A TASK-BASED PARALLELIZATION OF A FINITE VOLUME CODE FOR THE EULER FLOWS

Dr. Sangeeth Simon
- Team CAGIRE, INRIA Bordeaux Sud-Ouest &
Laboratoire de Mathématiques et de leurs applications, Université de Pau et des Pays de l'Adour

December 8, 2020

What is this research about?

- **The overall aim:** To have a highly scalable, hardware-agnostic, Higher Order, Discontinuous Galerkin - based, Navier-Stokes solver.
- **The problem:** How to *easily* develop scalable and portable applications for heterogeneous HPC clusters?
- **Our approach:** Use a task-based parallelization paradigm to benefit from coexisting multi-core & GPU architectures.
- **Our findings so far:** Decent scalability obtained using StarPU on multi-core architectures.

What is this research about?

- **The overall aim:** To have a highly scalable, hardware-agnostic, Higher Order, Discontinuous Galerkin - based, Navier-Stokes solver.
- **The problem:** How to *easily* develop scalable and portable applications for heterogeneous HPC clusters?
- **Our approach:** Use a task-based parallelization paradigm to benefit from coexisting multi-core & GPU architectures.
- **Our findings so far:** Decent scalability obtained using StarPU on multi-core architectures.

What is this research about?

- **The overall aim:** To have a highly scalable, hardware-agnostic, Higher Order, Discontinuous Galerkin - based, Navier-Stokes solver.
- **The problem:** How to *easily* develop scalable and portable applications for heterogeneous HPC clusters?
- **Our approach:** Use a task-based parallelization paradigm to benefit from coexisting multi-core & GPU architectures.
- **Our findings so far:** Decent scalability obtained using StarPU on multi-core architectures.

What is this research about?

- **The overall aim:** To have a highly scalable, hardware-agnostic, Higher Order, Discontinuous Galerkin - based, Navier-Stokes solver.
- **The problem:** How to *easily* develop scalable and portable applications for heterogeneous HPC clusters?
- **Our approach:** Use a task-based parallelization paradigm to benefit from coexisting multi-core & GPU architectures.
- **Our findings so far:** Decent scalability obtained using StarPU on multi-core architectures.

What is this research about?

- **The overall aim:** To have a highly scalable, hardware-agnostic, Higher Order, Discontinuous Galerkin - based, Navier-Stokes solver.
- **The problem:** How to *easily* develop scalable and portable applications for heterogeneous HPC clusters?
- **Our approach:** Use a task-based parallelization paradigm to benefit from coexisting multi-core & GPU architectures.
- **Our findings so far:** Decent scalability obtained using StarPU on multi-core architectures.

Outline of the presentation

- Higher order methods for Nonlinear balance laws & Discontinuous Galerkin approach
- Task Based Parallelization & StarPU
- Important findings from Project HODINS so far
- Taking HODINS ahead→ Approach 1: Extensions to Three Dimensions
- Taking HODINS ahead→ Approach 2: Increasing order of accuracy
- Conclusions
- Future work

Outline of the presentation

- Higher order methods for Nonlinear balance laws & Discontinuous Galerkin approach
- Task Based Parallelization & StarPU
- Important findings from Project HODINS so far
- Taking HODINS ahead→ Approach 1: Extensions to Three Dimensions
- Taking HODINS ahead→ Approach 2: Increasing order of accuracy
- Conclusions
- Future work

Outline of the presentation

- Higher order methods for Nonlinear balance laws & Discontinuous Galerkin approach
- Task Based Parallelization & StarPU
- Important findings from Project HODINS so far
- Taking HODINS ahead→ Approach 1: Extensions to Three Dimensions
- Taking HODINS ahead→ Approach 2: Increasing order of accuracy
- Conclusions
- Future work

Outline of the presentation

- Higher order methods for Nonlinear balance laws & Discontinuous Galerkin approach
- Task Based Parallelization & StarPU
- Important findings from Project HODINS so far
- Taking HODINS ahead→ Approach 1: Extensions to Three Dimensions
- Taking HODINS ahead→ Approach 2: Increasing order of accuracy
- Conclusions
- Future work

Outline of the presentation

- Higher order methods for Nonlinear balance laws & Discontinuous Galerkin approach
- Task Based Parallelization & StarPU
- Important findings from Project HODINS so far
- Taking HODINS ahead→ Approach 1: Extensions to Three Dimensions
- Taking HODINS ahead→ Approach 2: Increasing order of accuracy
- Conclusions
- Future work

