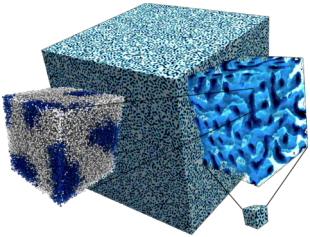
SOMA:
material science of soft, polymeric systems via HPC GPU computing

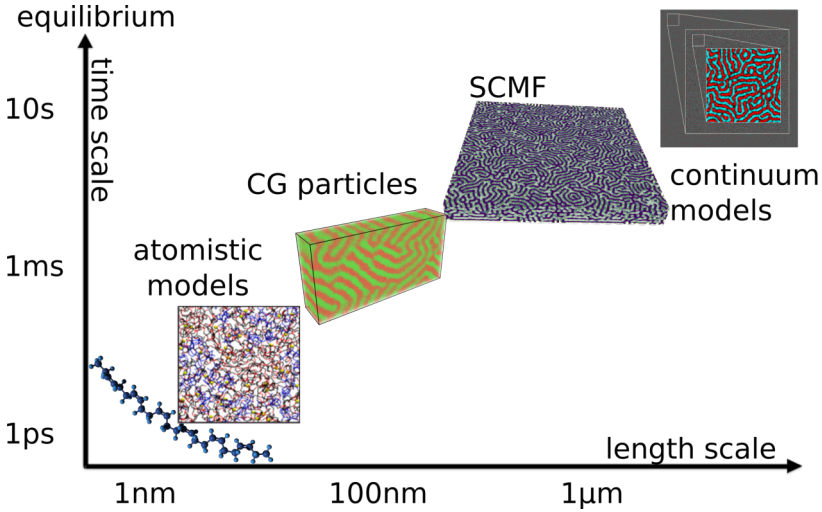
Ludwig Schneider & SOMA team

Pritzker School of Molecular Engineering, University of Chicago

June 21, 2022



Hierarchy of coarse-grained models

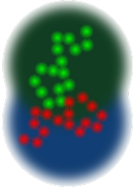


- omission of atomistic details
- large length scales
- long time scales
- DPD enables hydrodynamics
- SCMF unlocks engineering scales
- continuum models would allow full product simulation

Coarse-graining: single bead ↔ many atoms

Coarse-graining

- Gaussian chains
- fewer degrees of freedom
- universality
- higher parallelism
- soft interactions



density $\hat{\phi}_A, \hat{\phi}_B$ based Monte-Carlo

- harmonic bond potential: R_{e0}
  - $V_h(\boldsymbol{r}) = \frac{k_2}{2}\boldsymbol{r}^2$
- restrain density fluctuations: $\kappa_0 N$
  - $\mathscr{H}_{\text{fluc.}}[\hat{\phi}_A, \hat{\phi}_B] \propto \int \mathrm{d}\boldsymbol{r}\, \frac{\kappa_0 N}{2}\left(\hat{\phi}_A(\boldsymbol{r}) + \hat{\phi}_B(\boldsymbol{r}) - 1\right)^2$
- microphase separation: $\chi_0 N$
  - $\mathscr{H}_{\text{sep.}}[\hat{\phi}_A, \hat{\phi}_B] \propto \int \mathrm{d}\boldsymbol{r}\, \chi_0 N\, \hat{\phi}_A(\boldsymbol{r})\hat{\phi}_B(\boldsymbol{r})$

$$N = 2^{14}$$
$$\Downarrow$$
$$N = 2^{7}$$

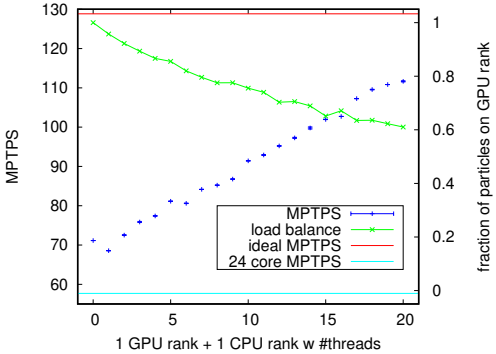
SOMA
yes, soft matters

https://gitlab.com/InnocentBug/SOMA
L. Schneider and M. Müller, Comput. Phys. Commun. **235C**, 463–476 (2019)

Why OpenACC?

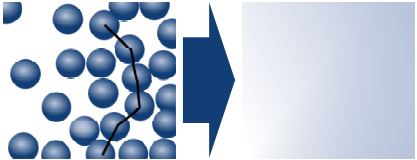
- OpenACC:
  - pragma based accelerator description
- open standard
- support for the implementation (GPU-Hackathon 2016)
- implemented in NVIDIA HPC SDK (`nvc`)
- CPU and GPU implementation:
  - single code base
  - CPU and GPU version work together
- high acceptance (backend) by scientists:
  - modifications only require single C
- unfortunate compiler support

OpenACC
More Science, Less Programming



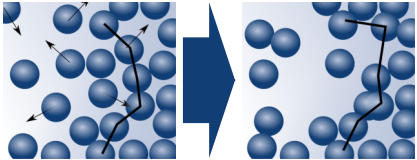
Single-Chain-in-Mean-Field algorithm

1. calculate density $\rho(t) \leftarrow \{\boldsymbol{r}_i(t)\}$



2. bond force-biased Monte-Carlo

$$\{\boldsymbol{r}_i(t)\} \overset{\rho(t)}{\rightarrow} \{\boldsymbol{r}_i(t+\Delta t)\}$$



