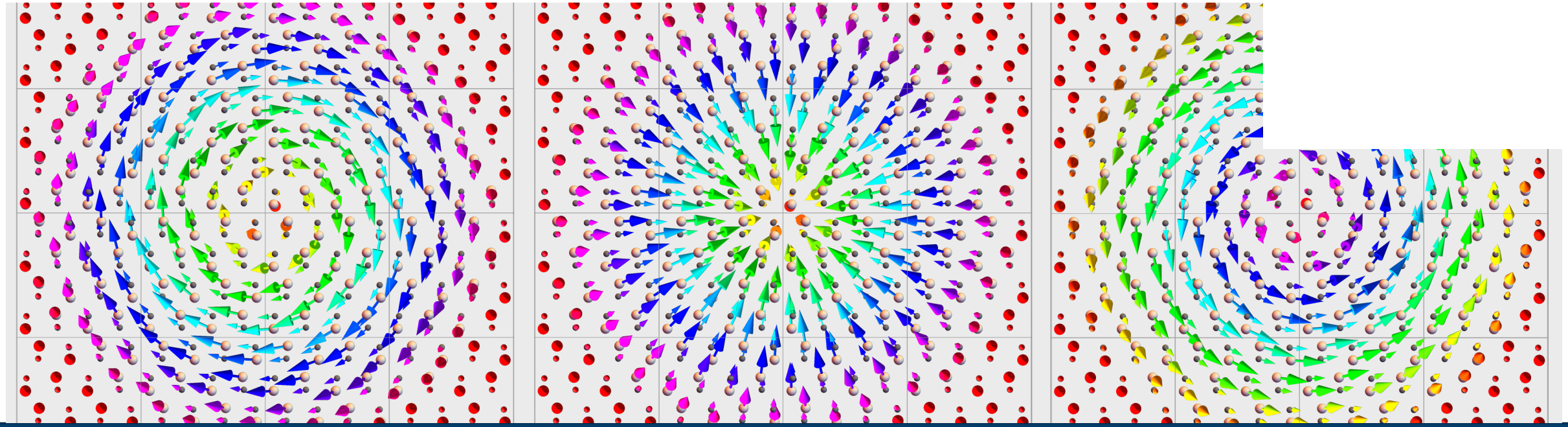


PARALLEL FLEUR

14.04.2021 I ULIANA ALEKSEEVA

MAX DRIVING THE EXASCALE TRANSITION

Mitglied der Helmholtz-Gemeinschaft

cecam Centre Européen de Calcul Atomique et Moléculaire

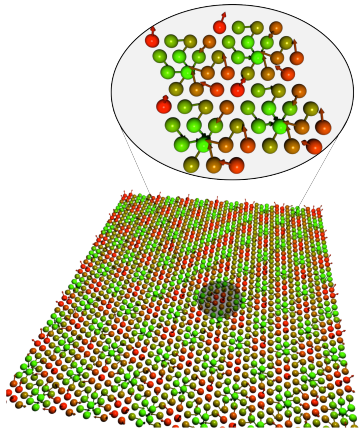
www.flapw.de

fleur

JÜLICH Forschungszentrum

WHY BOTHER?

- Execution time ~ (Number of atoms)^3
- Memory usage ~ (Number of atoms)^2



Calculating on a single core

# atoms	execution time	memory
64	1 hour	2.6 GB
256	2 days	72 GB
512	2 weeks?	556 GB
1000?	1 year?	
4000?	64 years ?	

Laptop:

cores : 4

memory : 16 GB

A BIT MORE CORES

Workstation:
cores : 48
memory : 128 GB

SuperMUC-NG:
(#15 in the world)
cores : 311040
memory : 719000 GB

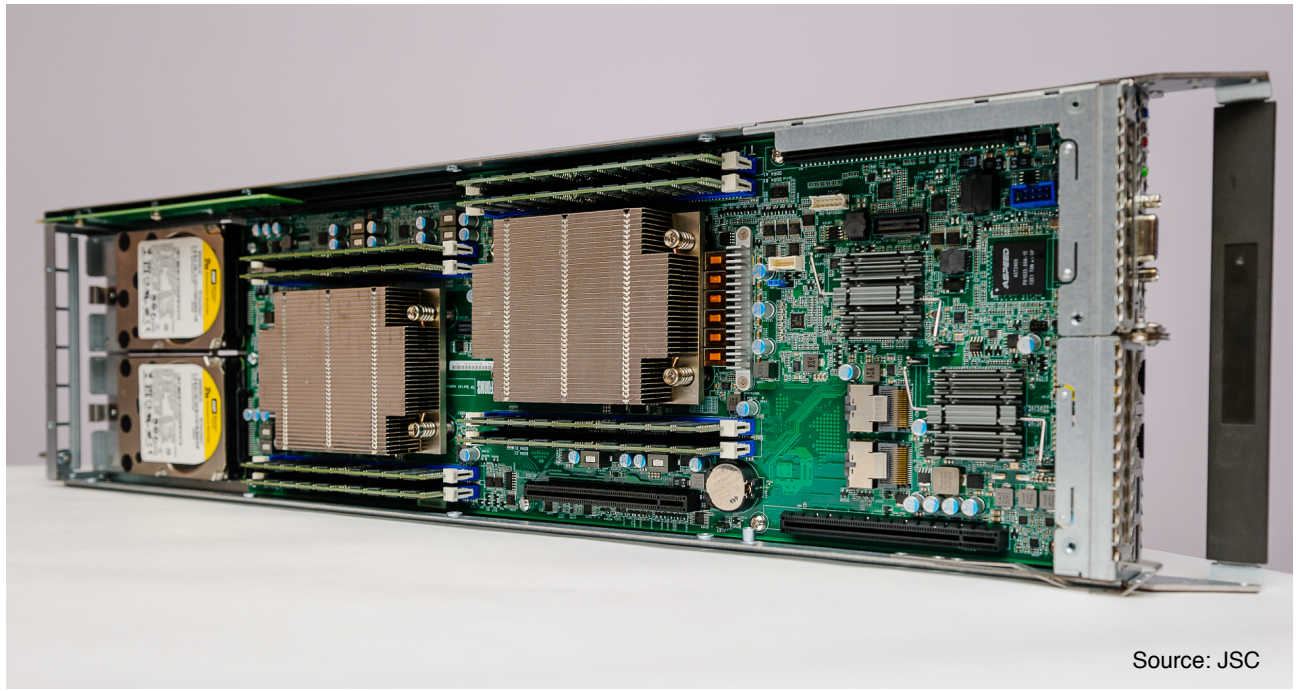
OUTLINE

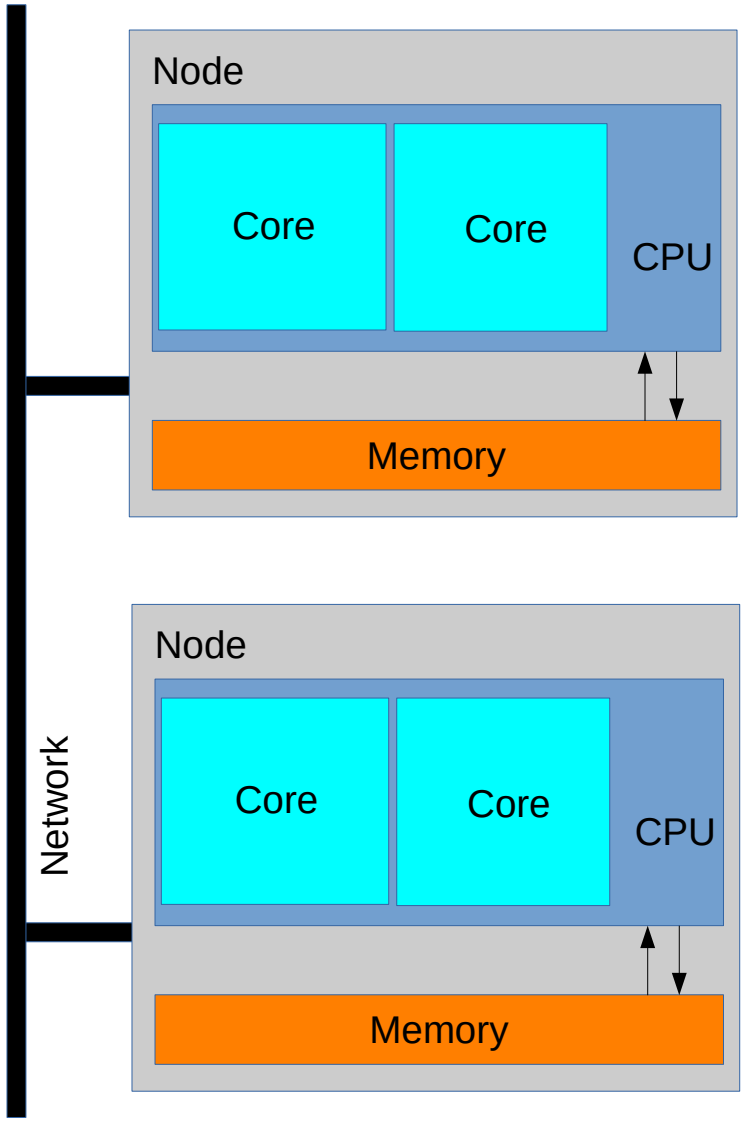
- Parallel programming
- Parallelization of FLEUR
- Examples

