Accelerating Earth System Models

Portability of NEMO diagnostics to GPU

Sergi Palomas
Mario Acosta
Ramón Grau

Index

1. Introduction
2. Context
3. Objectives
4. Diagnostics analysis
5. GPU diagnostic implementation
6. Results
7. Conclusions and future work

Introduction

- Earth system models are widely used in meteorological institutions and universities for weather and climate prediction studies

Introduction

- Earth system models are widely used in meteorological institutions and universities for weather and climate prediction studies

- Complex models that require a huge amount of computing resources/money

Introduction

- Earth system models are widely used in meteorological institutions and universities for weather and climate prediction studies

- Complex models that require a huge amount of computing resources/money

- Performance studies are key to:
  - Achieve the best *time to Solution* in operational forecasts
  - Reduce the costs of running the model on HPC infrastructures
  - Save time to the scientists

Introduction

- Earth system models are widely used in meteorological institutions and universities for weather and climate prediction studies

- Complex models that require a huge amount of computing resources/money

- Performance studies are key to:
  - Achieve the best *time to Solution* in operational forecasts
  - Reduce the costs of running the model on HPC infrastructures
  - Save time to the scientists

- NEMO is a state of the art framework for oceanographic studies. Its execution is extended by Diagnostics

Context

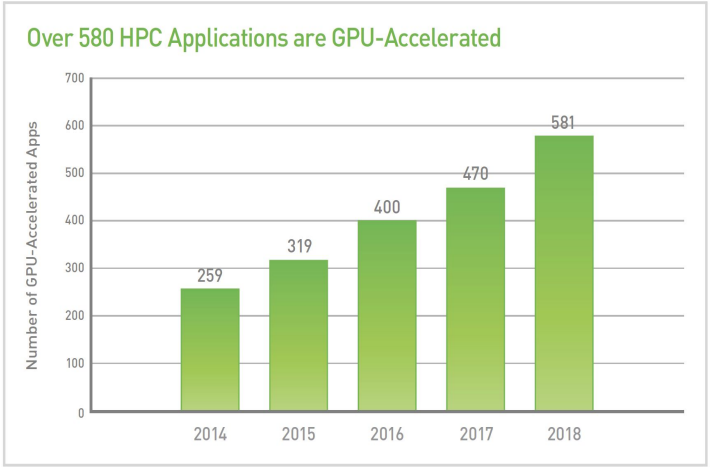
- Developed with the Computational Earth Science group (Earth-CES) at Barcelona Supercomputing Center (BSC-CNS)

- Earth model performance analysis team is in charge of providing new and more efficient approaches for Earth system models

- Earth models keep increasing in complexity and resolution every year

Context

GPUs usage in HPC is increasing



Good results for physics, ML and Data Science

Fewer attempts for large Earth models

COSMO
Nvidia on NEMO

Costs maintaining two versions
Hybrid computation

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

NEMO

NEMO

The Nucleus for European Modelling of the Ocean (NEMO) is a state-of-the-art modeling framework used for oceanographic research, climate studies, seasonal forecasting and also for operational oceanography

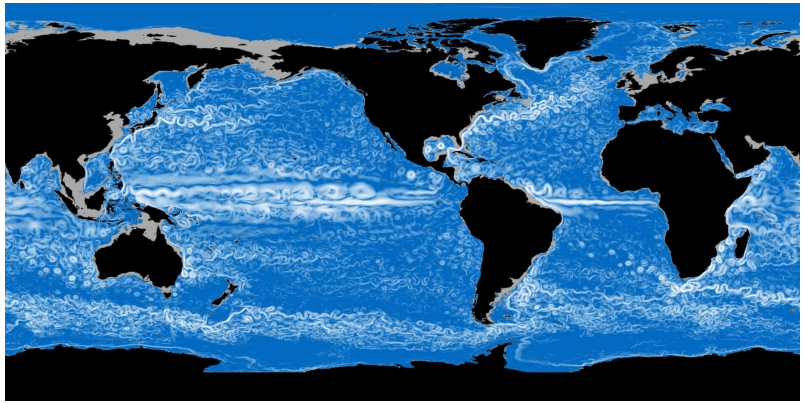
Maintained by a European community

Used in several studies that take

thousands of computing hours

Diagnostics are executed at the end of

every time-step to prepare the post-processing



NEMO

ORCA025 grid resolution:

