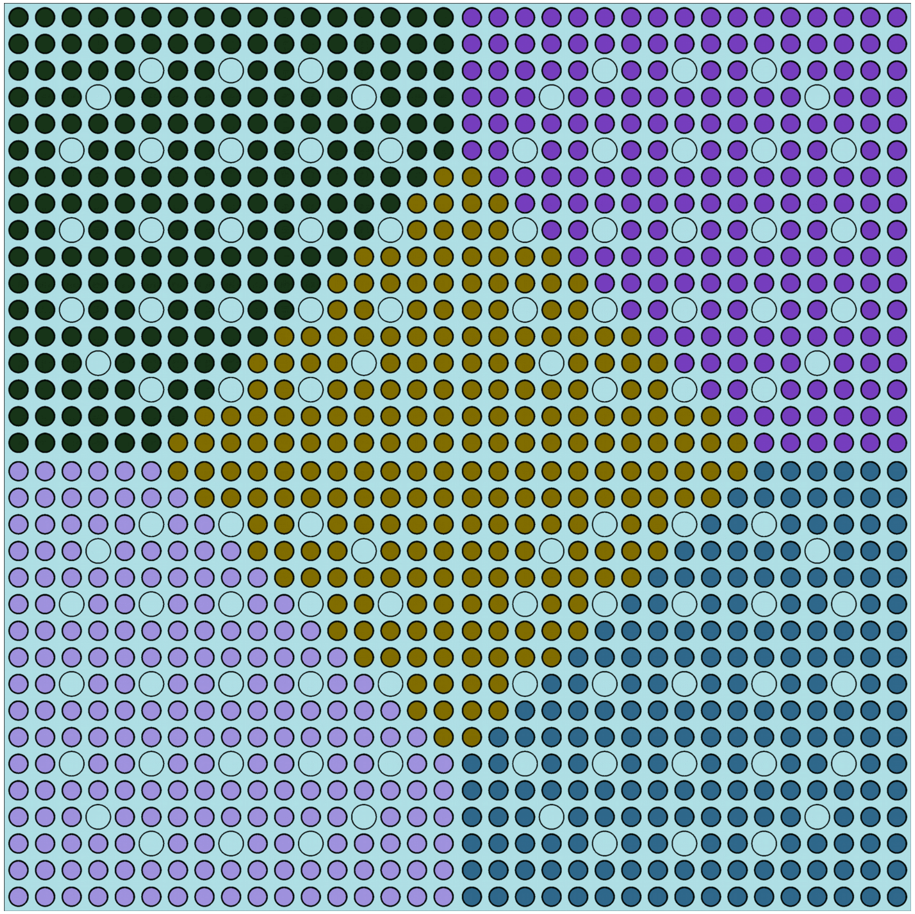
Material division

- Implemented options:
  - By material index (set dd 1).
  - By angular sector (set dd 2).