SUPERCOMPUTER

a simplified view

- cluster: many nodes
- node: compute cores + memory



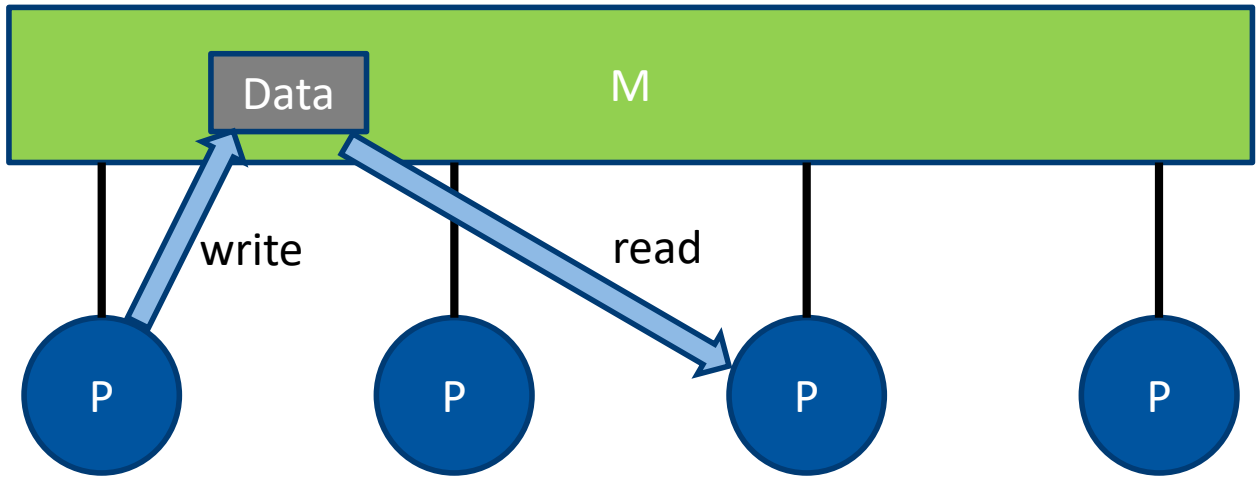


PARALLEL PROGRAMMING

shared memory

- memory is accessible for all processors
- no explicit communication
- what we use in FLEUR: OpenMP

```
!---> loop over atom types
!$OMP PARALLEL DO &
!$OMP& DEFAULT(none)&
!$OMP& PRIVATE(n,nn,natom,k,i,work_r,work_c,ccchi,kspin,fk,s,r1,fj,dfj,l,df,wronk,tmk,phase,&
!$OMP& inap,nap,j,fkr,fkp,ylm,ll1,m,c_0,c_1,c_2,jatom,lmp,inv_f,lm)&
!$OMP& SHARED(noco,atoms,sym,cell,oneD,lapw,nvmax,ne,zMat,usdus,ci,iintsp,&
!$OMP& jspin,bkpt,qss1,qss2,qss3,&
!$OMP& apw,const,nobd,&
!$OMP& alo1,blo1,clo1,kvec,nbasf0,nkvec,enough,&
!$OMP& acof,bcof,ccof)
DO n = 1,atoms%ntype
   !  ----> loop over equivalent atoms
   DO nn = 1,atoms%neq(n)
      natom = 0
      DO i = 1, n-1
         natom = natom + atoms%neq(i)
      ENDDO
      natom = natom + nn
      IF ((atoms%invsat(natom).EQ.0) .OR. (atoms%invsat(natom).EQ.1)) THEN
         !--->        loop over lapws
         IF (zmat%l_real) THEN
            ALLOCATE ( work_r(nobd) )
         ELSE
            ALLOCATE ( work_c(nobd) )
         ENDIF
         DO k = 1,nvmax
            IF (.NOT.noco%l_noco) THEN
               IF (zmat%l_real) THEN
                  work_r(:ne)=zMat%z_r(k,:ne)
               ELSE
```



Source: H.Iliev

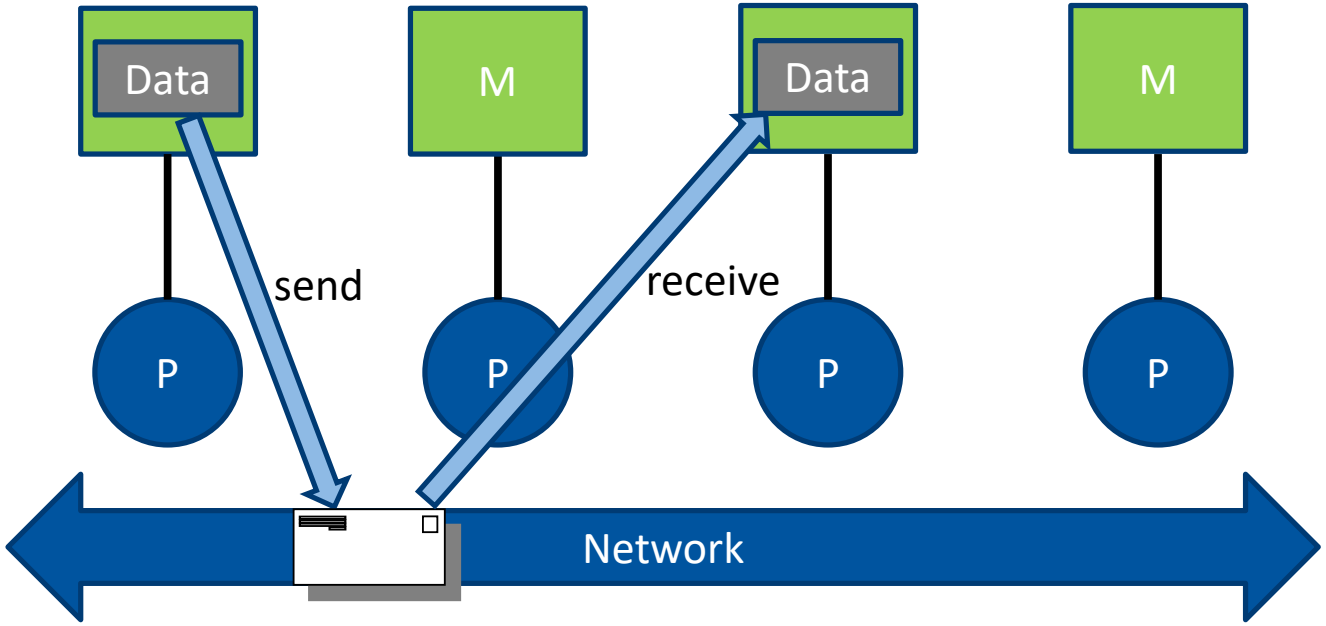
JÜLICH Forschungszentrum

PARALLEL PROGRAMMING

distributed memory

- each processor only access its own memory
- explicit communication between processors
- what we use in FLEUR: MPI

```
 if(isize.ne.1)then
  do ikpt=1,fullnkpts
   if(l_p0)then
    do cpu_index=1,isize-1
     if(mod(ikpt-1,isize).eq.cpu_index)then
      call MPI_RECV(
&             matrix4(1:num_dims,1:num_bands1,
&             1:num_bands2,ikpt),
&             num_bands1*num_bands2*num_dims,
&             CPP_MPI_COMPLEX,cpu_index,
&             ikpt,mpi_comm,stt,ierr)
     endif !processors
    enddo !cpu_index
   else
    if(mod(ikpt-1,isize).eq.irank)then
     call MPI_SEND(
&             matrix4(1:num_dims,1:num_bands1,
&             1:num_bands2,ikpt),
&             num_bands1*num_bands2*num_dims,
&             CPP_MPI_COMPLEX,0,
&             ikpt,mpi_comm,ierr)
    endif !processors
   endif ! l_p0
   call MPI_BARRIER(mpi_comm,ierr)
  enddo !ikpt
 endif !isize
```