- x=1442 y=1050 z=75

200.000 lines in Fortran 90

Parallelized with MPI (domain decomposition)

XIOS as IO server

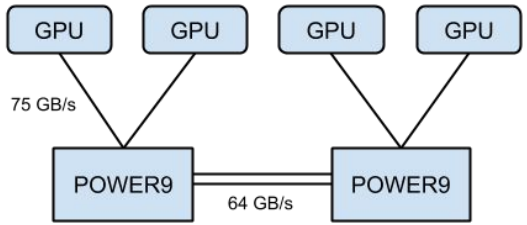
Environment

MareNostrum 4 used to run NEMO at BSC

CTE IBM Power9:

- 52 nodes
- 2 x Power9 CPUs each
- 40 cores/node
- 4 x Nvidia Tesla v100 16GB each

MareNostrum 4



Power9 intra-node connection

Objectives

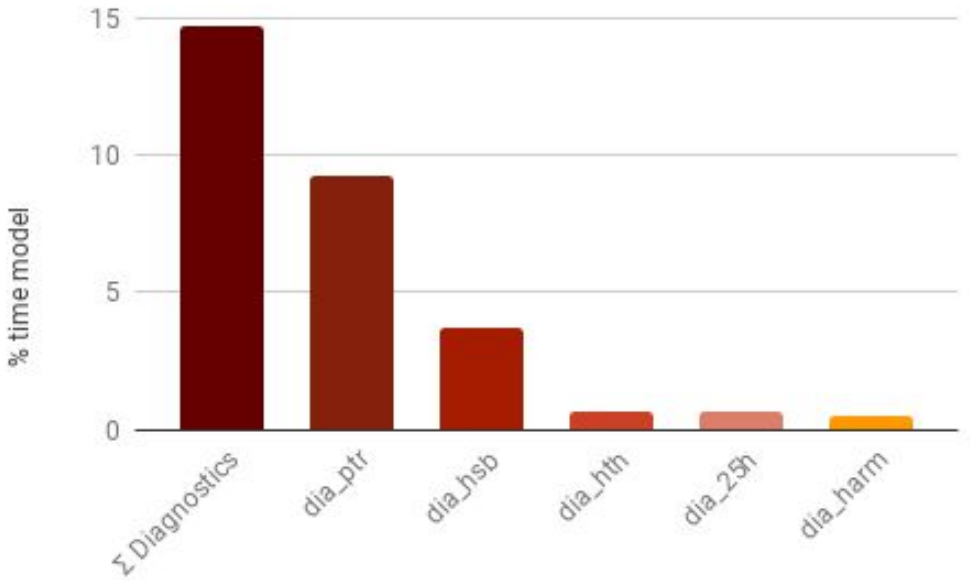
Diagnostics results are not needed for NEMO anymore. GPUs can be used to take them out of the critical path

- Improve NEMO execution time
- Use the GPU as an extra device to run the diagnostics asynchronously
    - Analyze the impact of diagnostics
    - Select a target diagnostic that bottleneck the execution
    - Propose solutions based on GPU
    - Compare and evaluate the new implementation

Diagnostics analysis

Diagnostics take almost 15% of the total execution time of NEMO

dia_ptr and dia_hsb are the most computer-hungry ones



Percentual time of most time-consuming diagnostics in NEMO

Select a target diagnostic

dia_ptr

- Control flow divergence
- Shared functions called 50 times per time-step
- Output vs. Computation

dia_hsb

- Imply all local domain
- Same operation
- Reduction
- Reuse of data
- Output scalar values