3. repeat

Step 1

- simple reduction problem
- non-bonded: calculation on a grid

Step 2

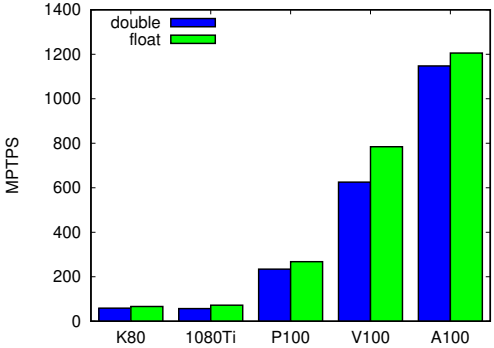
- bond force-biased Monte-Carlo
- non-bonded particles are independent

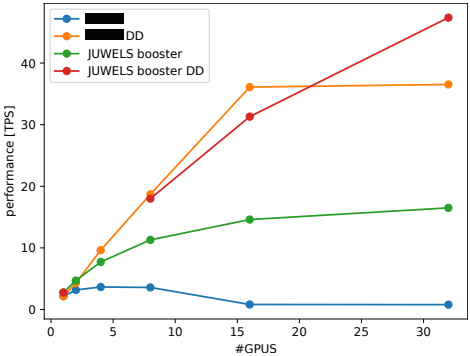
Step 3

- sync densities between GPUs via MPI

K. C. Daoulas and M. Müller, J. Chem. Phys. **125**, 184904 (2006)

SOMA performance



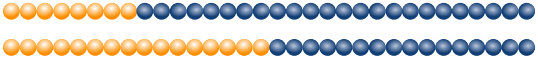


- recompile, bug fix, run
- increasing performance with new generations

- good performance
- software support of GPU-MPI challenging

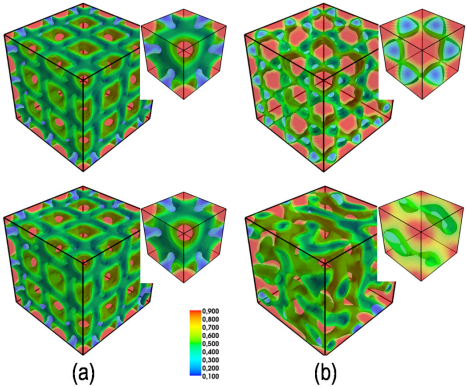
First success story 2017

- diblock copolymers form micro-phase separated morphologies
- access to new meta-stable morphologies
- alchemical transformation: change A $\rightarrow$ B



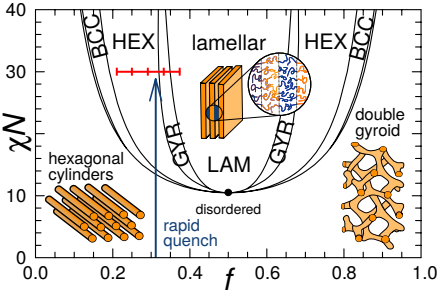
- prove stability with SOMA
- $\approx$ 100 million particles & high density

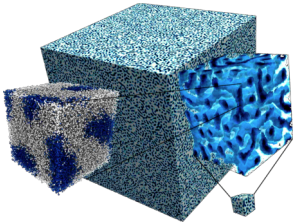
D.-W. Sun and M. Müller, Physical Review Letters **118**, 067801 (2017)

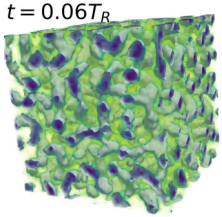


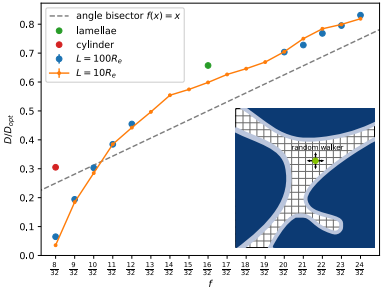
- a: I-WP formed from unstable BCC
- b: HEX2 $f = 9/32 \rightarrow f = 22/32$

Meta-stable non-periodic network phases







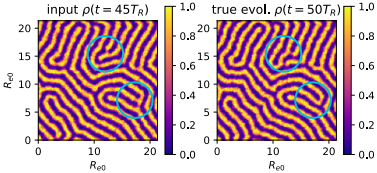


- application for battery materials
- mechanical stability & ion flux
- stable phases: limited option
- non-periodic: large-system size $nN \approx 4 \cdot 10^9$
- diffusive properties differ from equilibrium

Machine Learning Morphology Predictions from Particle-Based Simulations

SOMA based simulations

- 2048 short trajectories



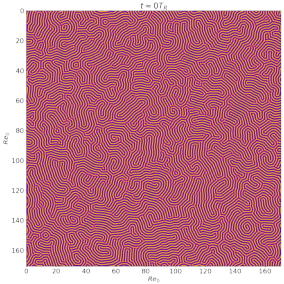
Learn defect kinetics

- $\Delta T > T_R$
- over-damped
- time independent

Train neural-net to perform transition

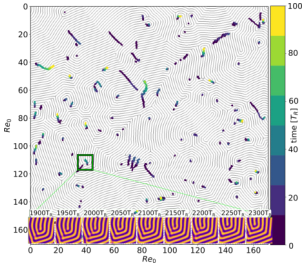
Artificial neural network

- size-free network
  $\rightarrow$ small training, large predictions
- stationary Markov
  $\rightarrow$ repeated application for long trajectory