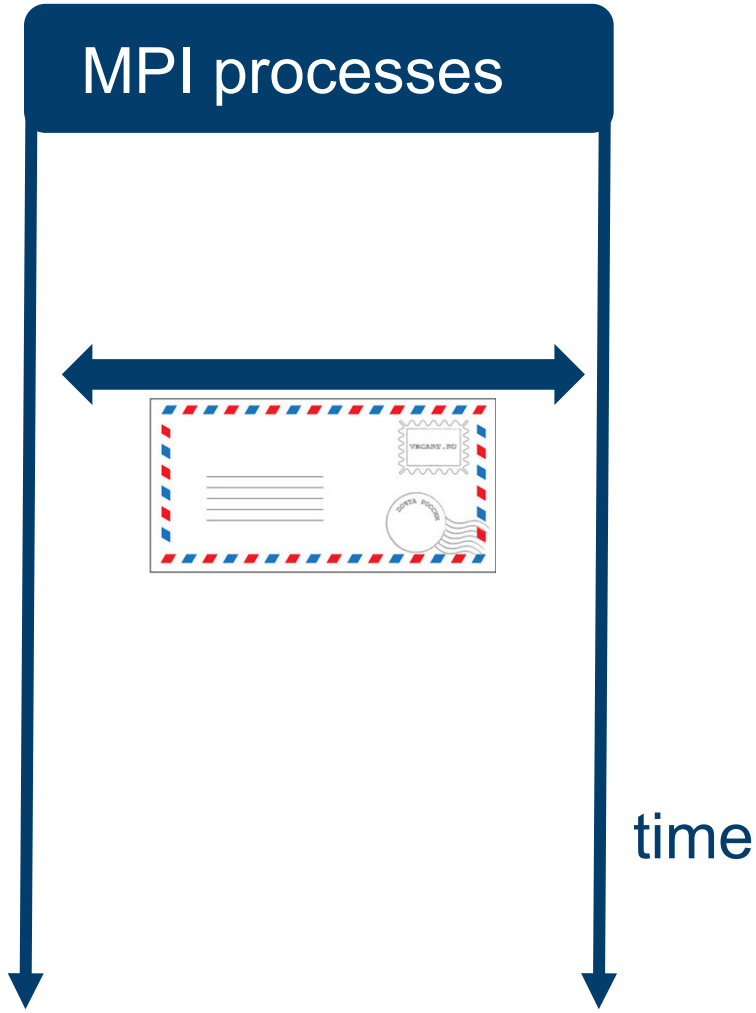


Source: H.Iliev

JÜLICH Forschungszentrum

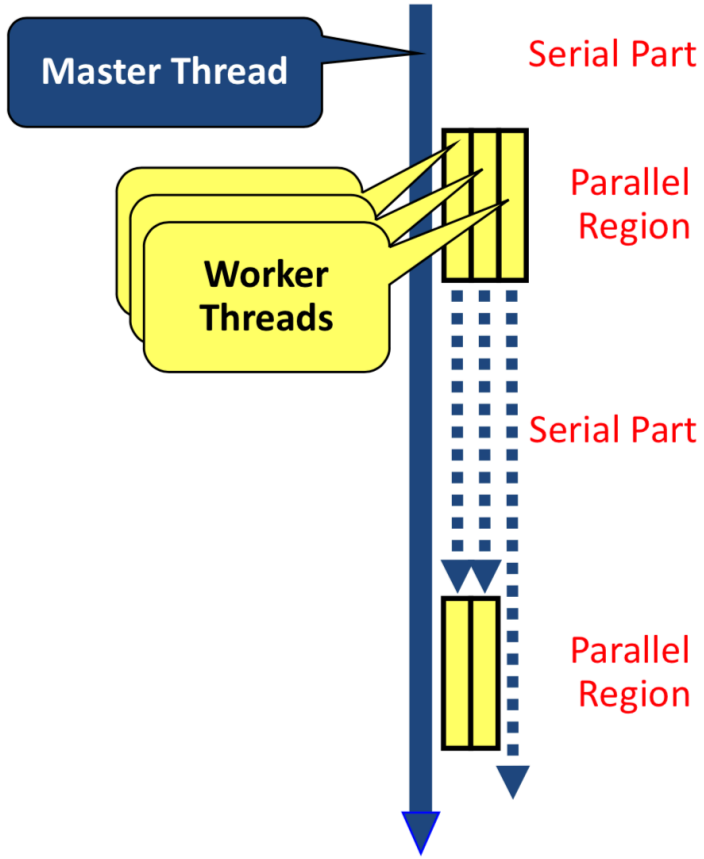
PARALLEL PROGRAMMING

execution model: MPI



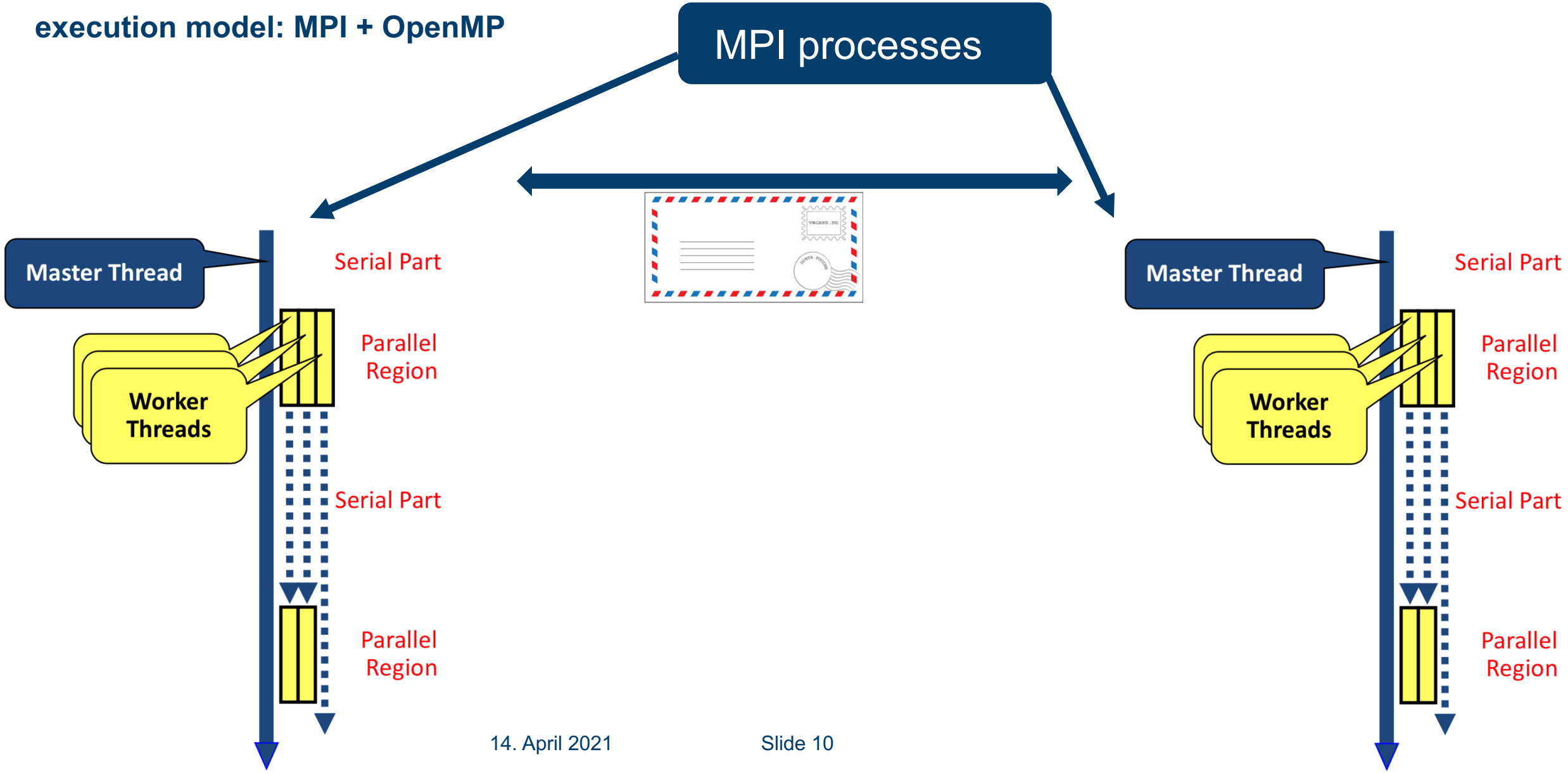
PARALLEL PROGRAMMING

execution model: OpenMP



PARALLEL PROGRAMMING

execution model: MPI + OpenMP

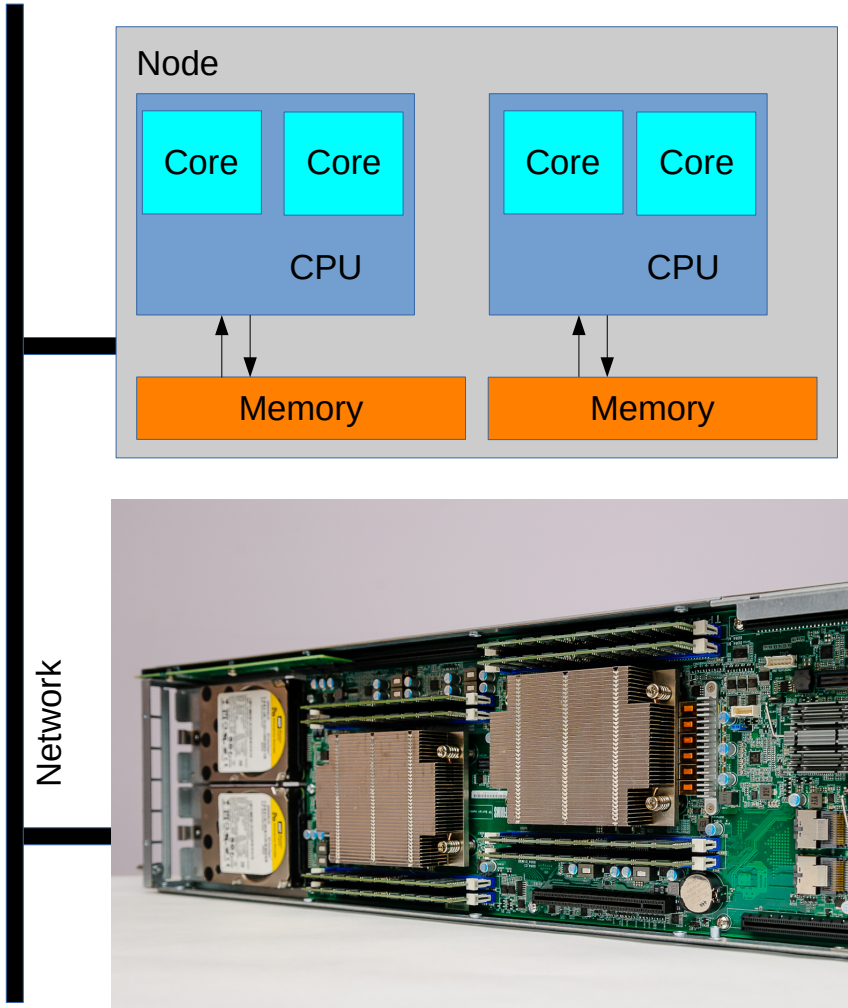


PARALLEL PROGRAMMING

mapping threads onto cores

- JURECA node: 24 cores
- How to map?
  - 1 MPI x 24 threads
  - 2 MPI x 12 threads
  - 4 MPI x 6 threads
  - 12 MPI x 2 threads
  - 24 MPI
- Good starting point :