GPU implementation

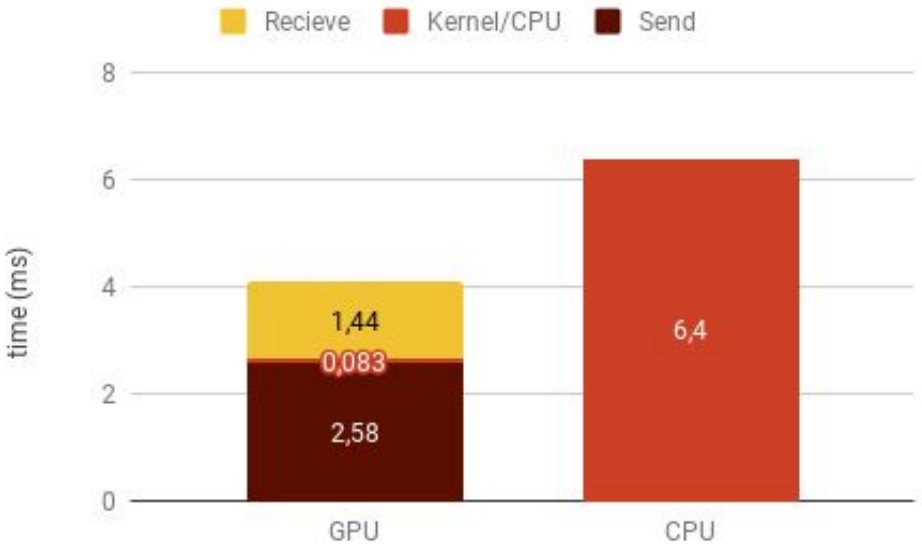
Using the data of a single MPI process: (x=87, y=105, z=75)

Independent “example model” to develop, compare and test the results without running the complete model

1. dia_hsb
    a. Data transfers optimization → pageable vs pinned memory
    b. Environmental implications → CPU-GPU intra-node connection
2. Reduction
    a. CUDA architecture → Warps, memory coalescing and bank conflicts

dia_hsb kernel

1. Send data to Device
2. Kernel:
   a. Blocks of 512 threads
   b. As many threads as elements (685125)
3. CPU receives the result



Absolute time (ms) per GPU procedure vs. CPU

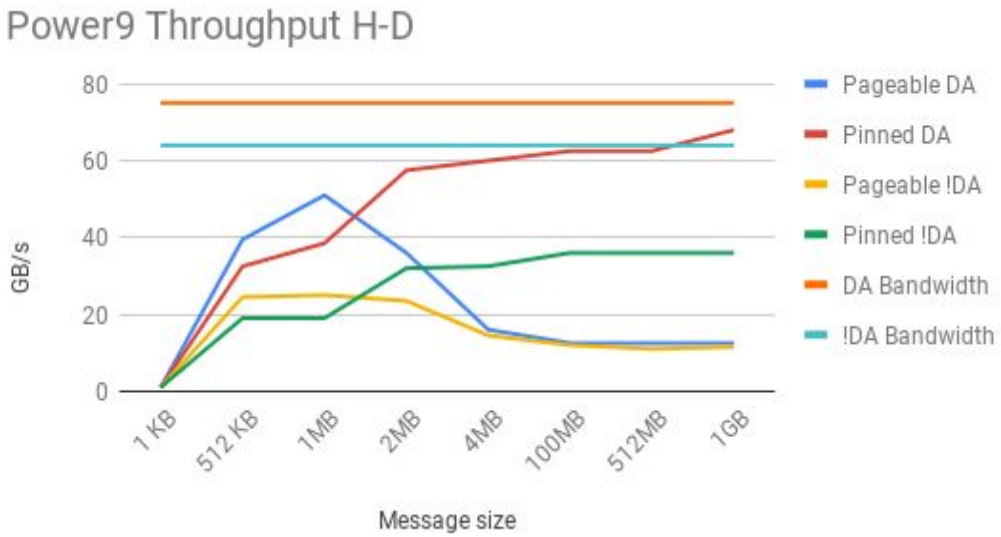
Speedup of **1,58x** (77x for the kernel)