Insights into defect kinetics

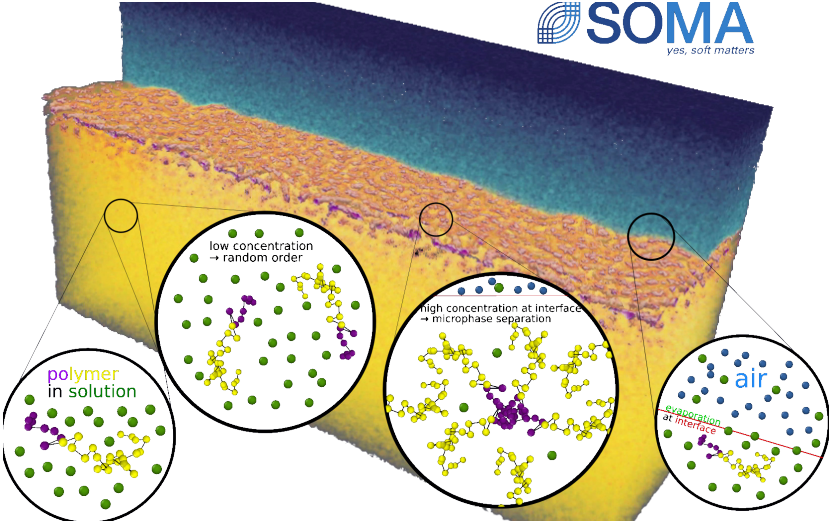
- fast slide defects



Computational efficiency

- training data: 3144 GPU-h
- 1 SOMA data: 40 GPU-min
- break-even: 5000 frames

Evaporation Drives Microphase Separation: Membrane Fabrication

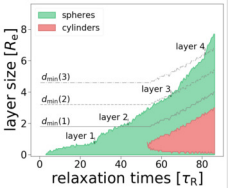


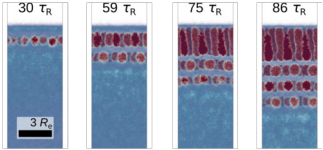
- cylinder orientation
- evaporation details
- non-equilibrium
- visualization: Jülich
- simulation: JUWELS booster

Evaporation speed and affinity control orientation

Slow evaporation & attraction

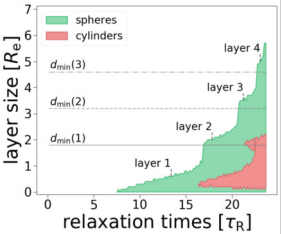
- initial sphere formation
- elongation of spheres
- standing cylinders & spheres

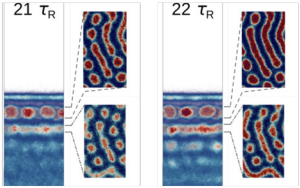




Fast evaporation & no attraction

- initially spheres
- horizontal connection
- layers of laying cylinders



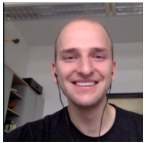
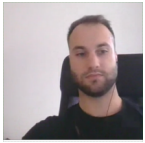


Summary

- hackathons & early access programs
- single and multi GPU performance
- applications:
  - polymer research
  - battery materials
  - membrane fabrication
- pushing computational boundaries
- future: platform independence

SOMA
yes, soft matters

Contributors:
Oliver Dreyer, Gregor Ibbeken, Niklas Blagojevic, Andreas Herten


Supervisors:
Marcus Müller, Göttingen; Juan de Pablo, Chicago


References I

[1] L. Schneider and M. Müller, Comput. Phys. Commun. **235C**, 463–476 (2019).

[2] K. C. Daoulas and M. Müller, J. Chem. Phys. **125**, 184904 (2006).

[3] D.-W. Sun and M. Müller, Physical Review Letters **118**, 067801 (2017).

[4] M. W. Matsen, J. Phys: Condens. Matter **14**, R21 (2001).

[5] L. Schneider and M. Müller, Macromolecules **52**, 2050–2062 (2019).

[6] L. Schneider and J. J. de Pablo, Macromolecules **54**, 10074–10085 (2021).

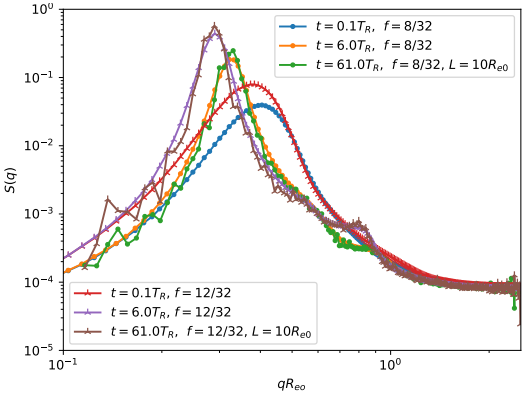
[7] D. Ben-Avraham and S. Havlin, Cambridge university press, (2000).

[8] H. J. Heijmans, SIAM review **37**, 1–36 (1995).

[9] M. E. O'NEILL, ACM Transactions on Mathematical Software (2014).

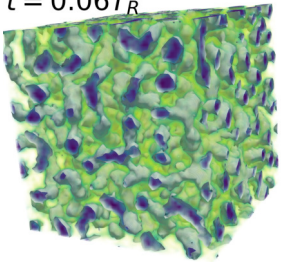
[10] J. K. Salmon, M. A. Moraes, et al., in Proceedings of 2011 international conference for high performance computing, networking, storage and analysis (2011), pp. 1–12.