  2-4 MPI processes per node



FLEUR

Potential

$$V[n]$$

H, S- dense Hermitian matrices

$$H = T + V[n]$$

Generalized eigenvalue problem

$$\boldsymbol{H\psi_i = \varepsilon_i S\psi_i}$$

Charge density

$$n(r) = \sum_{i}^{occ} |\psi_i|^2$$

FLEUR

Potential

$$V[n]$$

H, S- dense Hermitian matrices

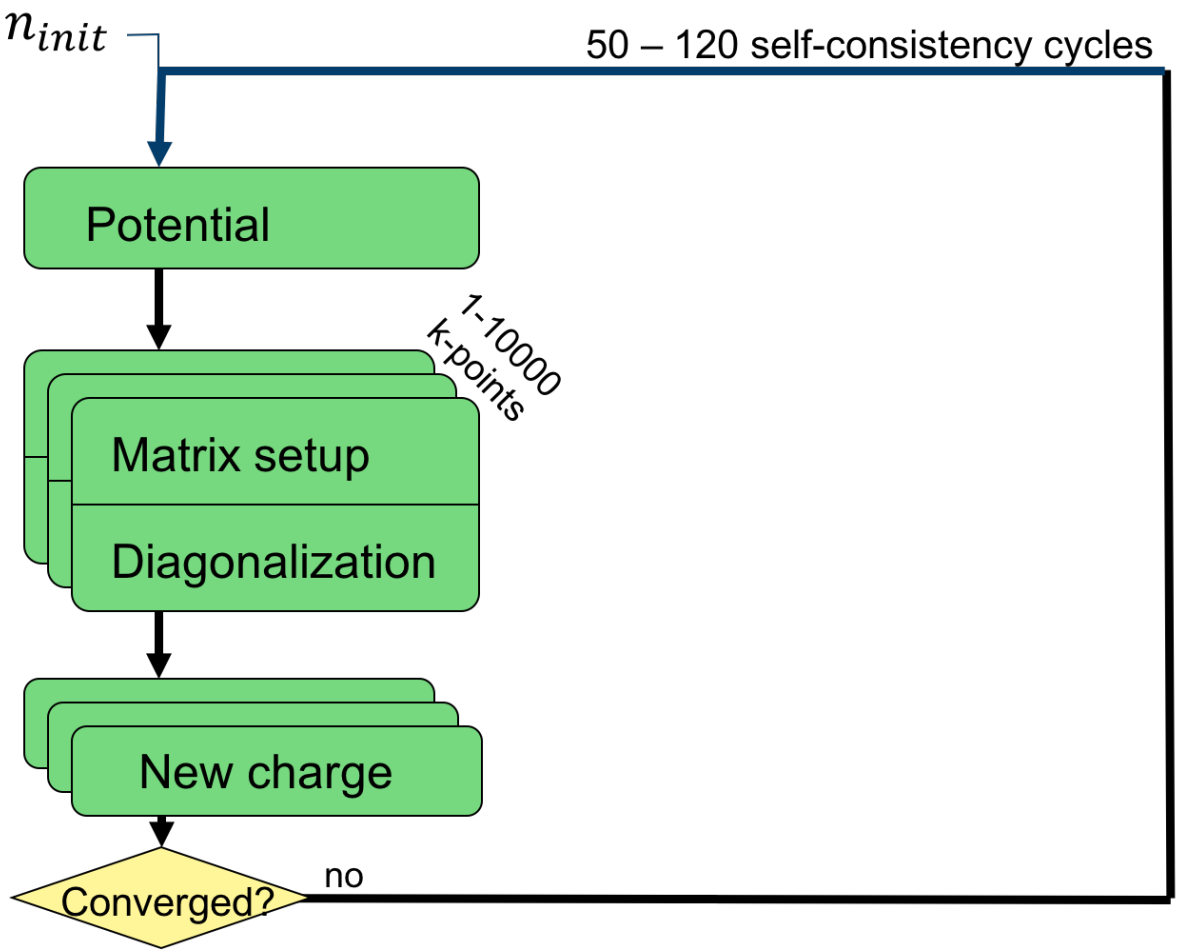
$$H = T + V[n]$$

Generalized eigenvalue problem

$$H\psi_i = \varepsilon_i S\psi_i$$

Charge density

$$n(r) = \sum_i^{occ} |\psi_i|^2$$



FLEUR

Potential

$$V[n]$$