Outline of the presentation

- Higher order methods for Nonlinear balance laws & Discontinuous Galerkin approach
- Task Based Parallelization & StarPU
- Important findings from Project HODINS so far
- Taking HODINS ahead→ Approach 1: Extensions to Three Dimensions
- Taking HODINS ahead→ Approach 2: Increasing order of accuracy
- Conclusions
- Future work

Outline of the presentation

- Higher order methods for Nonlinear balance laws & Discontinuous Galerkin approach
- Task Based Parallelization & StarPU
- Important findings from Project HODINS so far
- Taking HODINS ahead→ Approach 1: Extensions to Three Dimensions
- Taking HODINS ahead→ Approach 2: Increasing order of accuracy
- Conclusions
- Future work

Higher Order Methods for Nonlinear Balance laws & Discontinuous Galerkin approach

Linear Balance laws

- Linear balance laws have **solutions that are superpositions** and hence comparitively easier to obtain.
- Computationally, they reduce to **global matrix operations** that can be optimized for an HPC application.

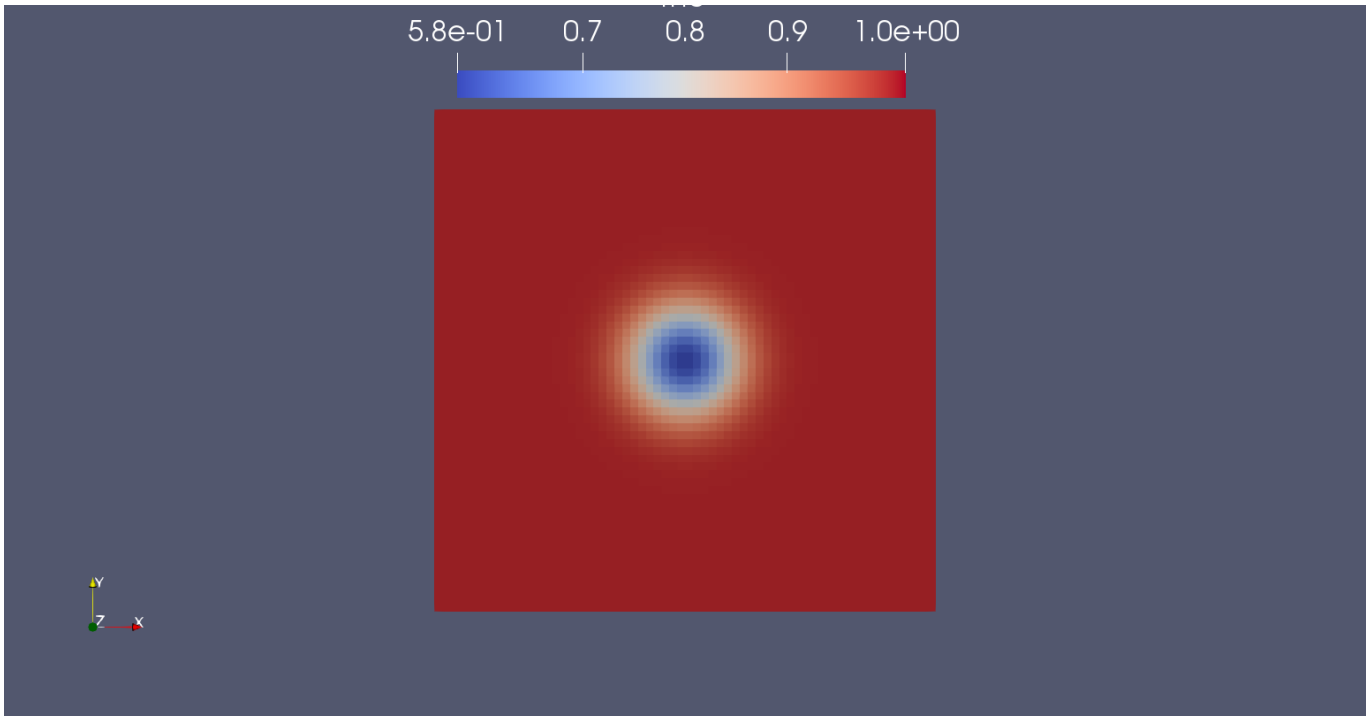
Higher Order Methods for Nonlinear Balance laws & Discontinuous Galerkin approach

Nonlinear Balance laws

- Nonlinear balance laws tend to **develop discontinuities** as part of their solution.
- Obtaining interface fluxes at the element boundaries involves **nonlinear function evaluations**.
- Relatively **higher computational complexity** per time step.