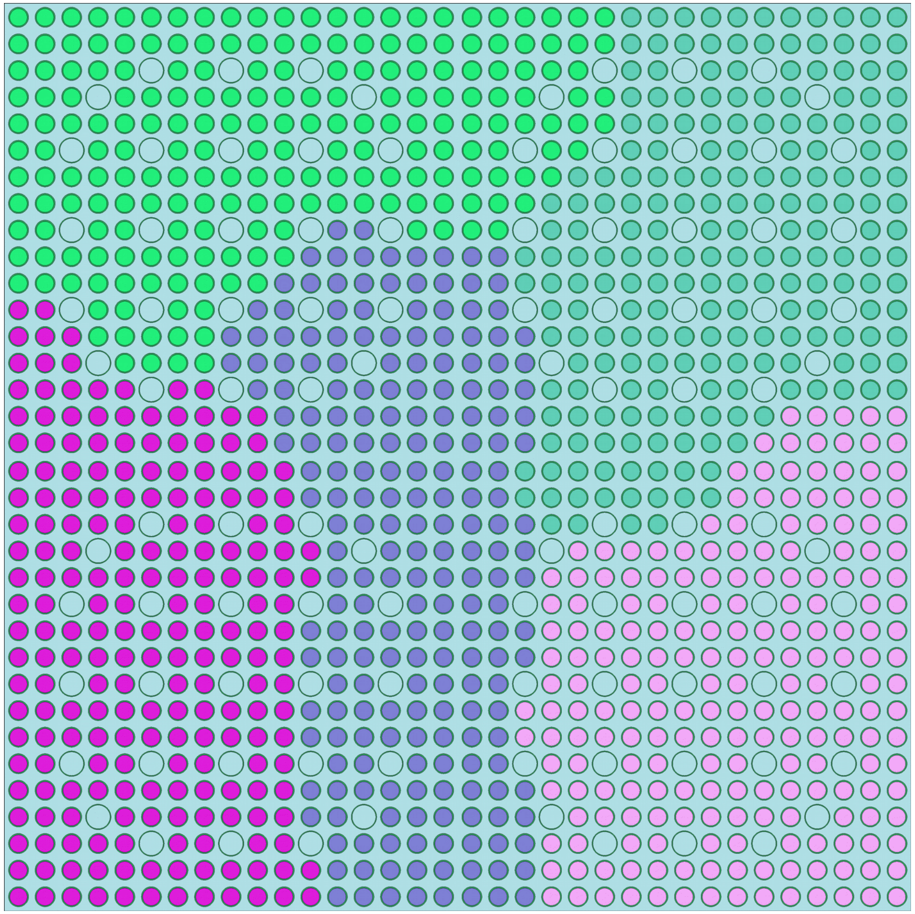
Material division

- Implemented options:
  - By material index (set dd 1).
  - By angular sector (set dd 2).
  - By angular sector with a central zone (set dd 3).



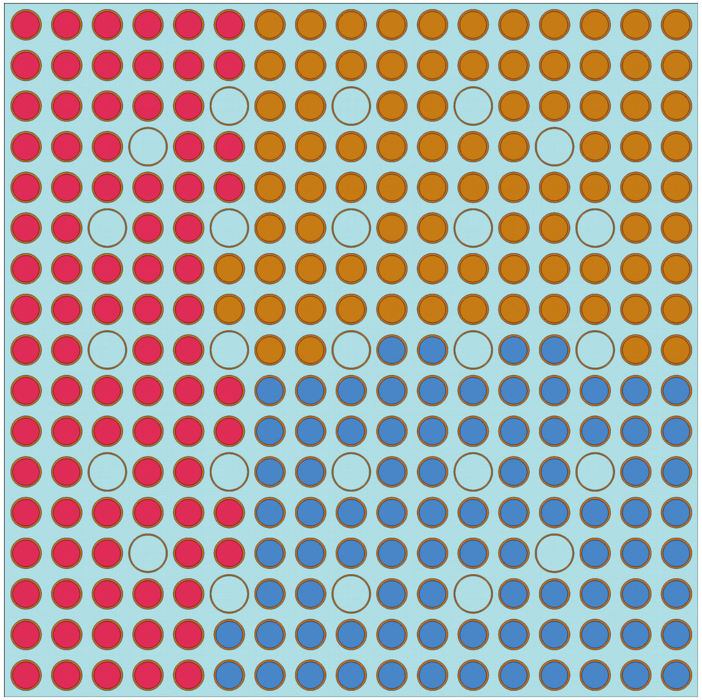
Material division

- Implemented options:
  - By material index (set dd 1).
  - By angular sector (set dd 2).
  - By angular sector with a central zone (set dd 3).
  - By graph partition (set dd 4).
- Graph-based division:
  - Material graph:
    - Vertices: materials.
    - Edges: material connections with cutoff maximum distance.
    - Weights: inverse of the distance.
  - Cartesian mesh to avoid comparing all materials ($O(n^2)$).
  - Partition done with Metis.