H, S- dense Hermitian matrices

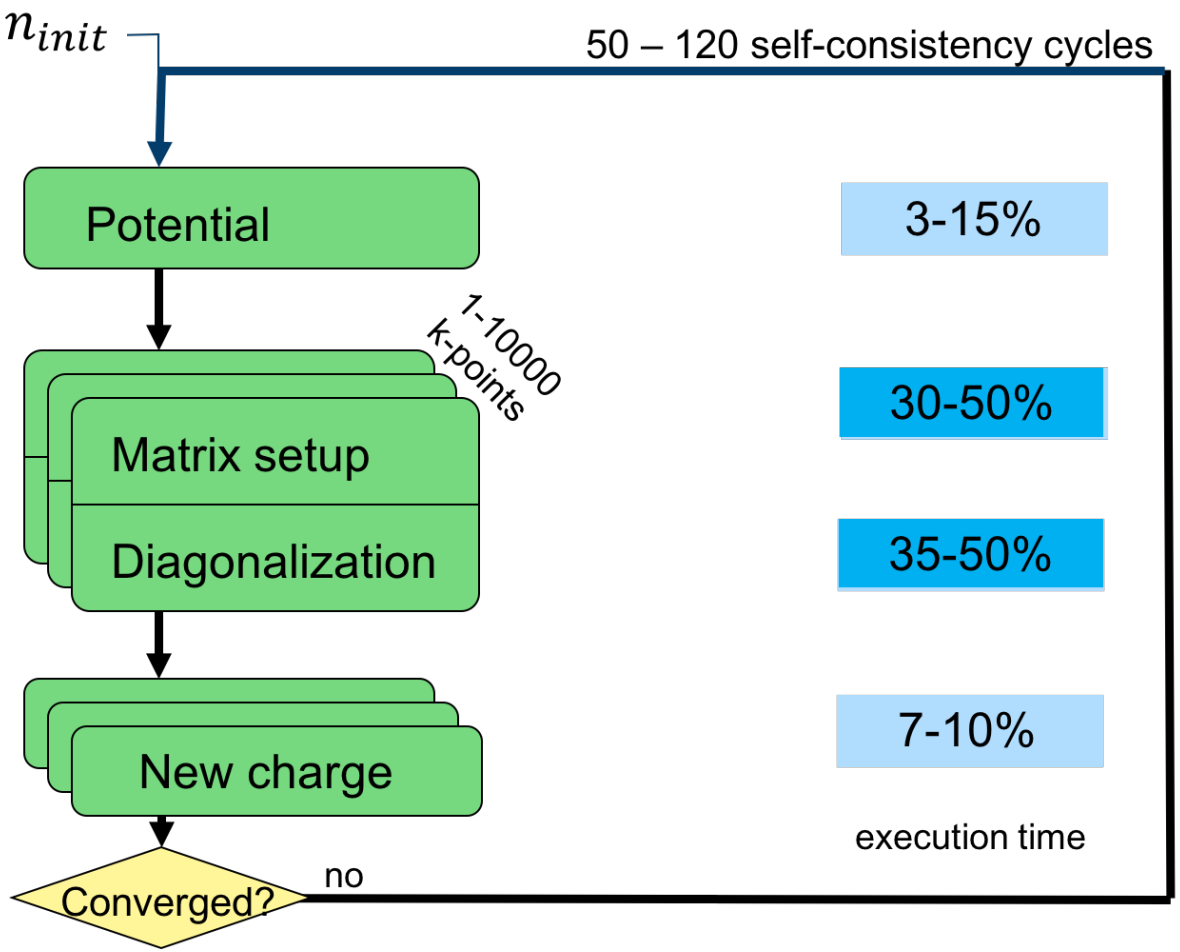
$$H = T + V[n]$$

Generalized eigenvalue problem

$$H\psi_i = \varepsilon_i S\psi_i$$

Charge density

$$n(r) = \sum_i^{occ} |\psi_i|^2$$



FLEUR

Potential

$$V[n]$$

H, S- dense Hermitian matrices

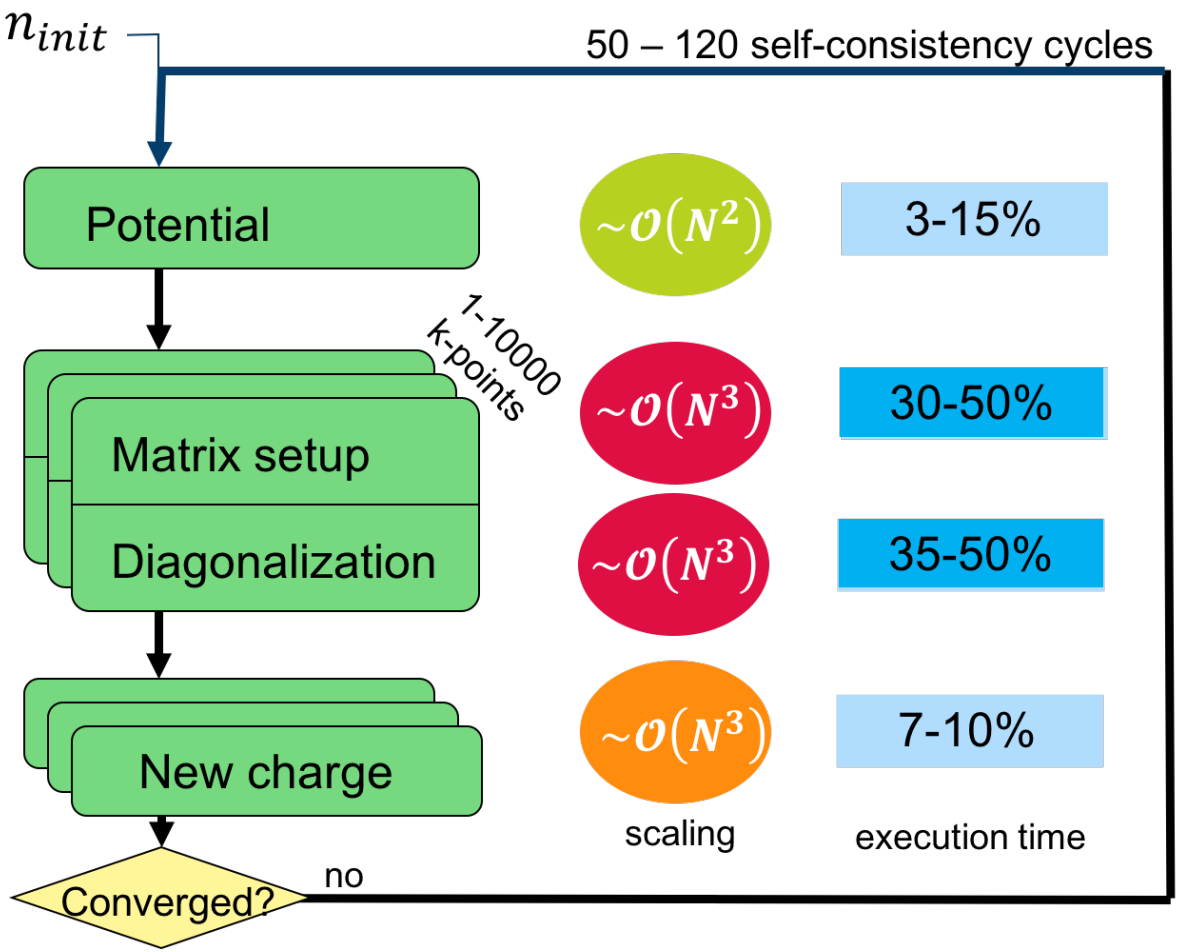
$$H = T + V[n]$$

Generalized eigenvalue problem

$$H\psi_i = \varepsilon_i S\psi_i$$

Charge density

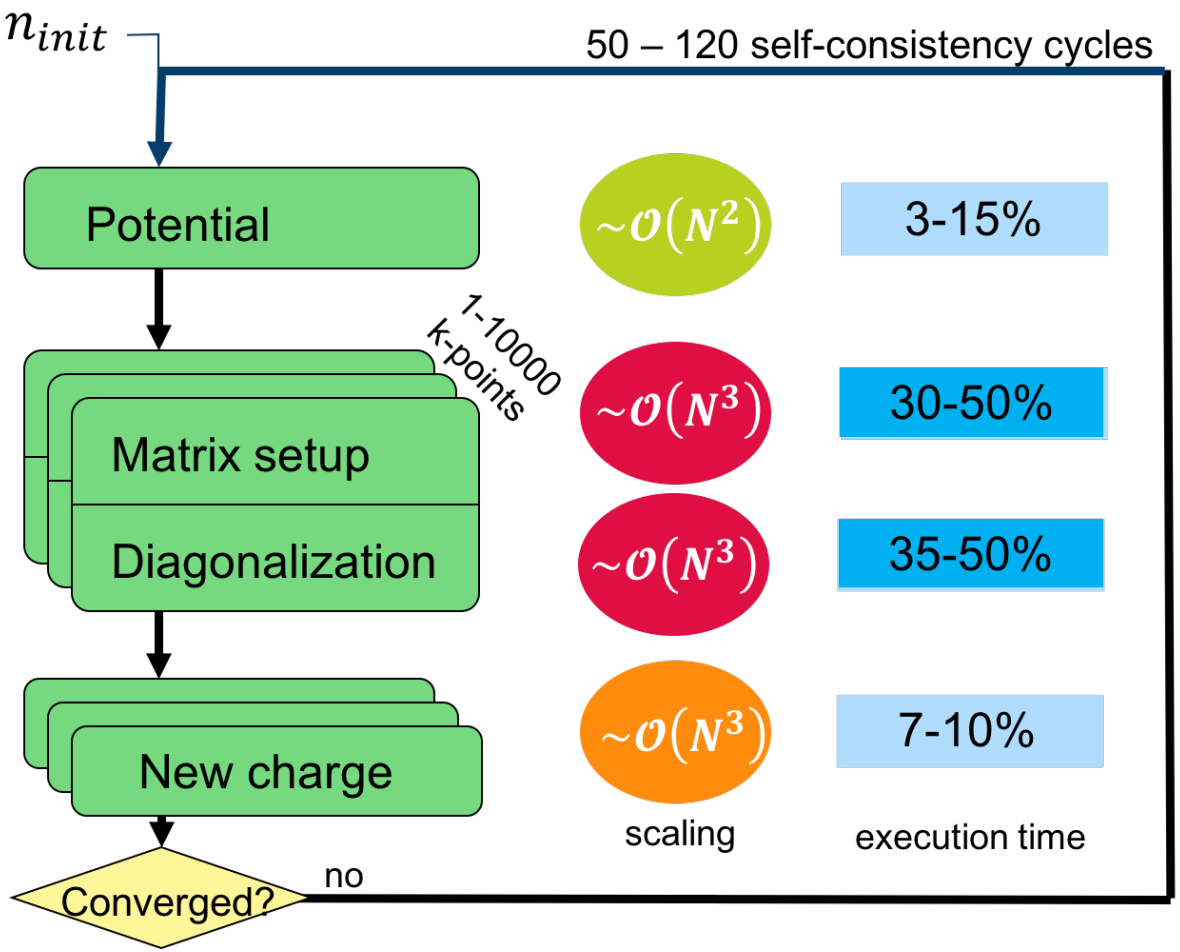
$$n(r) = \sum_{i}^{occ} |\psi_i|^2$$



FLEUR

Levels of parallelization:

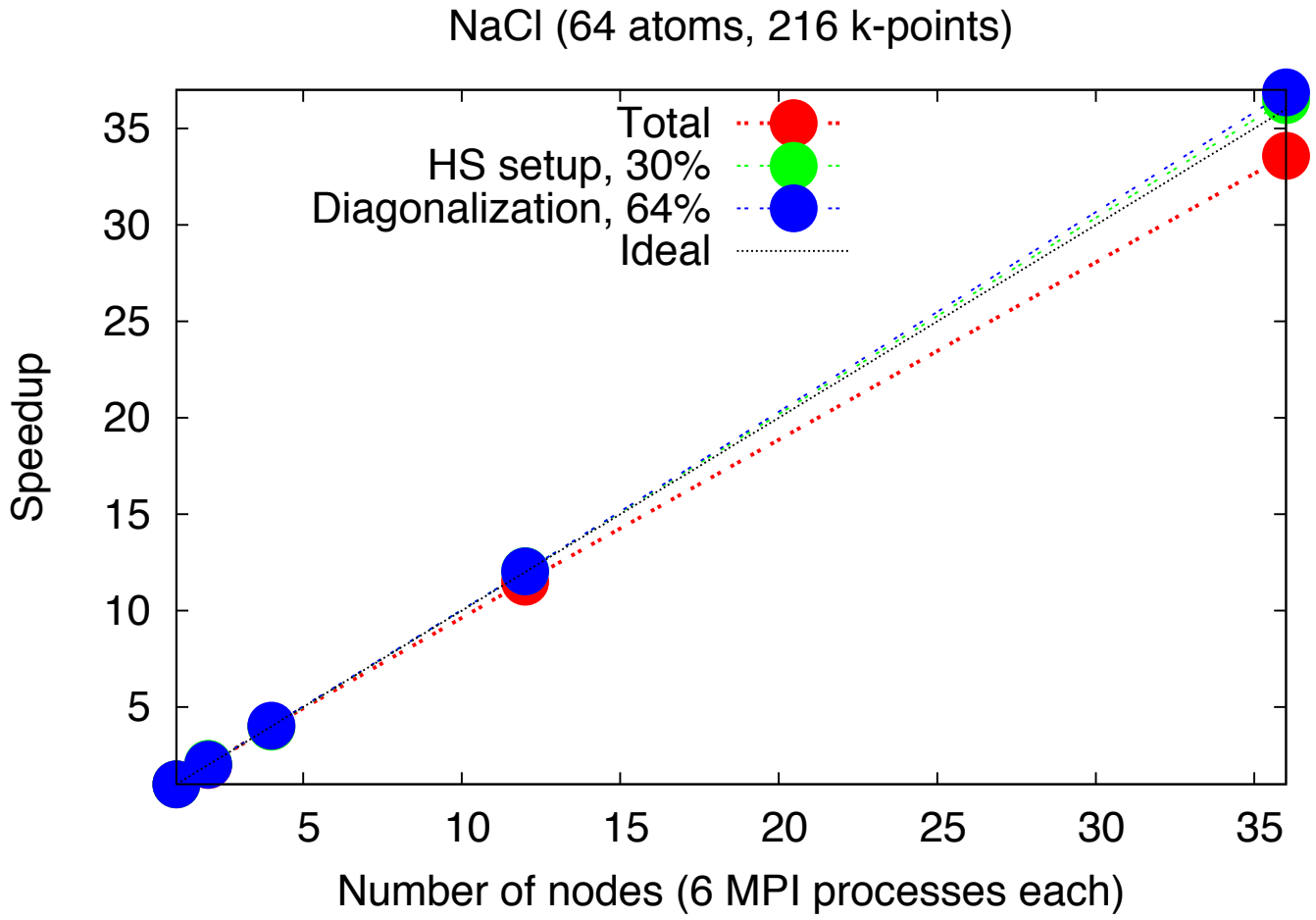
- MPI over k-points
- MPI eigenvector parallelization
- OpenMP parallelization
- vectorization



K-POINT PARALLELIZATION

ideal scaling

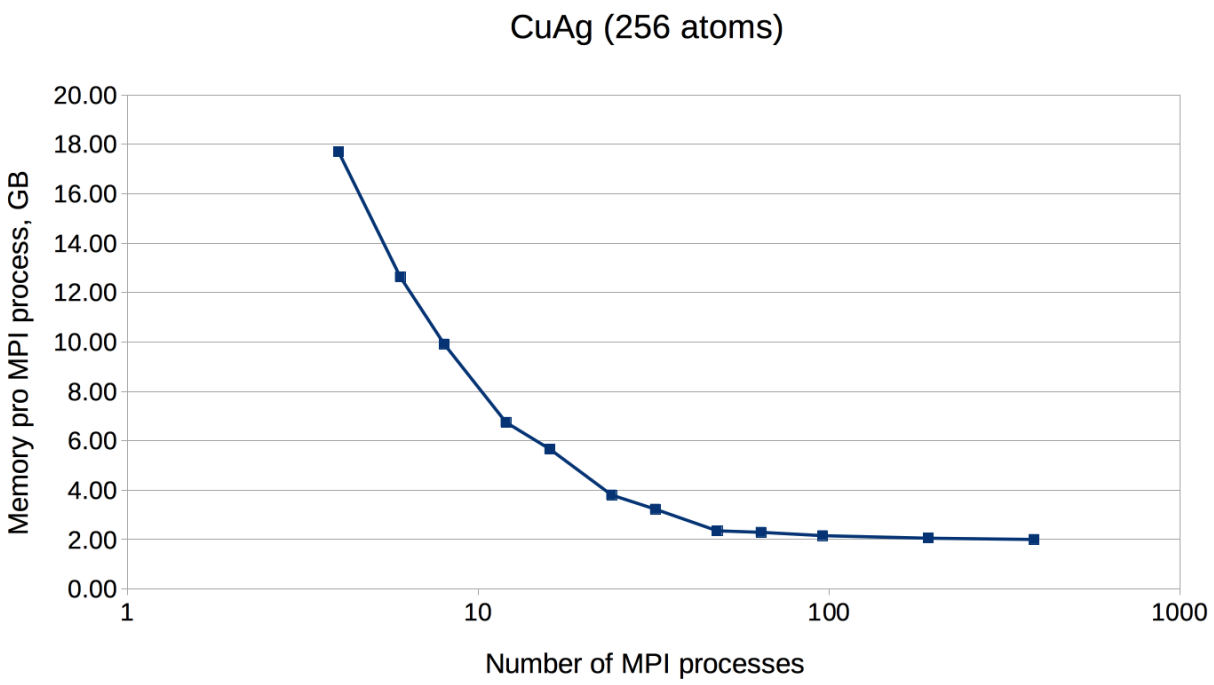
- most time-consuming part of the code are independent for different k-points
- FLEUR will distribute k-points to maximize the load balance
- good practice: try to adjust #k-points and #MPI processes