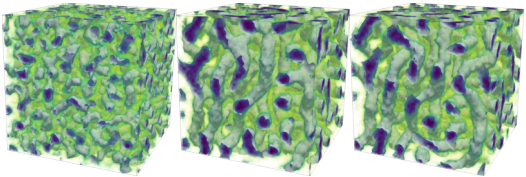
Metastable network phases



structure factor $S(|\boldsymbol{q}|)$

- rapid quench
- metastable
- $t = 6T_R$
- 3D diffusion?

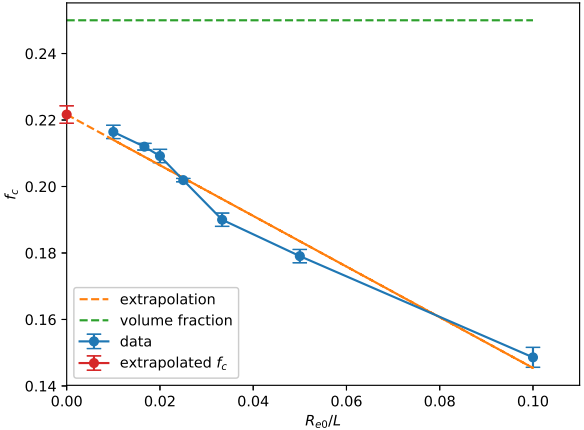


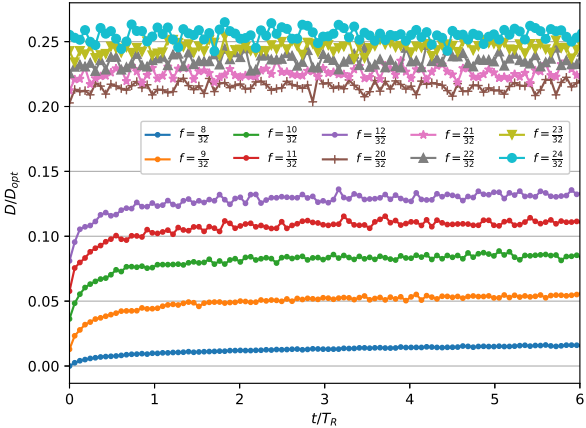


$t = 0.1T_R$ $t = 6T_R$ $t = 61T_R$

L. Schneider and M. Müller, Macromolecules 52, 2050–2062 (2019)

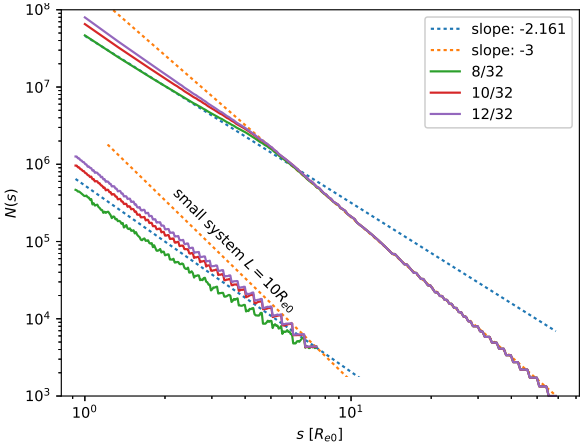
Finite-size effect for cluster fraction f_c





- systematic finite-size effect: $f_c = f_c^* - \alpha/L$
- convergences to plateau after coarsening

Space-filling characteristics of 3D network structures



- not space-filling $s < s^* \approx (5.4 - 6.4)R_{e0}$
- space-filling on large length scales
- characteristics of an overcritical cluster
- simulations of large sizes required

Box counting algorithm

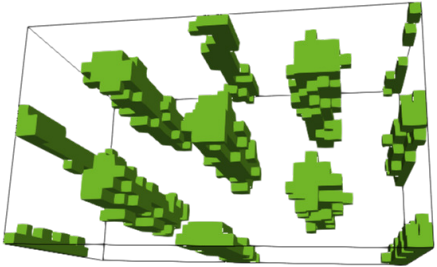
1. divide system if box of length s
2. count boxes that contain structure $N(s)$

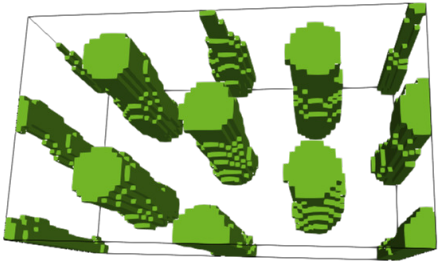
- $N(s) \propto s^{-d_f}$ fractal dimension

D. Ben-Avraham and S. Havlin, Cambridge university press, (2000)

L. Schneider and M. Müller, Macromolecules **52**, 2050–2062 (2019)

Grid smoothing for diffusion analysis





1. time average to reduce fluctuations
2. increase grid resolution
3. apply opening $\bigcirc$ and closing $\bullet$
   - $\rho \bigcirc s = (\rho \ominus s) \oplus s$
   - $\rho \bullet s = (\rho \oplus s) \ominus s$

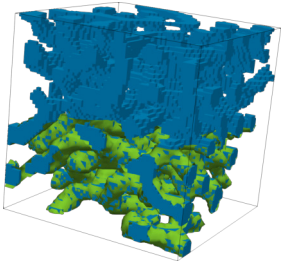
$$(\rho \ominus s)(\boldsymbol{r}) = \inf_{\boldsymbol{r}' \in s} [\rho(\boldsymbol{r} + \boldsymbol{r}') - s(\boldsymbol{r}')]$$