Higher Order Methods for Nonlinear Balance laws & Discontinuous Galerkin approach

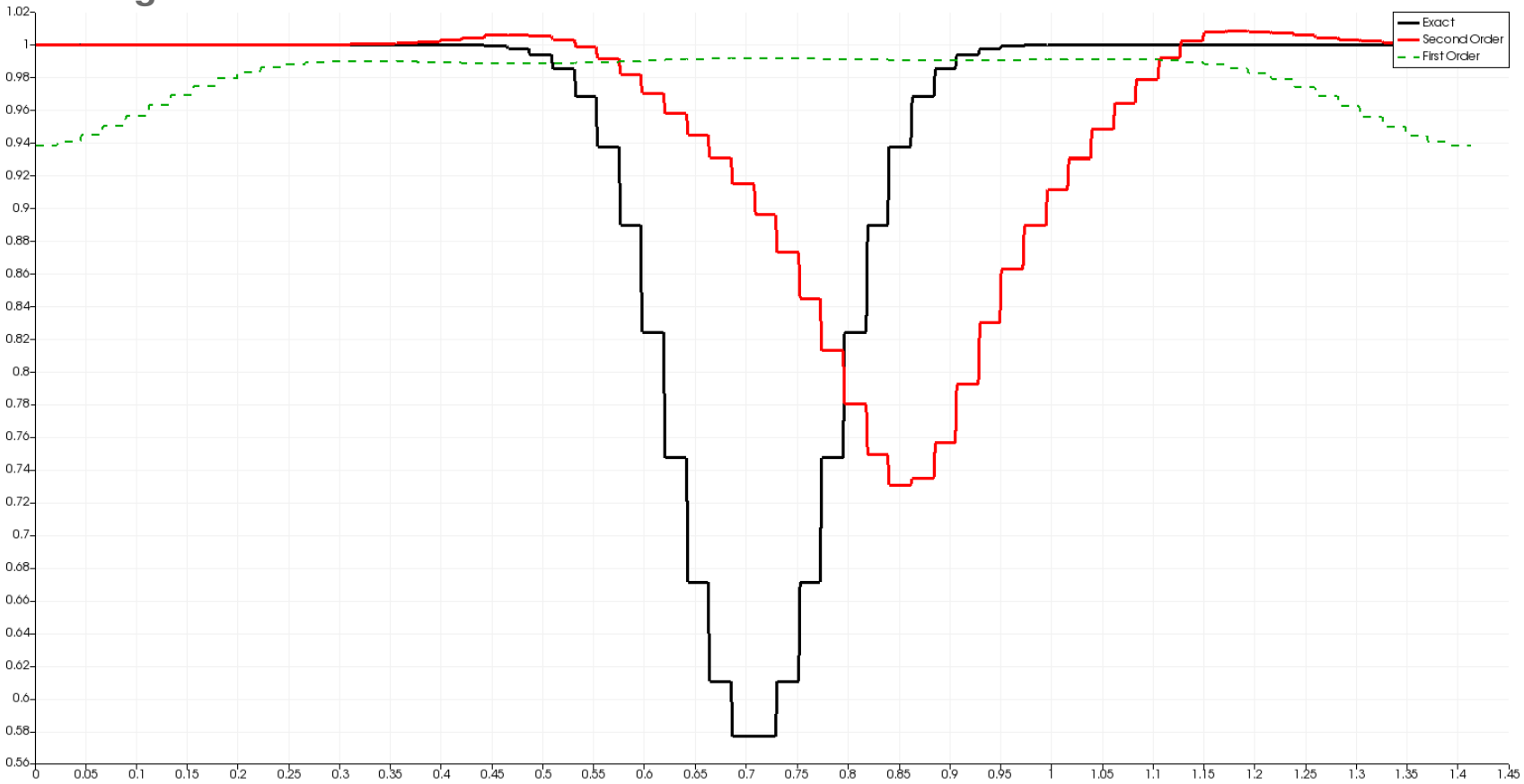
- **Why need higher order at all?** [1]



[1] Advection of a vortex on 64×64 grid

Higher Order Methods for Nonlinear Balance laws & Discontinuous Galerkin approach

- **Why need higher order at all?** [1]



First order is unacceptably more 'diffusive' than second order.

[1] Results for advection of a vortex on 64×64 grid for 5000 iterations. Second order accuracy using SSPRK-MUSCL.

Higher Order Methods for Nonlinear Balance laws & Discontinuous Galerkin approach

- **Why need higher order at all?**

Objective of any higher order method

Obtaining better resolution at cheaper cost compared to a lower order method!

Higher Order Methods for Nonlinear Balance laws & Discontinuous Galerkin approach

- **How does Discontinuous Galerkin approach help?**

Higher Order Methods for Nonlinear Balance laws & Discontinuous Galerkin approach

How does Discontinuous Galerkin approach help?