Data transfers optimization

Pinned memory

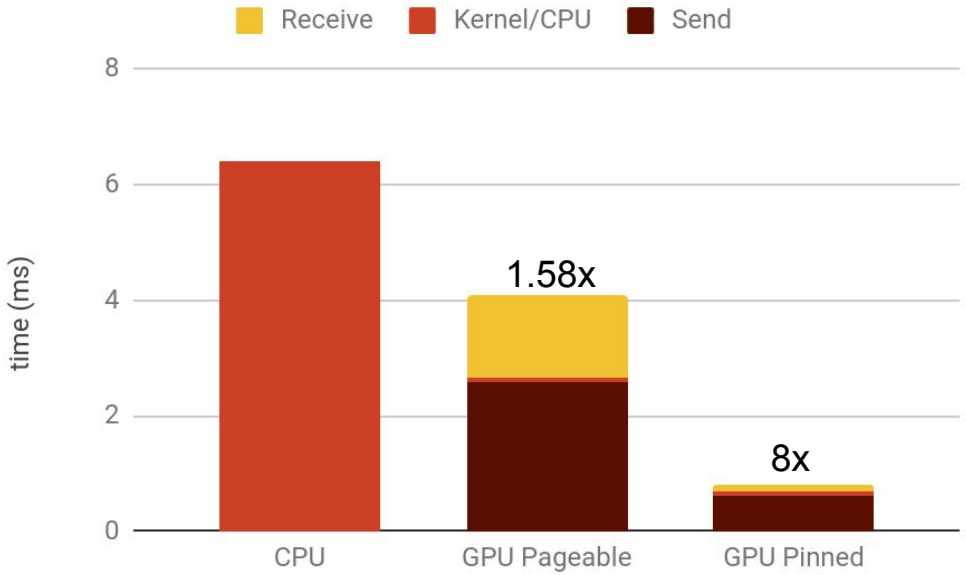
Worth after 1 MB

Use a GPU with a connection to the Power9 CPU



Power9 CPU-GPU throughput by message size

pinned dia_hsb kernel

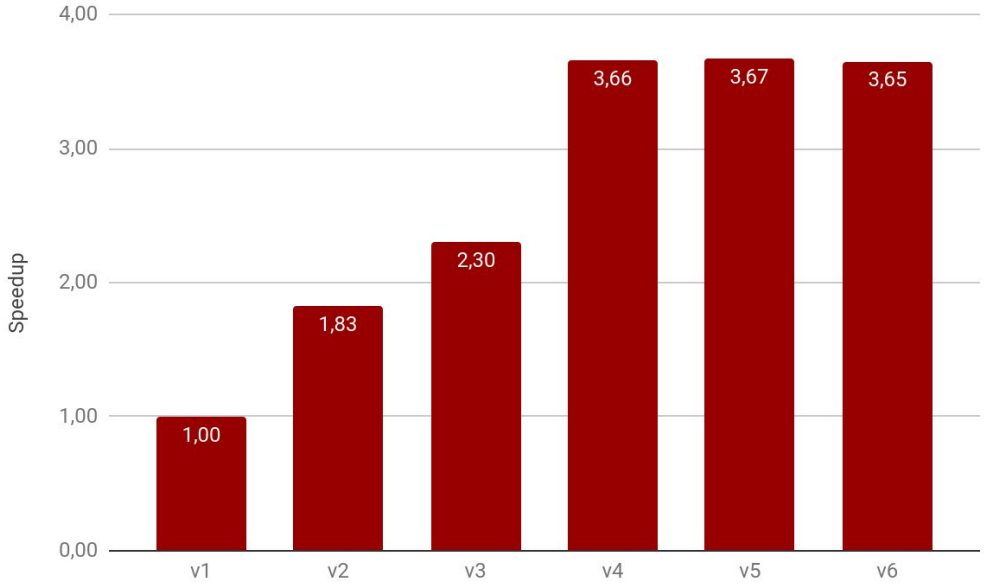


Absolute time (ms) CPU vs. Pageable vs. Pinned comparison

Pinned version is 5.1x faster than Pageable, achieving a total **speedup of 8** compared to the CPU

