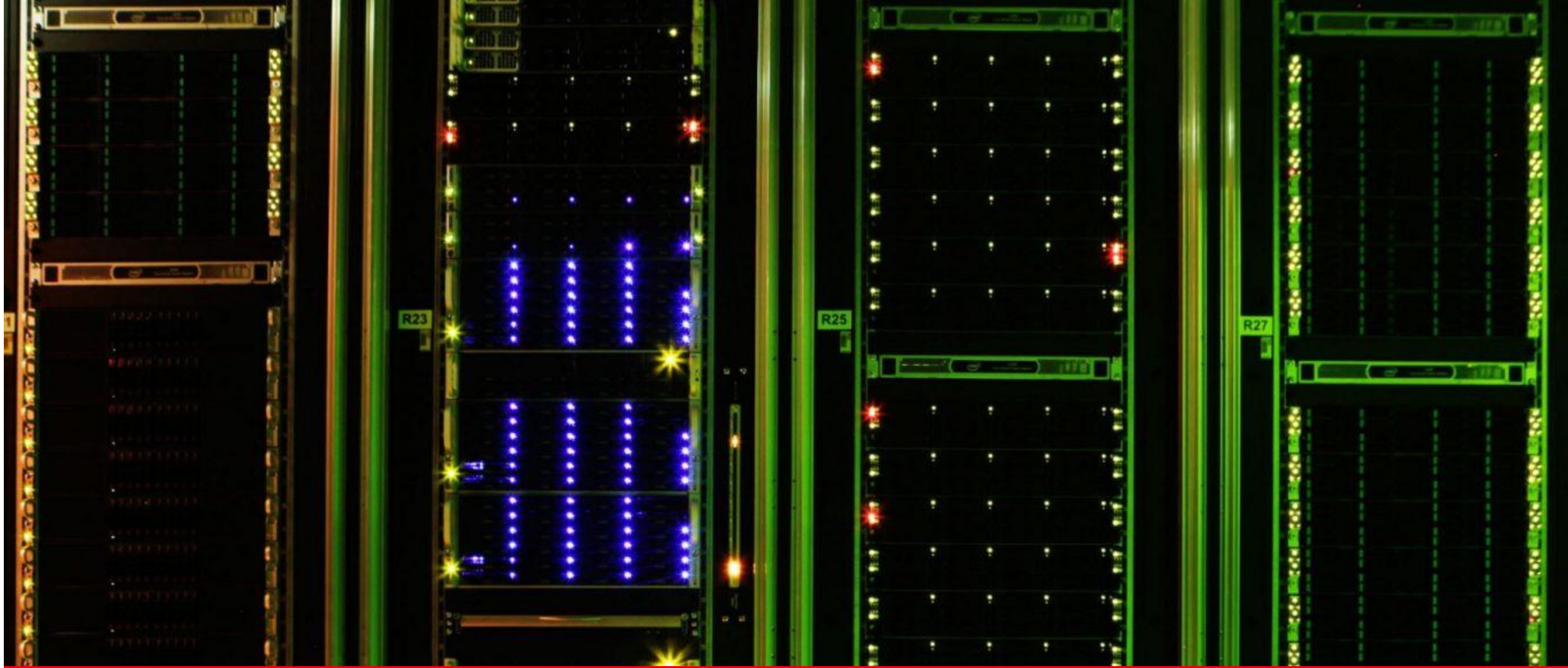


ORB5 solver optimization

Initial profiling

- CYCLONE-based simulation
- 1024 x 1024 x 256 grid, 800’000 ions, adiabatic electrons, 10 steps
- Focus on the Poisson/Ampère equations because dominant on GPU

64 tasks

Poisson	3191 s				
	field_dft	1656 s	52 %		
		pptransp		821 s	50 %
		fourcol		167 s	10 %
	field_idft	1417 s	44 %		
		pptranps		776 s	47 %
		fourcol		169 s	10 %

128 tasks

Poisson	8350 s				
	field_dft	4175.5 s	50 %		
		pptransp		3712 s	89 %
		fourcol		72 s	2 %
	field_idft	3978.5 s	48 %		
		pptranps		3578 s	90 %
		fourcol		74 s	2 %

Construction of the Poisson RHS in Fourier space

- In ORB5, the field solver works in Fourier space for the toroidal and poloidal directions
- Filtering is also applied to keep only relevant modes

$$\hat{\rho}_{ik}^{m} = \sum_{j=0}^{N_\theta - 1} \rho_{ijk} \exp\left(-\frac{2\pi \imath j m}{N_\theta}\right), \quad \forall m \in [m_1^f, m_2^f],$$