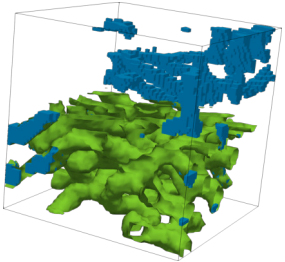
$$(\rho \oplus s)(\boldsymbol{r}) = \sup_{\boldsymbol{r}' \in s} [\rho(\boldsymbol{r}) - s(\boldsymbol{r} - \boldsymbol{r}')]$$

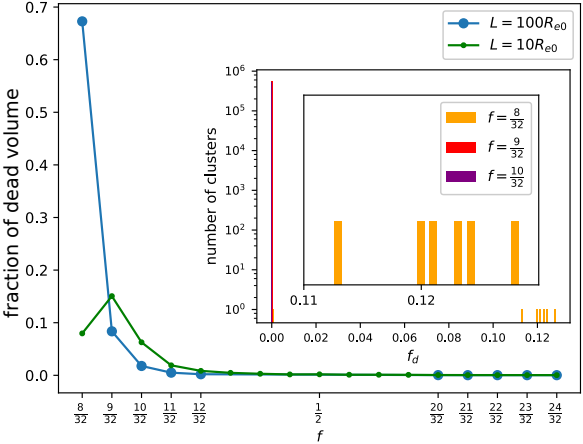
H. J. Heijmans, SIAM review **37**, 1–36 (1995)

Dead-end analysis

- diffusion $\neq$ directed transport
- dead-ends in network structures
- impact on small volume fractions f
- underestimated by small simulations

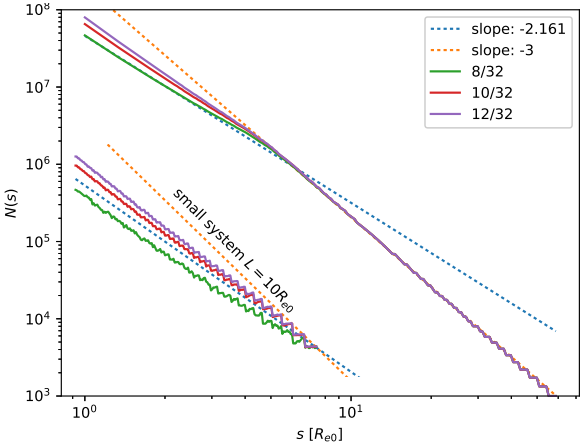






L. Schneider and M. Müller, Macromolecules **52**, 2050–2062 (2019)

Space-filling characteristics of 3D network structures



- not space-filling $s < s^* \approx (5.4 - 6.4)R_{e0}$
- space-filling on large length scales
- characteristics of an overcritical cluster
- simulations of large sizes required

Box counting algorithm

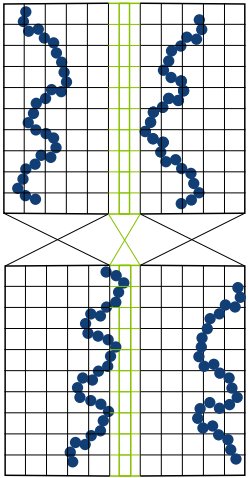
1. divide system if box of length s
2. count boxes that contain structure $N(s)$

- $N(s) \propto s^{-d_f}$ fractal dimension

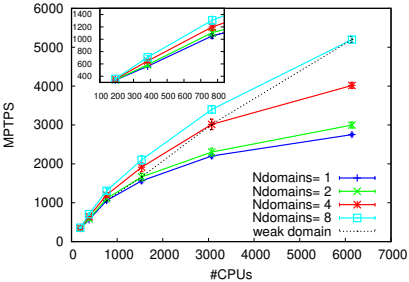
D. Ben-Avraham and S. Havlin, Cambridge university press, (2000)

L. Schneider and M. Müller, Macromolecules **52**, 2050–2062 (2019)

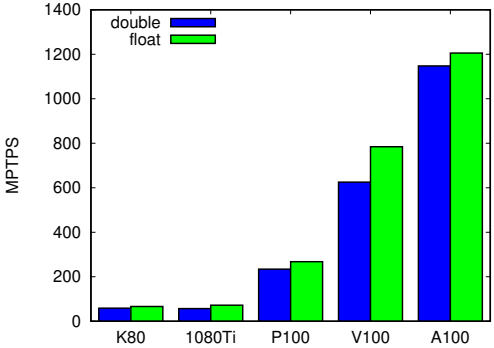
What is necessary for engineering scales: spatial domain decomposition!

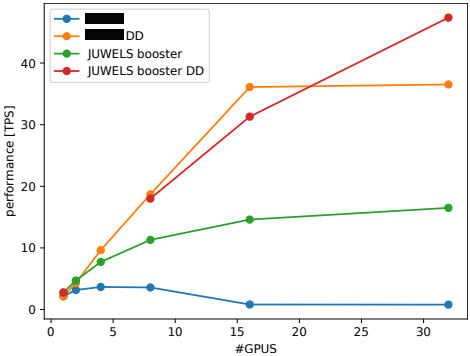


- less memory per `MPI` rank
- all-to-all vs next-neighbor communication
- engineering scale $nN = 4 \cdot 10^9$
- communication bound problem
- linear scaling if constant #CPUs per domain