CUDA reduction optimization

- v2: solves warp divergence
- v3: global coalesced accesses and solves bank conflicts
- v4: half block size. First addition stored in shared memory
- v5-6: unrolling loops



GPU reduction speedup compared to the first version

CUDA reduction

- 87*105*75 = 685125

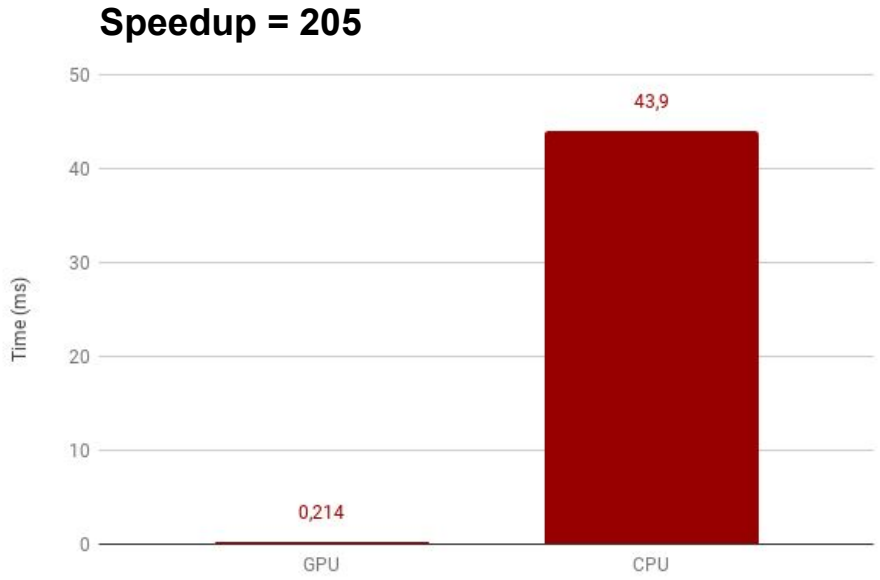
  Double floating point elements

- Knuth's trick

- 512 threads per block

- Data is already on the GPU

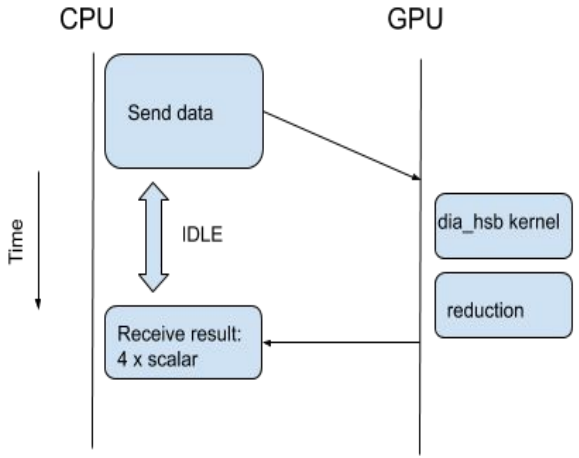
- Result is 4 scalars

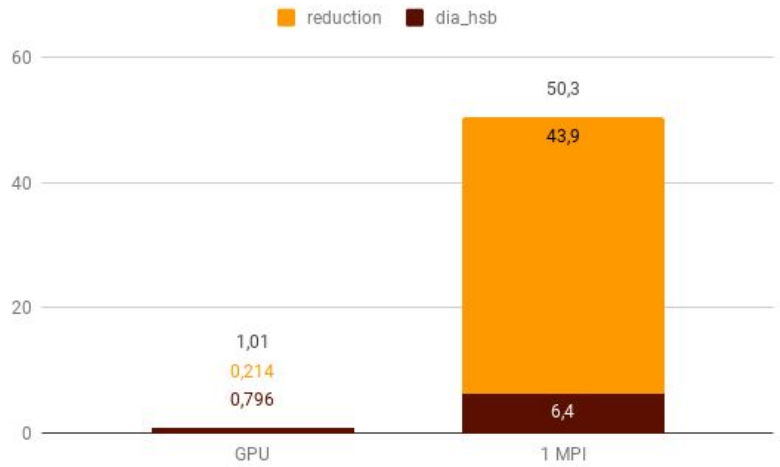


Absolut time (ms) GPU vs. CPU comparison

Results (single MPI process)