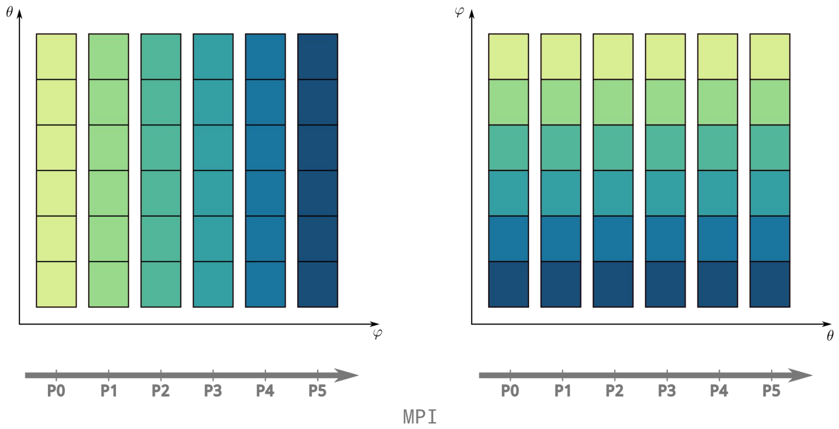
$$\hat{\hat{\rho}}_{i}^{nm} = \sum_{k=0}^{N_\varphi - 1} \hat{\rho}_{ik}^{m} \exp\left(-\frac{2\pi \imath k n}{N_\varphi}\right), \quad \forall n \in [n_1^f, n_2^f],$$

$$\forall n \in [n_1^f, n_2^f], \quad m \in [-nq(s) - \Delta m, -nq(s) + \Delta m],$$

Typically, $\Delta m \sim 5$

Parallel data transpose

- The FFTW algorithm used in ORB5 needs the dimension being transformed to be contiguous
- Since ORB5 uses domain cloning, parallel data transposes are needed



miniORB5

- A mini application called miniORB5 has been implemented
- It contains only the DFT (back and forth) of the RHS
- Lightweight and small, it allows one to easily test various strategies
- It copies ORB5 interface so that work can be copy/pasted back into ORB5
- Reproduces ORB5 timings

64 tasks

field_dft	310 s		
	pptransp	202 s	65 %
	fourcol	43 s	14 %
field_idft	351 s		
	pptranps	205 s	66 %
	fourcol	43 s	14 %

128 tasks

field_dft	564 s		
	pptransp	553 s	98 %
	fourcol	4 s	1 %
field_idft	228 s		
	pptranps	212 s	93 %
	fourcol	4 s	2 %

Alternative to original DFT algorithm
Local DFT

- Poloidal DFT same as original
- Toroidal DFT done partially on each subdomain
- Reduce scatter used to finalize the toroidal DFT

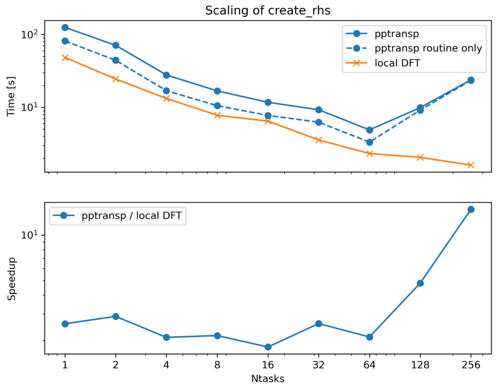
$$\hat{\rho}_{ik}^{m} = \sum_{j=0}^{N_\theta - 1} \rho_{ijk} \exp\left(-\frac{2\pi \imath j m}{N_\theta}\right), \quad \forall m \in [m_1^f, m_2^f],$$

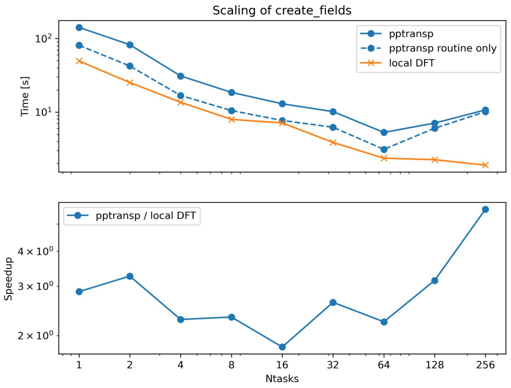
$$\hat{\hat{\rho}}_{i}^{nm} = \sum_{k=0}^{N_\varphi - 1} \hat{\rho}_{ik}^{m} \exp\left(-\frac{2\pi \imath k n}{N_\varphi}\right) = \sum_{s \in \mathcal{S}} \sum_{k=0}^{N_\varphi^{l} - 1} \hat{\rho}_{ik}^{m} \exp\left(-\frac{2\pi \imath k n}{N_\varphi}\right)$$

$$\forall n \in [n_1^f, n_2^f], \quad m \in [-nq(s) - \Delta m, -nq(s) + \Delta m],$$

Scalings on Jed (SCITAS - EPFL)

- Jed machine at EPFL
  - 420 nodes each equipped with 2 Intel Platinum with 36 cores (72 per node)
- GNU 11.3, FFTW 3.3.10, OpenMPI 4.1.3
- Timings done with ScoreP





- Same scaling up to 64 tasks, but “local DFT” ~2.3 faster
- Starting from 64 tasks, “pptransp” method does not scale anymore

Conclusions & outlooks

- The “local DFT” algorithm is ~2.3x faster and has a better scaling
- Better characterization needed with different hardware and MPI implementations
  - Preliminary results showed big influence from the MPI implementation
- Make a cost estimate of the MPI communication between the two methods
- Test FFTW capabilities to perform parallel transposes
- Re-test work developed by T. Ribeiro in the frame of the HLST