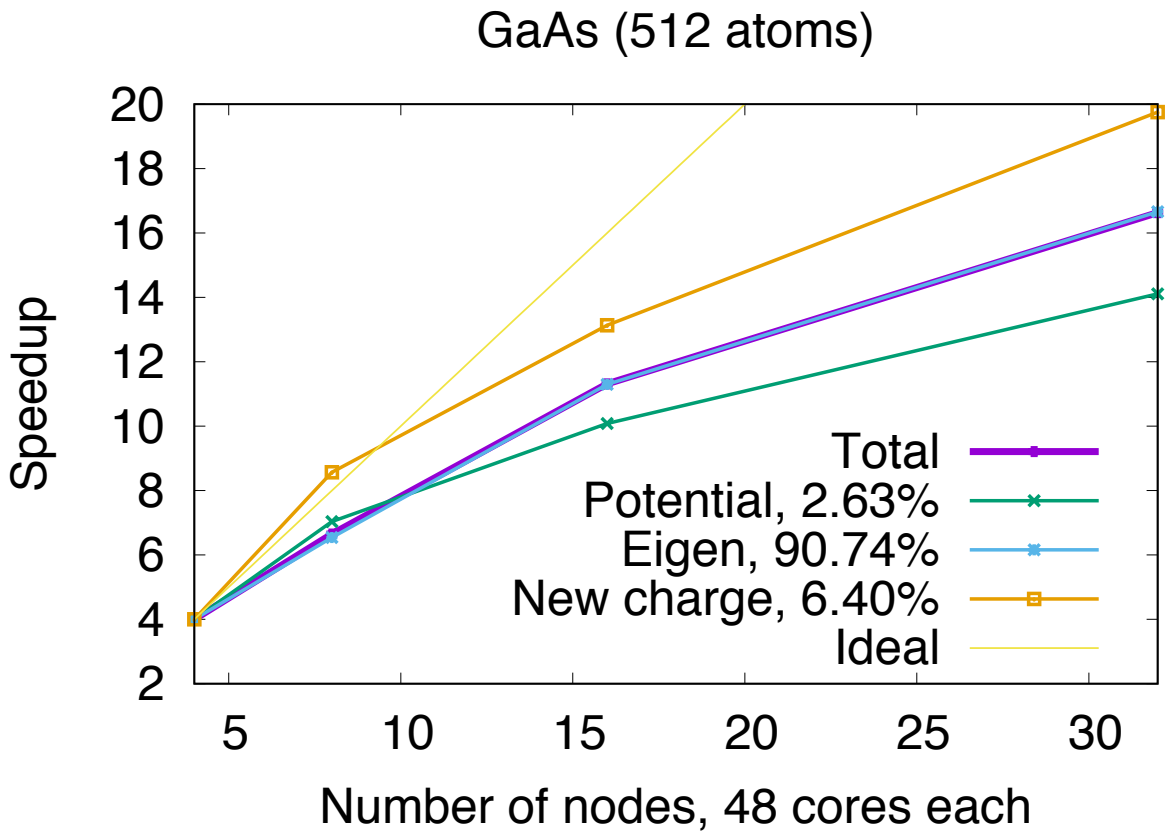


Intel Broadwell EP E5-2650v4
24 cores/node

THE EIGENVECTOR PARALLELIZATION

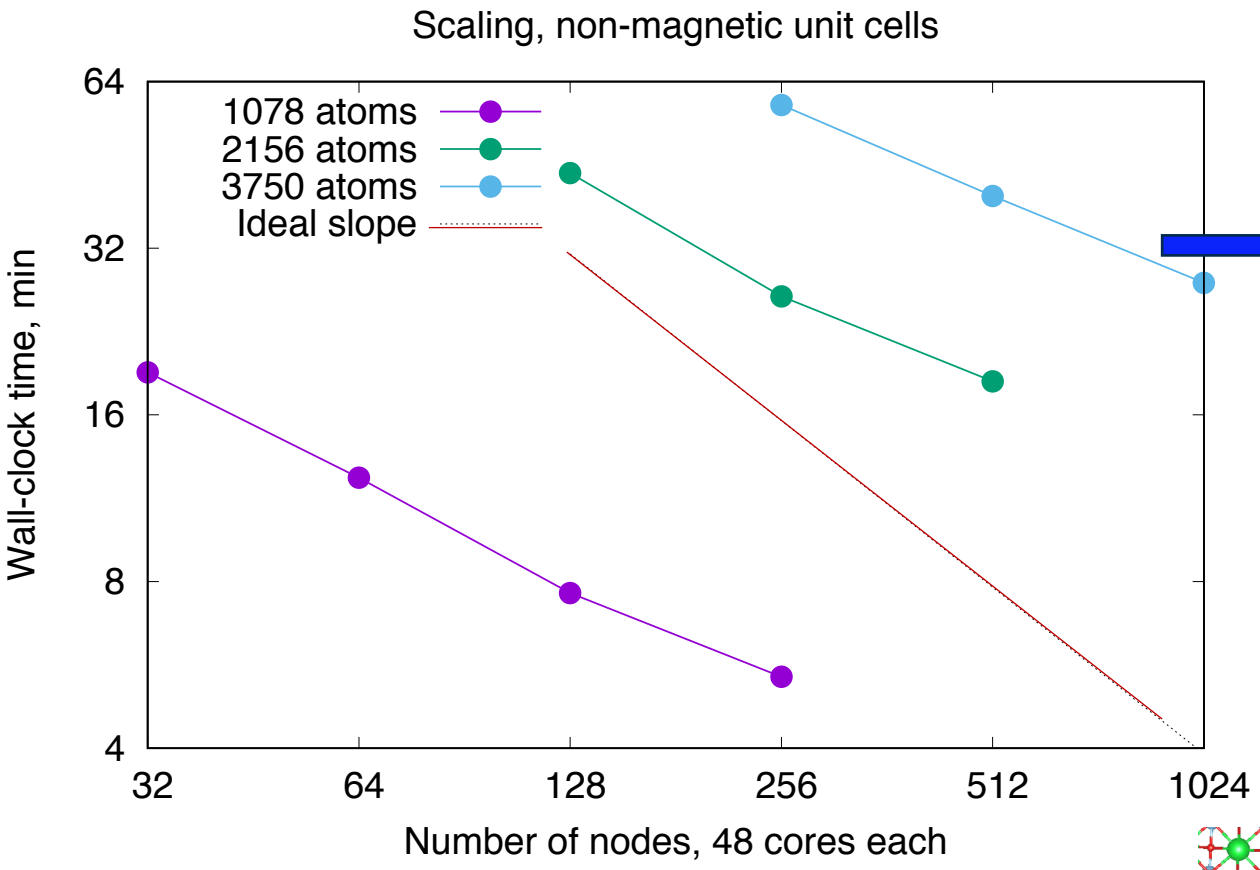
- gives an additional speedup
- allows to tackle larger systems
  (by reducing the memory
  usage per MPI process)





Intel Skylake Platinum 8160
48 cores/node

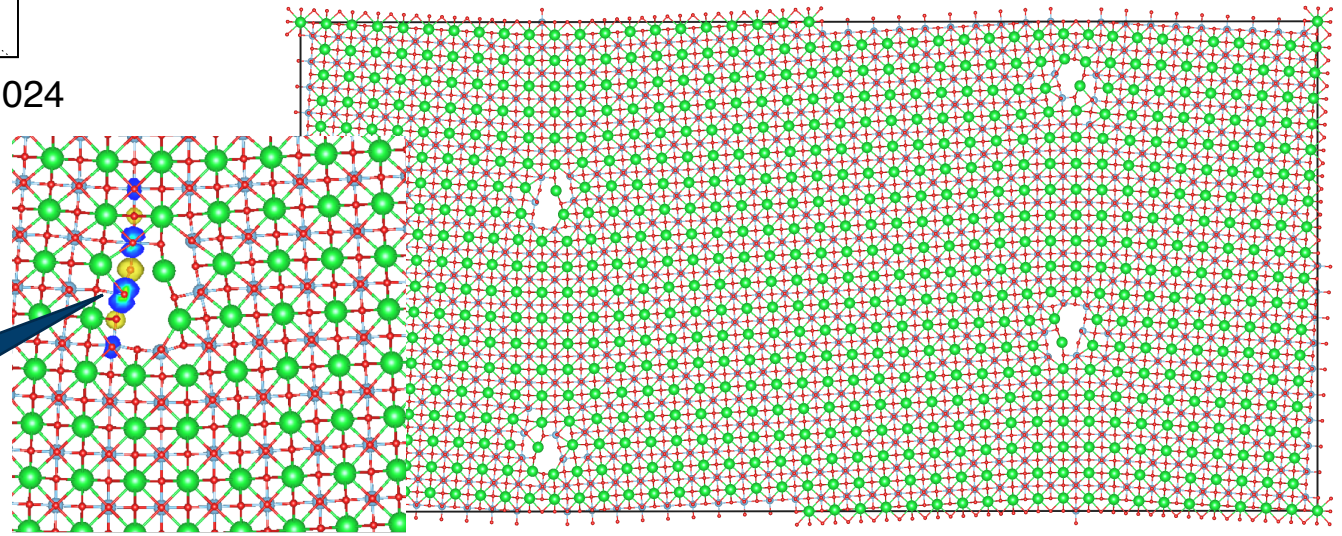
EXTENDED DEFECTS



$SrTiO_3$, 3750 atoms, 4 k-points

- 1024 nodes (Intel, 48 cores)
- Size of the matrices:
  345k x 345k (dense Hermitian)
- 100 minutes/ iteration
- 45 iterations to convergence
  (800k core-hours)

Intel Skylake Platinum 8174
48 cores/node

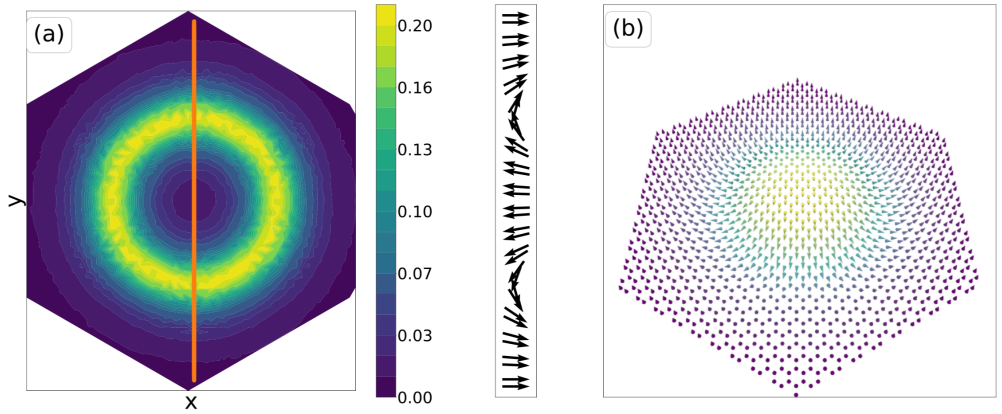


Defect state

NANO-SIZED MAGNETIC OBJECTS

Nanomagnetism:

- Complex spin-structures on the atomic scale
- Topological protection leads to stabilization
- Possible candidates for e.g. data-processing, neuromorphic or reservoir computing



M.Redies, Masterthesis 2019 RWTH

Basic concepts can be studied by multiscale approach:

1. DFT to extract material properties
2. Spin-dynamic simulations

Large scale FLEUR simulations:

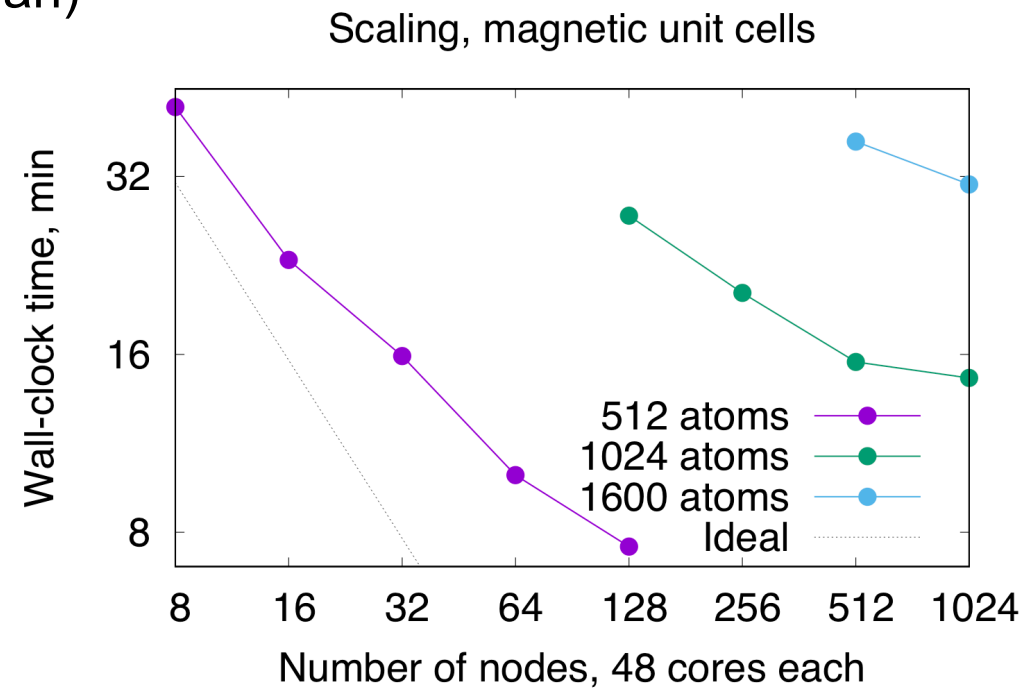
- Verify multiscale model
- Magnetism influences electronic structure

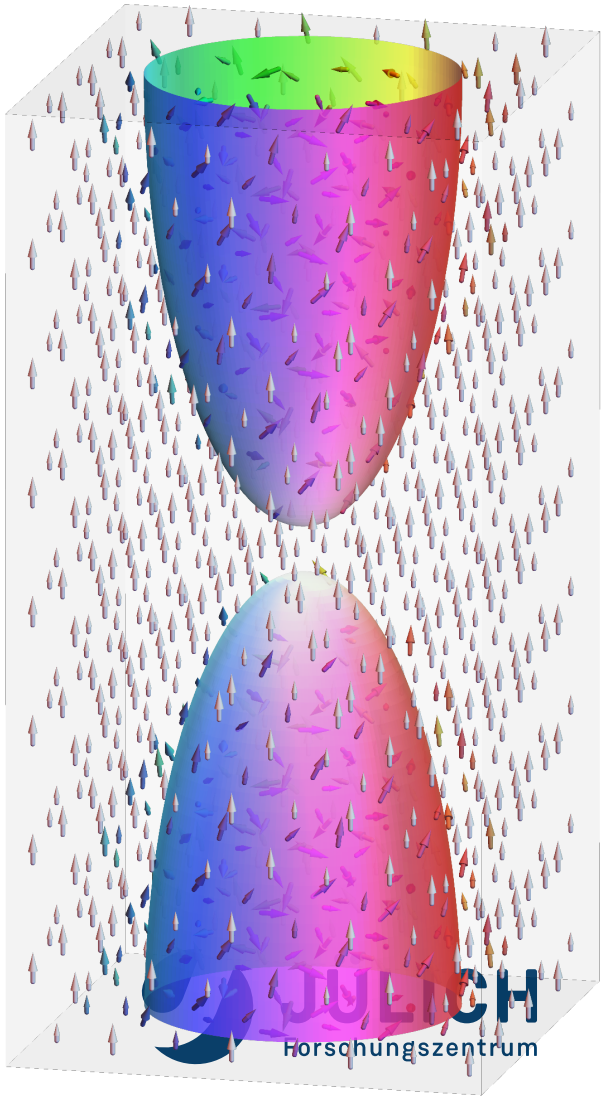
LARGE MAGNETIC SETUPS: A GLOBULE

MnGe Supercell 4x4x8, 1024 atoms, 1 k-point

- 256 nodes (Intel, 48 cores)
- Size of the matrices:
  156k x 156k (dense Hermitian)
- 25 minutes / iteration
- 100 iterations to convergence
  (2 Mio core-hours)

Intel Skylake Platinum 8174
48 cores/node





FLEUR IS PART OF MAX PROJECT

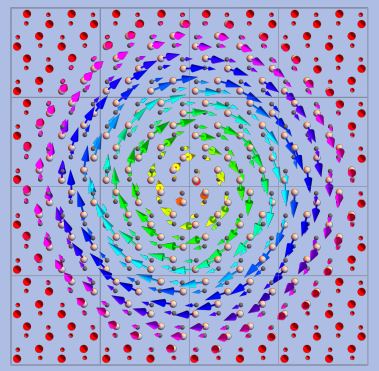
- MaX: Materials design at the eXascale
- a European centre of excellence
- Top500 #1: “Fugaku” 442 PFlops
- Exascale:

  coming soon

A PATH TO EXASCALE COMPUTING

Today:

Single magnetic structure



- several hundred nodes
- Mio core-hour

Magnetic structures are manifold

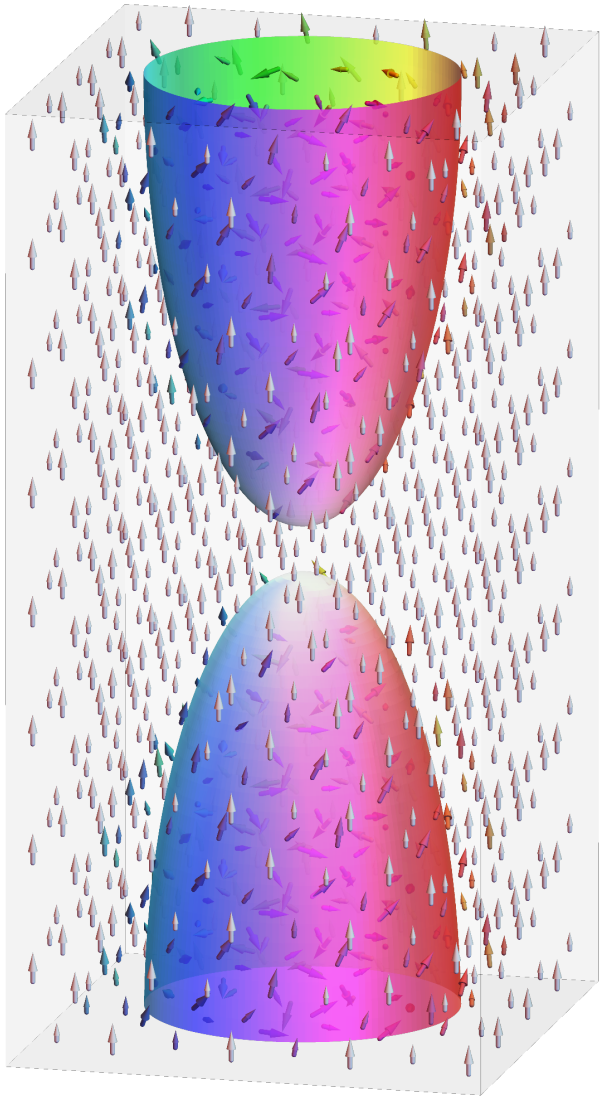
- Easy upscaling with HTC
- Intrinsically parallel

AiiDA

Magnetic structures evolve

- Time dependence easily requires orders of magnitude more computational effort

SUMMARY



- HPC cluster
  - many nodes, each with many cores
- Parallel programming
  - two paradigms: shared vs. distributed memory
- FLEUR
  - part of the MaX (Material design on eXascale) project
  - three levels of parallelization:
    1) MPI over k-point
    2) MPI over eigenvectors
    3) OpenMP
  - can be used on HPC clusters
  - able to simulate large unit cells (thousands of atoms)