- **Best of both frameworks** → Finite Element (polynomial representation of solution on elements & matrix operations) & Finite Volume (conservation, interface fluxes, limiters).

Higher Order Methods for Nonlinear Balance laws & Discontinuous Galerkin approach

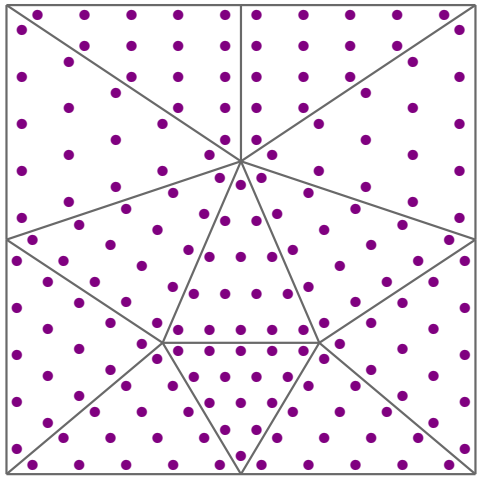
How does Discontinuous Galerkin approach help?

- **Good data locality & compact stencil** → Each element is equipped with enough data (except interface fluxes) to compute its own residuals. Each element interacts only with its immediate neighbours.

Higher Order Methods for Nonlinear Balance laws & Discontinuous Galerkin approach

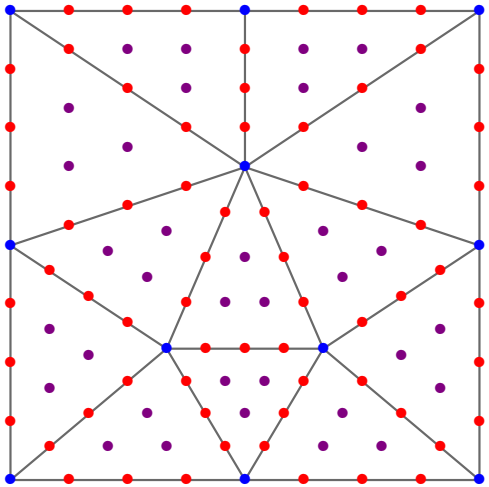
Elemental polynomial order k = 4

DG



pointers of size 12×15=180

CG



pointer of size 12×1=12

pointer of size 22×3=66

pointer of size 12×3=36

Higher Order Methods for Nonlinear Balance laws & Discontinuous Galerkin approach

How does Discontinuous Galerkin approach help?

- **Local matrices instead of global matrices** → In DG we completely avoid global matrix operations.
- **Diffusion operator adds more computations** → Evaluation of second order gradients using same DOF.
- **Enhanced arthimetic intensity** → Each element ideally does more computation with local data compared to communication.

Task-Based Parallelization & StarPU

- Heterogeneous clusters entail:
  1. Several multiprocessor architectures coexisting on a single node (CPU cores, specialized coprocessors, GPGPUs).
  2. Need for using different hardware specific API's to interact with them (CUDA, OpenMP, OpenACC, openCL).
  3. Developing new applications/ porting legacy codes etc. requires substantial effort.
  4. Poor portability across different clusters.
  5. Performance gains depend on how memory management is achieved.
  6. Domain experts often need to double-up as HPC experts.
- The ideal dream of today's CFD developer:
  1. A framework that enables swift HPC code development.
  2. Minimal memory management efforts.
  3. Offers good scalability with little effort.
  4. Same code runs on CPU/GPU/COProcessors depending on what is available.
- StarPU: A dynamic runtime scheduler supporting task based parallelism.
  1. Offers a unified execution model (unifies all computing units and their embedded memory).
  2. Automatically takes care of low-level memory management between host and accelerators.
  3. StarPU efficiently maps tasks to available computing units.
  4. Offers scope for development/tuning of scheduling policies.

Task-Based Parallelization & StarPU

- Heterogeneous clusters entail:
  1. Several multiprocessor architectures coexisting on a single node (CPU cores, specialized coprocessors, GPGPUs).
  2. Need for using different hardware specific API's to interact with them (CUDA, OpenMP, OpenACC, openCL).
  3. Developing new applications/ porting legacy codes etc. requires substantial effort.
  4. Poor portability across different clusters.
  5. Performance gains depend on how memory management is achieved.
  6. Domain experts often need to double-up as HPC experts.
- The ideal dream of today's CFD developer:
  1. A framework that enables swift HPC code development.
  2. Minimal memory management efforts.
  3. Offers good scalability with little effort.
  4. Same code runs on CPU/GPU/COProcessors depending on what is available.
- StarPU: A dynamic runtime scheduler supporting task based parallelism.
  1. Offers a unified execution model (unifies all computing units and their embedded memory).
  2. Automatically takes care of low-level memory management between host and accelerators.
  3. StarPU efficiently maps tasks to available computing units.
  4. Offers scope for development/tuning of scheduling policies.