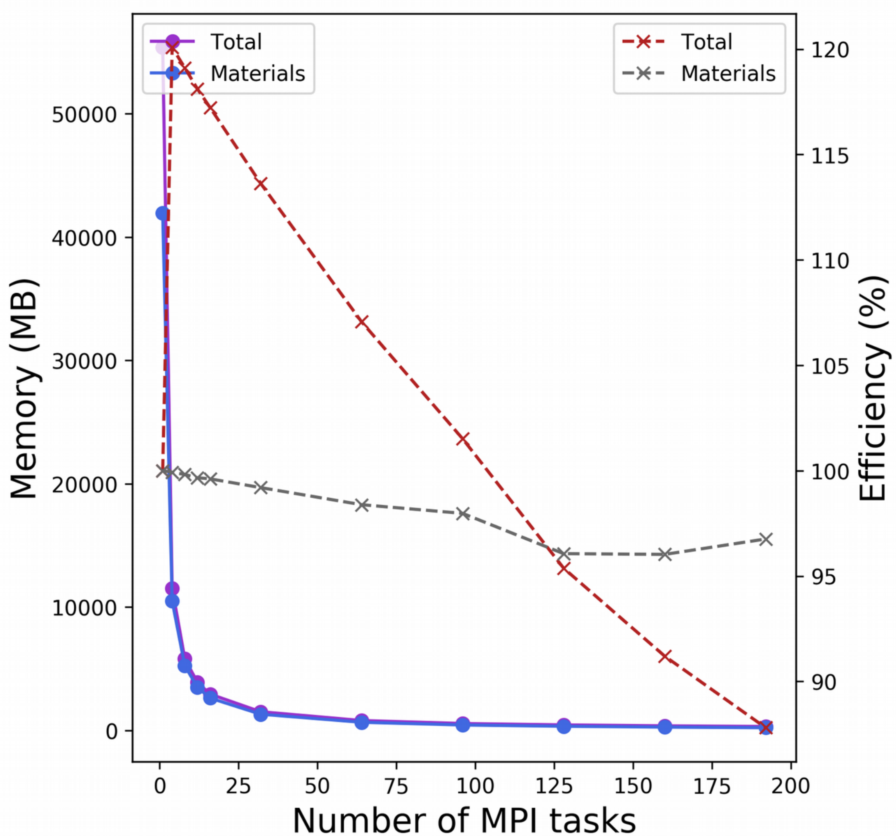
Tracking algorithm

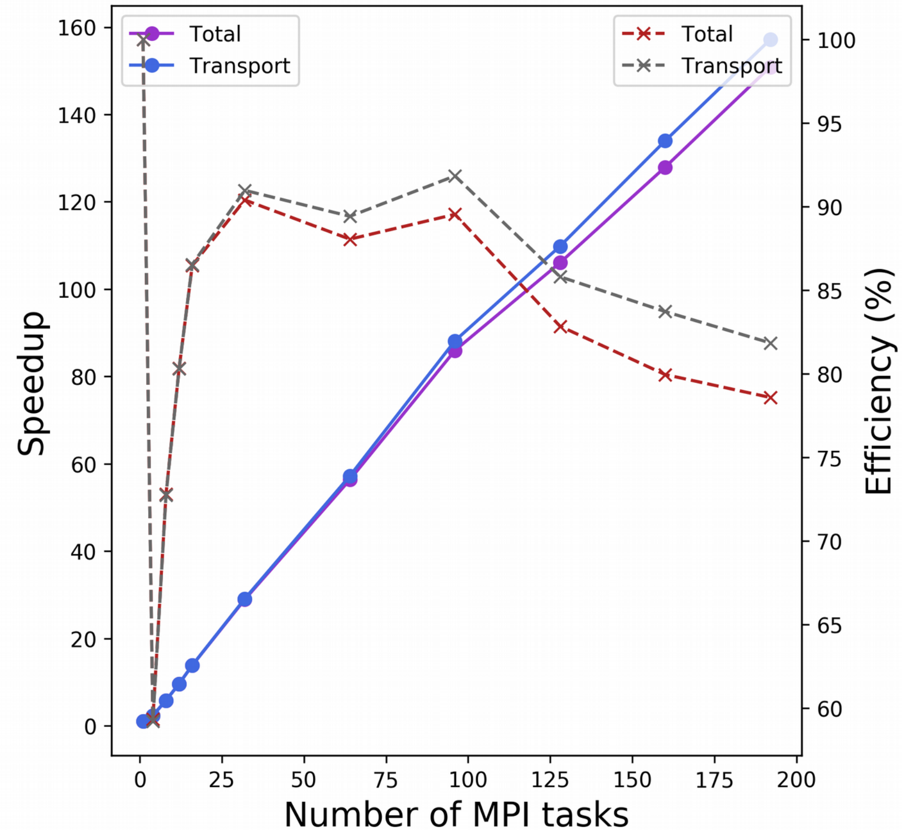
- Particle transfers:
  - Particles are sent across domains when collisions in non-local materials occur.
  - Buffers used to send particles in larger messages.
  - Asynchronous communications (MPI_Isend(), MPI_Irecv()).
  - Direct task-to-task messages.
- Termination control:
  - All histories in the whole system need to be completed.