Control flow:





Absolute time (ms) GPU vs. CPU comparison

The speedup achieved is **50x** respect a single MPI process

Results

Comparisons vs. 1 MPI process:

- Fine when developing and optimizing the CUDA implementation
- Serial code
- Lack of a global view

Results

Comparisons vs. 1 MPI process:

- Fine when developing and optimizing the CUDA implementation
- Serial code
- Lack of a global view

Considerations:

- 40 cores and 4 GPUs per node
- MPI_Gather is expensive
- MPI processes have to share GPUs

Results

Comparisons vs. 1 MPI process:

- Fine when developing and optimizing the CUDA implementation
- Serial code
- Lack of a global view

Considerations:

- 40 CPUs and 4 GPUs per node
- MPI_Gather is expensive
- MPI processes have to share GPUs

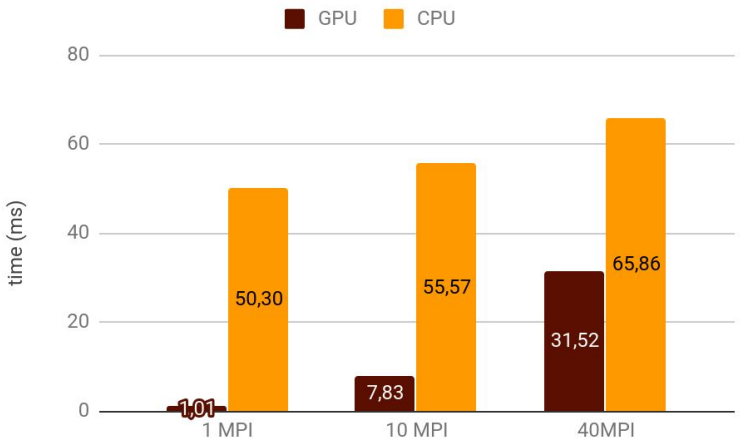
New results:

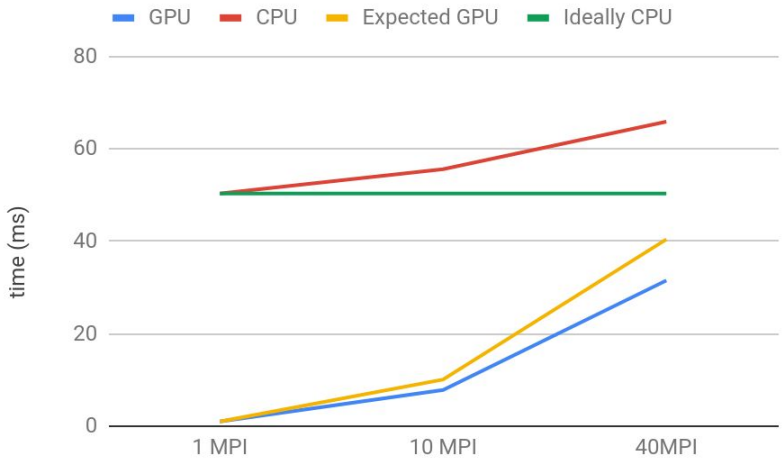
- 1 GPU per node
- 10 MPI processes per GPU

