



ChEESE

XSHELLS

**A. Fournier, H. Frezat, T. Gastine, O. Hamitou,
G. Moguilny, N. Schaeffer**

September 17th, 2024



Project funded by EuroHPC under the grant agreement No 101093038.

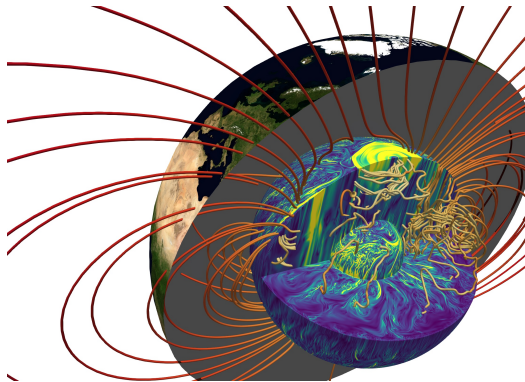


The XSHELLS code

<https://gricad-gitlab.univ-grenoble-alpes.fr/schaeffn/xshells>

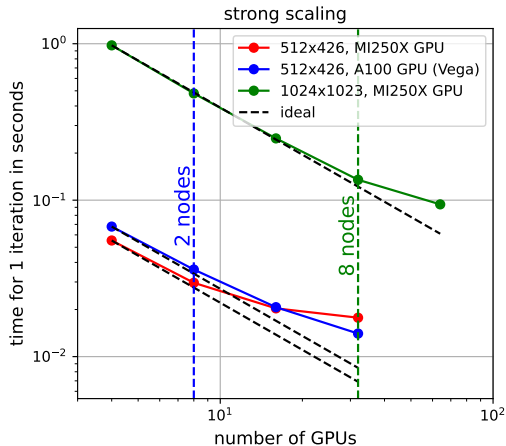
XSHELLS simulates the rotating Navier-Stokes equation in spherical geometry, coupled to the induction and heat equations.

- ▶ C++ code with MPI + OpenMP parallelization (multi-core version), MPI+CUDA/HIP (multi-gpu version)
- ▶ Mixed precision



WP2 (multi-GPU scalability)

Current state of scalability for medium and large test cases, on NVIDIA and AMD machines

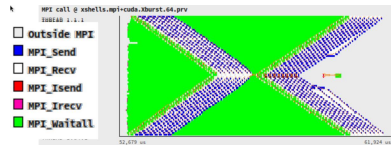


- ▶ Large case : ≈ 84 million DoF per GPU at 8 nodes
- ▶ Possible room for improvement for large number of nodes

WP2 (multi-GPU scalability)

PoP code audit:

- ▶ Main (expected) bottleneck is dependency chain between processes during (custom) linear solve (X-shape pattern)
- ▶ Unexpected: in addition to **communication latency, too little parallel work per GPU** during this phase leads to increased latency (small kernels do not scale well)



Mitigation strategy

1. MPI comms → direct memory read/write to peer GPU where applicable (within nodes)
2. Use some Alltoall (within nodes?) to make the solve more local and reduce dependency chain
3. Replace the current pipelined data-parallel solver with a truly parallel Thomas solver (see e.g. SPIKE algorithm, Tolmachev's pfSolve)

Contributions to WP3: Status (1/2)

Code	Validation	x86 profiling status	HBM study	ARM porting	ARM profiling	SVE study	Optimization status + comments
xshells	Done	Done	Done	Done (NVIDIA Grace)	Done (NVIDIA G.)	On going (NVIDIA G.)	Good vectorization on x86

Legend
Not tested
Fetch the code
Compile code
Run test cases
Validate results

Environment

- x86 : [GNU 13.2.0 + OpenMPI 5.0.0 or Intel 2023.1.0+ Intel MPI 2021.11]
- aarch64 : [GNU 13.2.0 + OpenMPI 5.0.0]

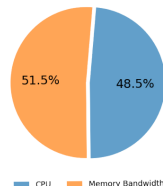
Compilation

- Disable internal vectorization (-DVEC=0) and enable compiler auto-vectorization (-march/-mcpu)
- Fix some x86 intrinsics for porting to aarch64

Run

- xsbig_hyb2 with 16 MPI ranks and 4 OMP threads per MPI rank with GNU compiler and OpenMPI
- On x86, GNU and Intel compiler give the same performance (<1.5% difference)