SOMA performance



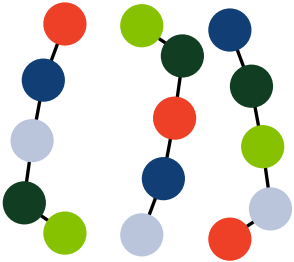


- recompile, bug fix, run
- increasing performance with new generations

- good performance
- software support of GPU-MPI challenging

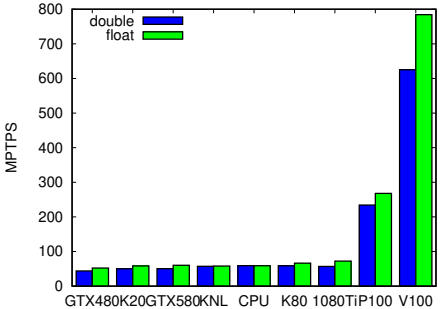
GPU parallel level: independent molecules

Polymer iteration:



- same color: ⇒ parallel
- iterate colors
- 3 threads
- 5 iterations

```
#pragma acc parallel loop
#pragma omp parallel for
for(uint64_t mol=0; mol < Nmol; mol++){
#pragma acc loop seq
  for(int mono=0;mono<getN(mol);mono++){
    const unsigned int i = rng(N);
    smart_MC(i); }}
```

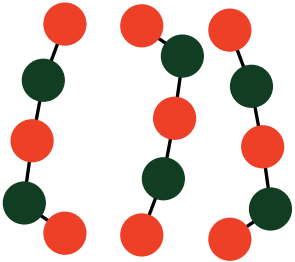


+ simple independent units
+ parallelism scales with system
− large polymers (networks)
− no dynamic bonds

L. Schneider and M. Müller, Comput. Phys. Commun. **235C**, 463–476 (2019)

GPU parallel level: independent sets of not-bonded beads

Independent set iteration:



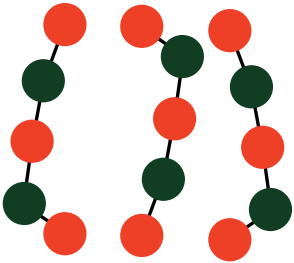
- higher parallelism
- 9 (6) threads
- 2 iterations

+ full utilization of parallelism
+ networks or many nodes
− non-trivial set decomposition
− additional memory and complexity

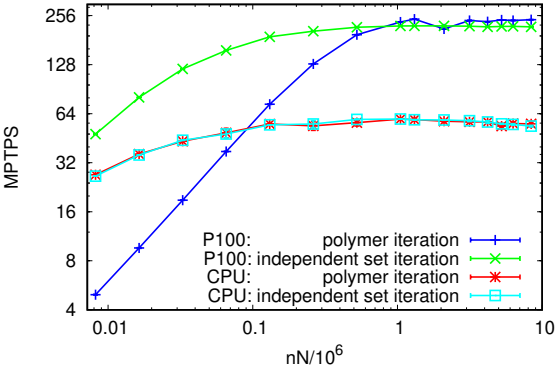
```
#pragma acc parallel loop
#pragma omp parallel for
for (uint64_t mol=0; mol < Nmol; mol++){
//Generate rdm permutation of the sets
//and store result in set_permutation[]

#pragma acc loop seq
  for(int i_set=0; i_set < Nsets;
      i_set++){
    const int set_id =
        set_permutation[i_set];
#pragma acc loop vector
    for(int i_p=0; i_p < setL[set_id];
        i_p++){
      unsigned int ip = sets[...];
      smart_MC(ip);}}
```

GPU parallel level: independent sets of not-bonded beads

Independent set iteration:



- higher parallelism
- 9 (6) threads
- 2 iterations



+ full utilization of parallelism
+ networks or many nodes
− non-trivial set decomposition
− additional memory and complexity

L. Schneider and M. Müller, Comput. Phys. Commun. **235C**, 463–476 (2019)

Pseudo random number definition

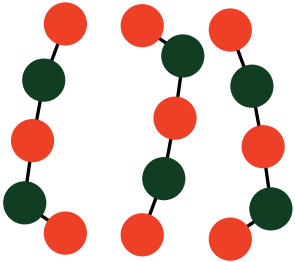
Many random numbers in parallel

1. streamable pRNGs
   - 1 seed, multiple streams
   - (small) state of pRNG
   - example: PCG **O'NEILL (2014)**
2. hash based pRNGs
   - hash time step, thread, seed and ID
   - requires good hash functions
   - example: random123
     Salmon, Moraes, et al. (2011)
3. precompute random numbers
   - memory intensive
   - number of RNGs must be known
   - example: `curand`

- SOMA uses PCG32
- 128 bit state (SOMA is memory bound)
- number of RNGs per step unknown
- easy C implementation
- multiple streams
- one RNG state per parallel thread
- SOMA offers other RNG for verification
  - Mersenne Twister (2504 bytes state)
- SOMA is offers bit-wise reproducible trajectories
  - parallel reductions only on integer numbers

Optimized architecture storage

- SOMA is memory bound
  - loading RNG state
  - position access (optimized)
  - density field access (“random”)
- reduction of memory access
- utilize different types of GPU memory



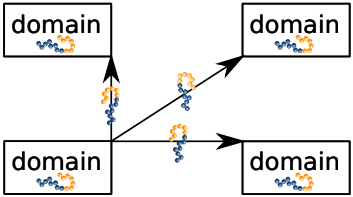
- typical scenario:
  - many molecules (polymers)
  - same bond architecture
- introduce types of molecules
- store bond architecture only once per type
- load architecture to shared GPU memory
- OpenACC: automatically