Results

CPU: Same work per core
GPU: takes the sum of all MPI processes

Weak scaling

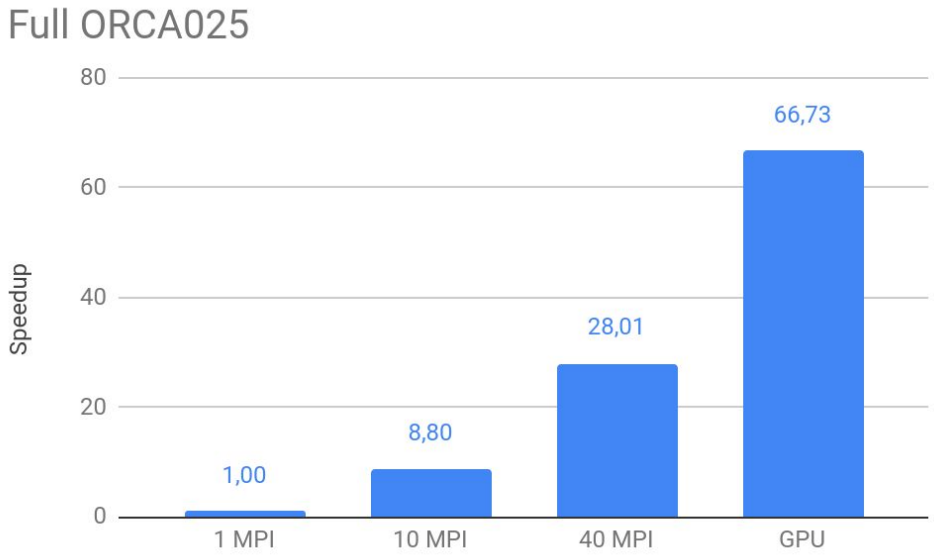




Weak scalability time comparison

Results

Strong scaling



dia_hsb Speedup respect full serial ORCA025

Conclusions

- Diagnostics take almost 15% of the total execution time
- They are not needed for the model but to prepare the output
- A GPU solution for dia_hsb has been implemented and tests are favorable
- Data transfers can bottleneck CUDA implementation
- Reductions take huge advantage of a GPU architecture
- Scalability tests show that the Kernel scales better than MPI

Future

- Integrate this work to NEMO
- Asynchronous GPU diagnostic execution
- Use this project as an example for the study of other diagnostics
- dia_ptr was not selected but also can be worth the effort
- Diagnostics executed as post-processing could also be taken into account
- Hybrid computation in the future (MareNostrum 5)

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Thank you

sergi.palomas@bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Accelerating Earth System Models

Portability of NEMO diagnostics to GPU

Sergi Palomas
Mario Acosta
Ramón Grau

Select a target diagnostic

GPU considerations:

- SIMD instructions
- Overhead init buffers and kernel
- Data transfers
- Throughput effectiveness with message data size

Meridional transports and zonal means (dia_ptr)