Machine	Internal timer	Exec time	FFTW features	Compiler options
Nvidia Grace	199.86 s	241.35 s	FMA + Neon+ SVE	"-O3 -mcpu=neoverse-v2+sve2"
Intel Icelake	187.92 s	212.85 s	FMA + AVX512	"-O3 -march=icelake-server"



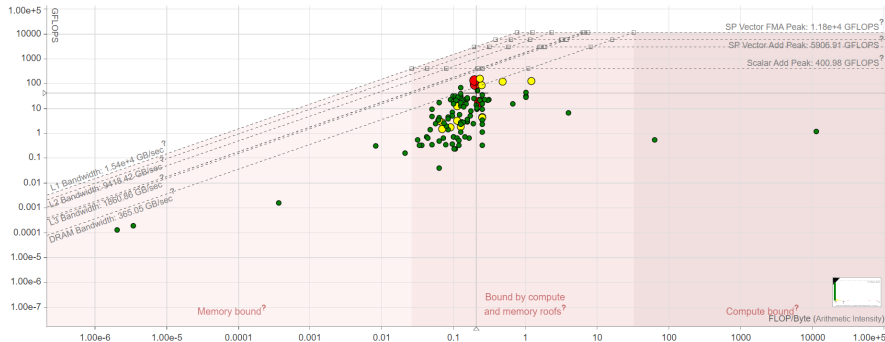
Large dependency on the memory bandwidth (> 50%), HBM is beneficial

Compiler / MPI	DDR5 (Internal timer)	HBM2e (Internal timer)	Ratio
GNU + OpenMPI	108.09 s	89.95 s	17% improvement

Stream bandwidth on DDR5	Stream bandwidth on HBM	Ratio
61 GB/s	208 GB/S	X 3.4 improvement

Contributions to WP3: Status (2/2)

- ▶ Vectorization reports show nearly 100% AVX2+AVX512
- ▶ Maqao profile on aarch64 shows the same hotspots as on x86



Loop id	Source Location	Source Function	Coverage run_0 (%)
3042	xsbig_hyb2 - SHqst.to.spat.kernel.c:376-394	_sy3_hi2_l	28.63
2802	xsbig_hyb2 - spat.to.SHqst.kernel.c:528-554 [...]	_an32_l	16.11
4302	xsbig_hyb2 -	n1bv_64	9.23
3001	xsbig_hyb2 - SHqst.to.spat.kernel.c:322-336	_sy1s_hi2_l	5.14
2920	xsbig_hyb2 - SHqst.to.spat.kernel.c:292-300	_sy1_hi2_l	3.25
2593	xsbig_hyb2 - spat.to.SH.kernel.c:387-401	_an12_l	3.21

Conclusions

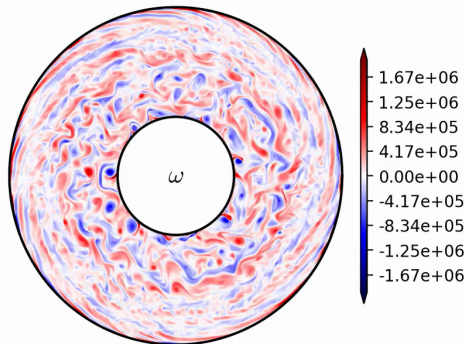
- ▶ Preliminary results show good auto-vectorization capabilities of GNU and Intel on the mini-app
- ▶ To be compared with manual vectorization with x86 intrinsics already implemented
- ▶ Large dependency of the mini-app on memory bandwidth, therefore HBM is greatly beneficial
- ▶ On-going SVE study

Contributions to WP5 – Task T5.1.5

Objective: To implement a dynamic subgrid-scale model based on deep learning to efficiently simulate turbulent dynamo action on geological time scales.