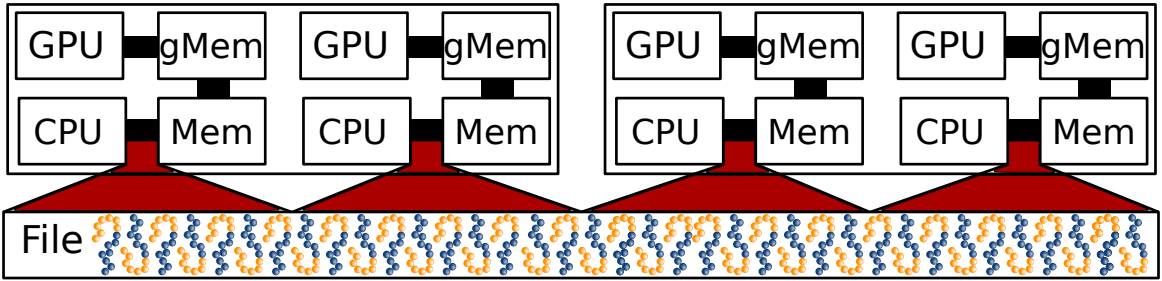
trade off

- complex storage
- reduced flexibility
- large molecule become complicated (networks)

Large systems require parallel IO

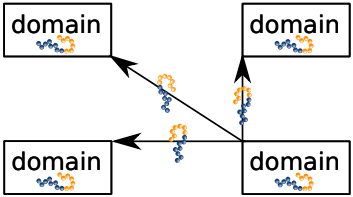
- HDF5 hides MPI-IO which hides the hardware
- HDF5 is widespread and fits in many pipelines
- enables handling files > memory
- domain decomposition requires 2nd communication

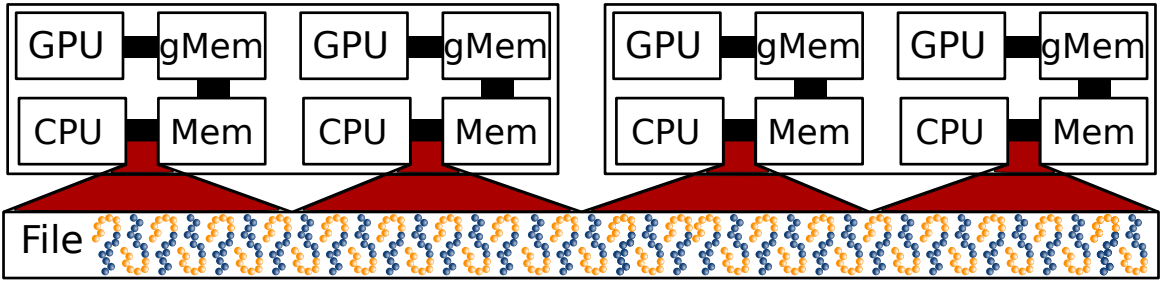




Large systems require parallel IO

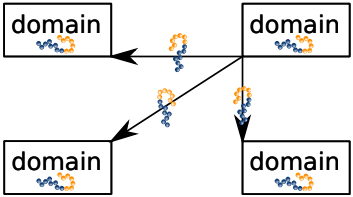
- HDF5 hides MPI-IO which hides the hardware
- HDF5 is widespread and fits in many pipelines
- enables handling files > memory
- domain decomposition requires 2nd communication

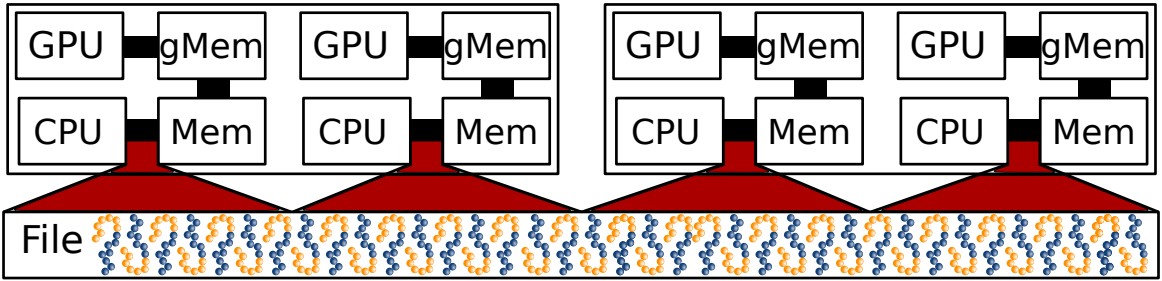




Large systems require parallel IO

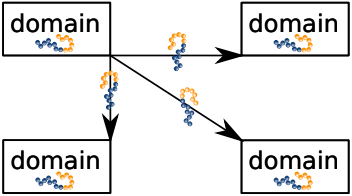
- HDF5 hides MPI-IO which hides the hardware
- HDF5 is widespread and fits in many pipelines
- enables handling files > memory
- domain decomposition requires 2nd communication