  - Tricky due to the use of asynchronous communications.
  - Stopping criteria: all local particles tracked, all messages sent received.
  - Two step calculation of the particle balance:
    - Asynchronous reduction to get an estimation.
    - Synchronous reduction to make sure.
  - Binary tree to handle collective communications.



Performance

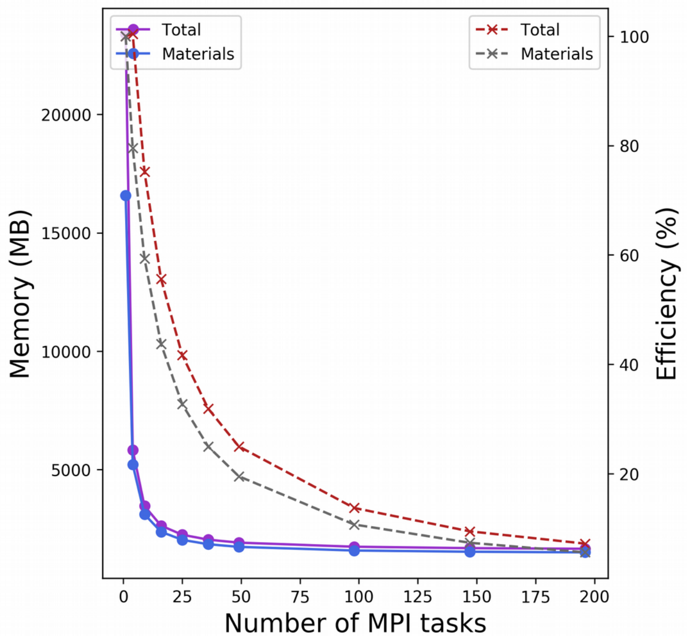
- Homogeneous system:

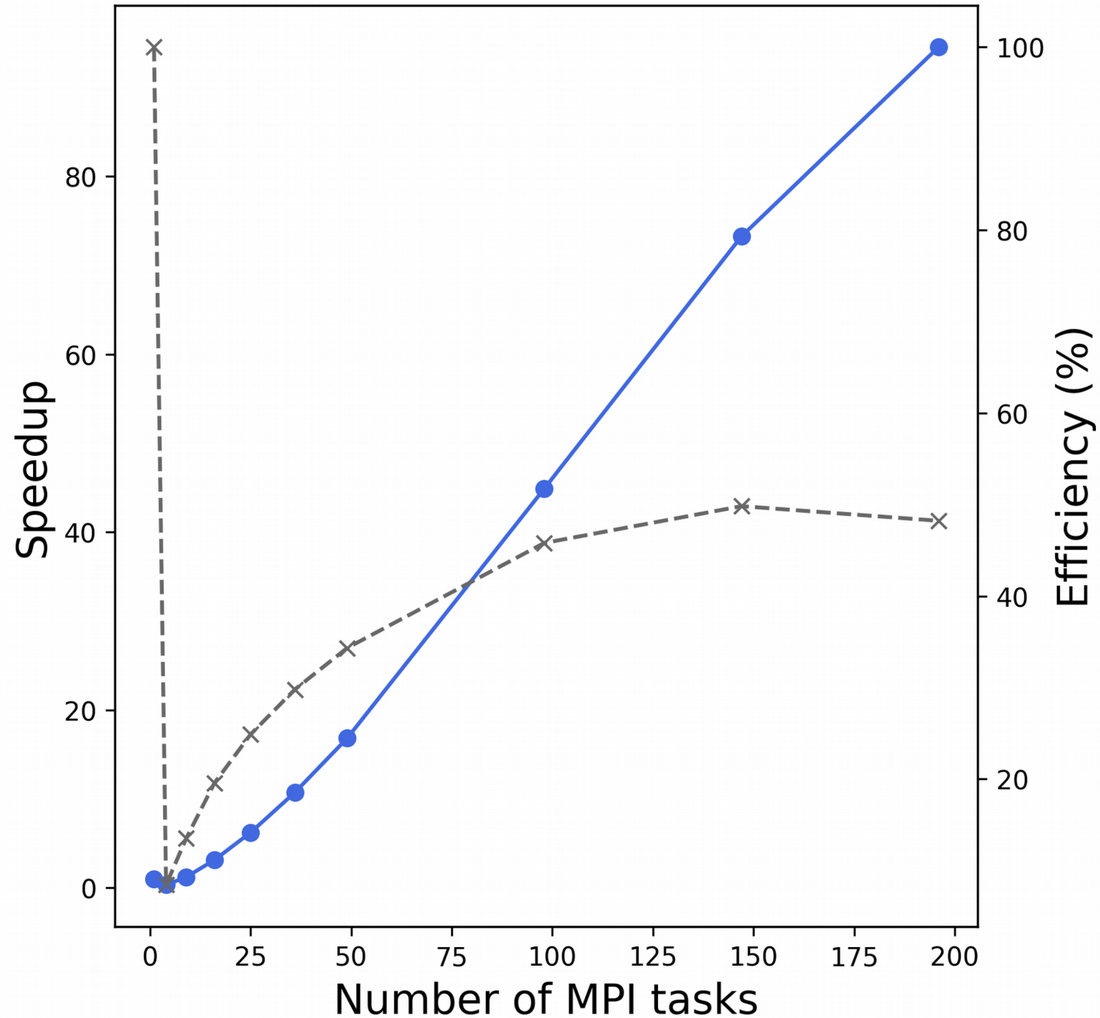




Performance

- 12x12 square PWR core:





Load balance

- Material decomposition:
  - Based on material positions, volumes could be used as well.
  - Neutron source not known.
- Dynamic load balance:
  - Materials could switch domains to compensate load imbalances.
  - Not easy to implement.
- Uniform Fission Source (UFS) method:
  - Source size (number of source neutrons) uniform over a superimposed mesh.
  - Intended to even out the statistical uncertainty.
  - Would force all domains to have the same number of particles.
  - Not working with domain decomposition at the moment.

Conclusions and further work

- Collision-based Domain Decomposition in Serpent2:
  - First version developed and included in the 2.1.31 version.
  - Most features working.
  - Optimizations might be possible (UFS, OpenMP).
  - Good memory scalability.
- Testing:
  - Weak scalability test for memory and speedup, probably more fair.
  - Optimization modes.
  - Performance for the target problems (PWR and VVER full-core depletion).
- McSAFE project:
  - Development phase almost over.
  - Validation of Serpent2-SCF-TU for full-core depletion.

Questions? Comments?