Status:

- ▶ 2D prototype: turbulent quasi-geostrophic convection in a spherical container
- ▶ differentiable spectral solver (JAX, github.com/google/jax)



Offline and online subgrid-scale (SGS) modelling

Projection operator

$$\mathcal{T}(\mathbf{y}) = \bar{\mathbf{y}}, \mathbf{y} \in \Omega, \bar{\mathbf{y}} \in \bar{\Omega}, \bar{\Omega} \subset \Omega.$$

True and reduced-order dynamics

$$\partial_t \mathbf{y} = f(\mathbf{y}),$$

$$\partial_t \bar{\mathbf{y}} = g(\bar{\mathbf{y}}) + \mathcal{M}(\bar{\mathbf{y}}).$$

$\mathcal{M}$: trainable SGS model $\approx R(\mathbf{y}) = \mathcal{T}(f(\mathbf{y})) - g(\mathcal{T}(\mathbf{y}))$.

- ▶ **A priori (offline) learning**: find optimal $\mathcal{M}$ given groundtruth **static** dataset $\{R(\mathbf{y}_i), \bar{\mathbf{y}}_i\}$.
- ▶ **A posteriori (online) learning**: find optimal $\mathcal{M}$ given **time sequence** of $\{\mathcal{T}(\mathbf{y}_i), \bar{\mathbf{y}}_i\}$.

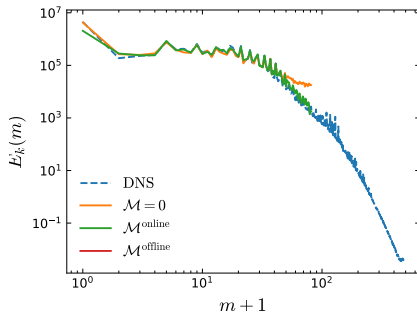
Online learning: computation of the gradient wrt all the components of the forward model.

Results

- ▶ High resolution ($\mathbf{y}$): 384 modes in r and φ . Reduced resolution ($\bar{\mathbf{y}}$): 64 modes in r and φ .
- ▶ Neural network: ConvNext (github.com/facebookresearch/ConvNeXt), 4 blocks, features [32, 64, 128, 128], conv. kernel 7x7.

Offline learning: simulation blows up

Online learning: stable and accurate prediction

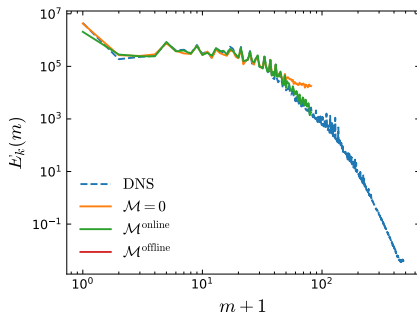


Results

- ▶ High resolution ($\mathbf{y}$): 384 modes in r and φ . Reduced resolution ($\bar{\mathbf{y}}$): 64 modes in r and φ .
- ▶ Neural network: ConvNext (github.com/facebookresearch/ConvNeXt), 4 blocks, features [32, 64, 128, 128], conv. kernel 7x7.

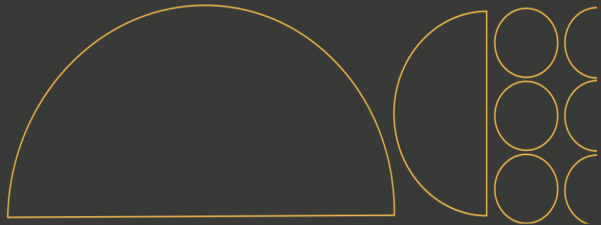
Offline learning: simulation blows up

Online learning: stable and accurate prediction



- ▶ En route to 3D: Differentiable programming seems crucial
- ▶ Design of $\mathcal{M}$: catalog of high-fidelity xshells simulations (EuroHPC regular access call: 34,000 node hours on VEGA/GPU)

Þakka þér fyrir!