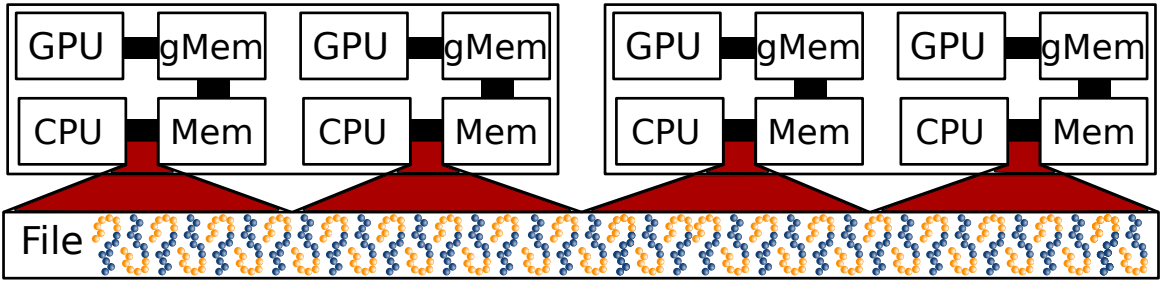




Large systems require parallel IO

- HDF5 hides MPI-IO which hides the hardware
- HDF5 is widespread and fits in many pipelines
- enables handling files > memory
- domain decomposition requires 2nd communication





Networks

- SOMA can simulate networks
- set decomposition parallelism
- there is no XML/python front end
- understanding/implementing network architecture can be tricky
- only single GPU support for networks

Gaoyuan

- set decomposition
- initialization of large networks
- cross-link algorithm

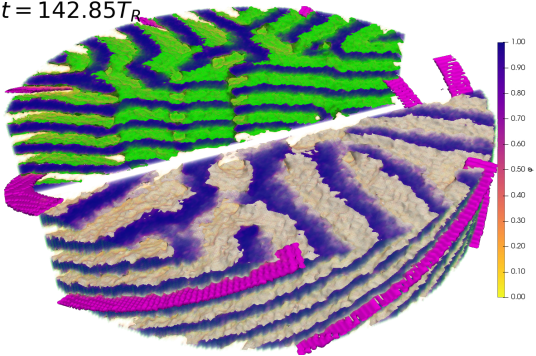
Leon

- regular diblock networks
- stress calculation in networks
- instantaneous strain

Confinements

- confinements/non-periodic boundary condition
  - inaccessible grid-cells
  - “area51”
- easy are: boxes, cylinder, spheres
- all other morphologies accessible via python/C
- surface and volume interactions
- external fields
- component dependent energy contributions
- same syntax as “area51”

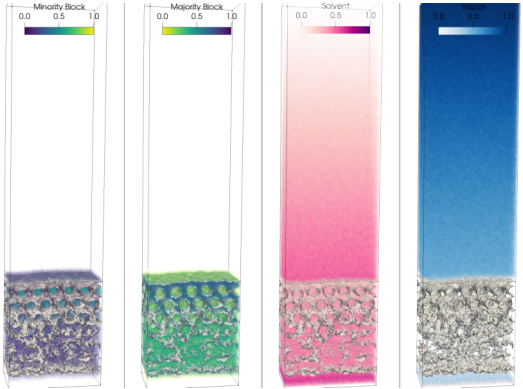
$t = 142.85 T_R$



- Georg Lichtenberg

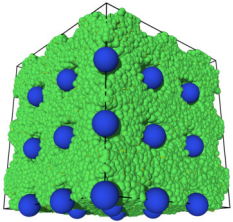
Evaporation

- solvent: short polymer chains $\chi N \approx 0$
- gas: short polymer chains $\chi N \gg 1$
- conversion zone: solvent $\rightarrow$ gas
- morphology dynamics?
- tune evaporation speed
- same syntax as “area51”



JUWELS booter

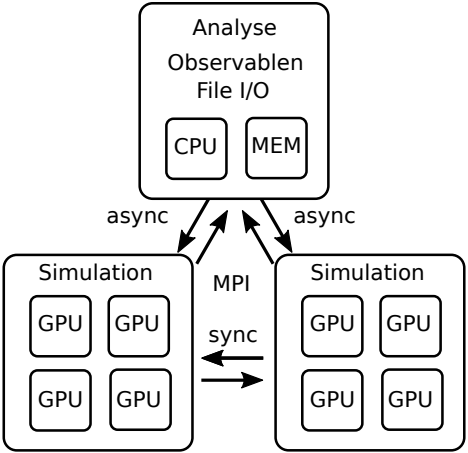
- many new nodes
- “weak” host CPU
- next-gen GPUs
- optimization of communication between GPUs
- Tunathon in May



- early access to JUWELS
- large-scale application required
- evaporation in large-scale systems
  - rework memory management
  - explicit gas particles
  - implicit gas phase
- rework configuration file I/O
- include nano-particles
  - interactions with polymers
  - NP-NP interaction

Dedicated analysis server

- large check-point files ($\approx$ 100GB) unsuitable for analysis
- solution: on-the-fly analysis
- problem: simulation resources idle during analysis
- idea: dedicated analysis server
- simulation snapshot sent via asynchronous MPI
- analysis and simulation parallel on different resources
- easy extension of the analysis server
  - GPU parallel not necessary



Density field: atomic reductions

- assign particles on a grid: atomic reductions

```
#pragma acc parallel loop gang num_gangs(n_polymers)
#pragma omp parallel for
for (uint64_t i = 0; i < n_polymers; i++){
#pragma acc loop vector
  for (unsigned int j = 0; j < N; j++){
    unsigned int monotype = get_particle_type(...);
    unsigned int idx = coord_to_index_unified(...);
#pragma acc atomic update
#pragma omp atomic
    p->fields_32[idx] += 1;}}
```

- 16-bit MPI communication between GPUs

```
#pragma acc update self(p->fields_unified[:])
  MPI_Allreduce( ... , MPI_UINT16_T, ... );
#pragma acc update device(p->fields_unified[:])
```