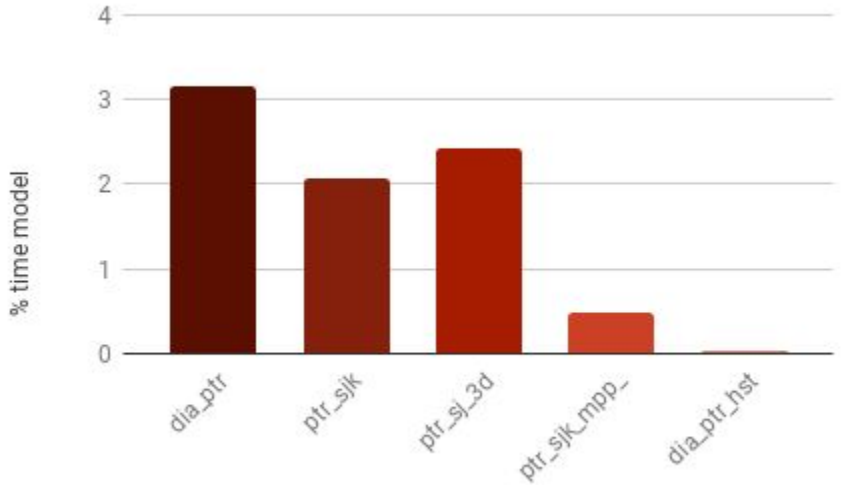
Poleward heat and salt transports

Advective and diffusive component

Called twice every ts

dia_ptr controlled by if-else

shared functions: sjk, sj_3D



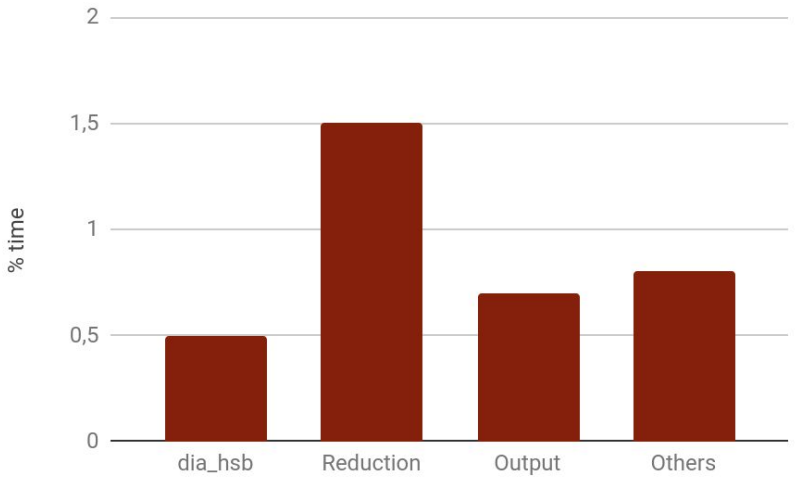
Percentual time per procedure in dia_ptr

Heat and salt budgets (dia_hsb)

At the end of every time-step

Global volume, volume variation, heat content and salt content deviation of the ocean

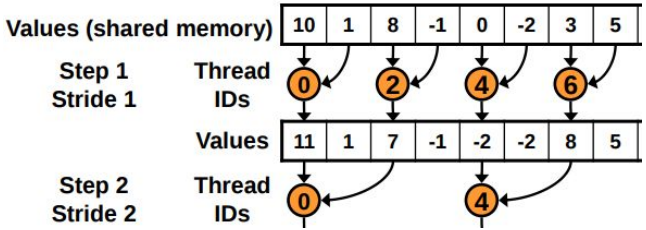
$$Volume\ variation = \int_{i,j,k}^{jpi,jpj,jpk} \big(m(i,j,k) \cdot s(i,j) - m\_ini(i,j,k) \cdot s\_ini(i,j)\big) \cdot mask(i,j,k)$$



Percentual time per procedure in dia_hsb

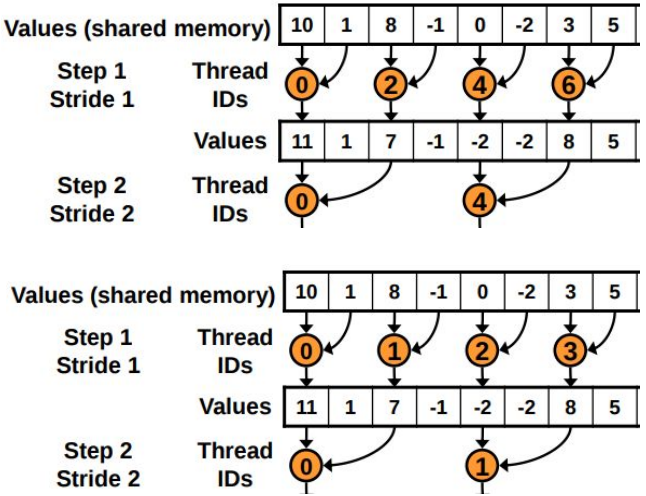
CUDA reduction v1

- Every CUDA block uses its own shared memory
- CUDA schedule warps
- Shared memory is 100x faster than Global
- Problem: Warp divergence serialize thread execution
- Problem: Memory coalescing (Global memory)
- Problem: Bank conflicts (Shared memory)



CUDA reduction v2

- Every CUDA block uses its own shared memory
- CUDA schedule warps
- Shared memory is 100x faster than Global
- ~~Problem: Warp divergence serialize thread execution~~
- Problem: Memory coalescing (Global memory)
- Problem: Bank conflicts (Shared memory)



CUDA reduction v3

- Every CUDA block uses its own shared memory
- CUDA schedule warps
- Shared memory is 100x faster than Global
- ~~Problem: Warp divergence serialize thread execution~~
- ~~Problem: Memory coalescing (Global memory)~~
- ~~Problem: Bank conflicts (Shared memory)~~