@cheese-coe



<http://cheese2.eu>



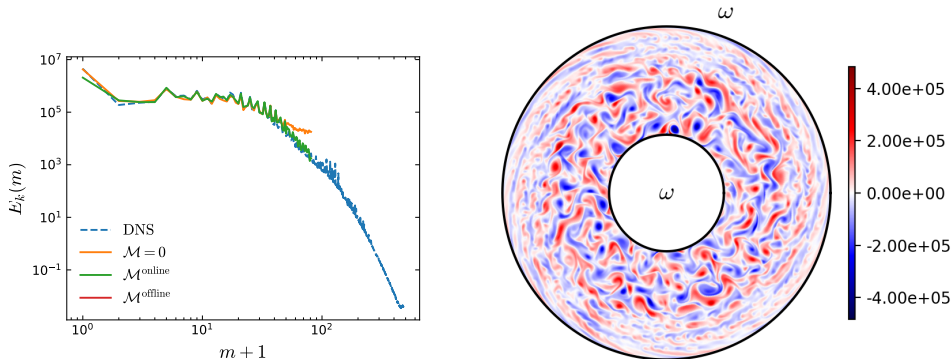
@cheese-coe

Results

- ▶ High resolution ($\mathbf{y}$): 384 modes in r and φ . Reduced resolution ($\bar{\mathbf{y}}$): 64 modes in r and φ .
- ▶ Neural network: ConvNext (<https://github.com/facebookresearch/ConvNeXt>), 4 blocks, features [32, 64, 128, 128], conv. kernel 7x7.

Offline learning: simulation blows up

Online learning: stable and accurate prediction

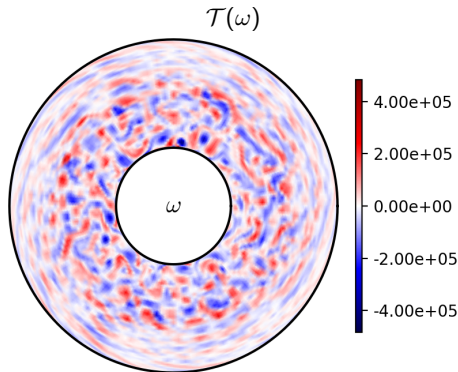
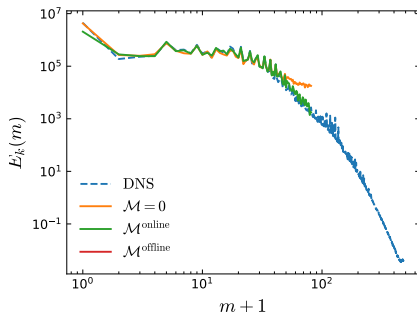


Results

- ▶ High resolution ($\mathbf{y}$): 384 modes in r and φ . Reduced resolution ($\bar{\mathbf{y}}$): 64 modes in r and φ .
- ▶ Neural network: ConvNext (<https://github.com/facebookresearch/ConvNeXt>), 4 blocks, features [32, 64, 128, 128], conv. kernel 7x7.

Offline learning: simulation blows up

Online learning: stable and accurate prediction

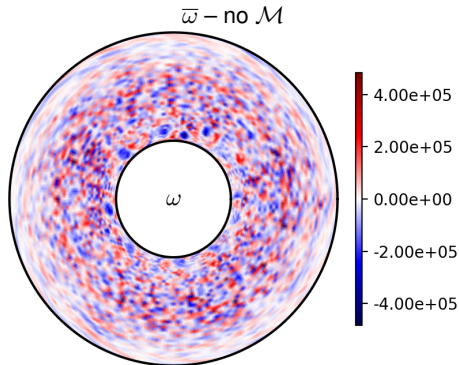
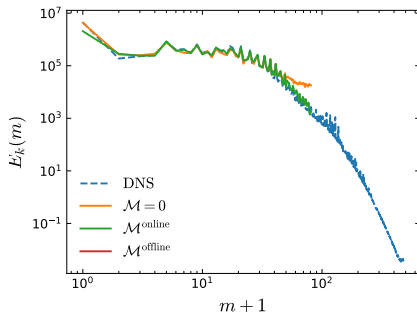


Results

- ▶ High resolution ($\mathbf{y}$): 384 modes in r and φ . Reduced resolution ($\bar{\mathbf{y}}$): 64 modes in r and φ .
- ▶ Neural network: ConvNext (<https://github.com/facebookresearch/ConvNeXt>), 4 blocks, features [32, 64, 128, 128], conv. kernel 7x7.

Offline learning: simulation blows up

Online learning: stable and accurate prediction



Results

- ▶ High resolution ($\mathbf{y}$): 384 modes in r and φ . Reduced resolution ($\bar{\mathbf{y}}$): 64 modes in r and φ .
- ▶ Neural network: ConvNext (<https://github.com/facebookresearch/ConvNeXt>), 4 blocks, features [32, 64, 128, 128], conv. kernel 7x7.

Offline learning: simulation blows up

Online learning: stable and accurate prediction