Task-Based Parallelization & StarPU

- Heterogeneous clusters entail:
  1. Several multiprocessor architectures coexisting on a single node (CPU cores, specialized coprocessors, GPGPUs).
  2. Need for using different hardware specific API's to interact with them (CUDA, OpenMP, OpenACC, openCL).
  3. Developing new applications/ porting legacy codes etc. requires substantial effort.
  4. Poor portability across different clusters.
  5. Performance gains depend on how memory management is achieved.
  6. Domain experts often need to double-up as HPC experts.
- The ideal dream of today's CFD developer:
  1. A framework that enables swift HPC code development.
  2. Minimal memory management efforts.
  3. Offers good scalability with little effort.
  4. Same code runs on CPU/GPU/COProcessors depending on what is available.
- StarPU: A dynamic runtime scheduler supporting task based parallelism.
  1. Offers a unified execution model (unifies all computing units and their embedded memory).
  2. Automatically takes care of low-level memory management between host and accelerators.
  3. StarPU efficiently maps tasks to available computing units.
  4. Offers scope for development/tuning of scheduling policies.

Task-Based Parallelization & StarPU

- Heterogeneous clusters entail:
  1. Several multiprocessor architectures coexisting on a single node (CPU cores, specialized coprocessors, GPGPUs).
  2. Need for using different hardware specific API's to interact with them (CUDA, OpenMP, OpenACC, openCL).
  3. Developing new applications/ porting legacy codes etc. requires substantial effort.
  4. Poor portability across different clusters.
  5. Performance gains depend on how memory management is achieved.
  6. Domain experts often need to double-up as HPC experts.
- The ideal dream of today's CFD developer:
  1. A framework that enables swift HPC code development.
  2. Minimal memory management efforts.
  3. Offers good scalability with little effort.
  4. Same code runs on CPU/GPU/COProcessors depending on what is available.
- StarPU: A dynamic runtime scheduler supporting task based parallelism.
  1. Offers a unified execution model (unifies all computing units and their embedded memory).
  2. Automatically takes care of low-level memory management between host and accelerators.
  3. StarPU efficiently maps tasks to available computing units.
  4. Offers scope for development/tuning of scheduling policies.

Task-Based Parallelization & StarPU

- Heterogeneous clusters entail:
  1. Several multiprocessor architectures coexisting on a single node (CPU cores, specialized coprocessors, GPGPUs).
  2. Need for using different hardware specific API's to interact with them (CUDA, OpenMP, OpenACC, openCL).
  3. Developing new applications/ porting legacy codes etc. requires substantial effort.
  4. Poor portability across different clusters.
  5. Performance gains depend on how memory management is achieved.
  6. Domain experts often need to double-up as HPC experts.
- The ideal dream of today's CFD developer:
  1. A framework that enables swift HPC code development.
  2. Minimal memory management efforts.
  3. Offers good scalability with little effort.
  4. Same code runs on CPU/GPU/COProcessors depending on what is available.
- StarPU: A dynamic runtime scheduler supporting task based parallelism.
  1. Offers a unified execution model (unifies all computing units and their embedded memory).
  2. Automatically takes care of low-level memory management between host and accelerators.
  3. StarPU efficiently maps tasks to available computing units.
  4. Offers scope for development/tuning of scheduling policies.

Task-Based Parallelization & StarPU

- Heterogeneous clusters entail:
  1. Several multiprocessor architectures coexisting on a single node (CPU cores, specialized coprocessors, GPGPUs).
  2. Need for using different hardware specific API's to interact with them (CUDA, OpenMP, OpenACC, openCL).
  3. Developing new applications/ porting legacy codes etc. requires substantial effort.
  4. Poor portability across different clusters.
  5. Performance gains depend on how memory management is achieved.
  6. Domain experts often need to double-up as HPC experts.
- The ideal dream of today's CFD developer:
  1. A framework that enables swift HPC code development.
  2. Minimal memory management efforts.
  3. Offers good scalability with little effort.
  4. Same code runs on CPU/GPU/COProcessors depending on what is available.
- StarPU: A dynamic runtime scheduler supporting task based parallelism.
  1. Offers a unified execution model (unifies all computing units and their embedded memory).
  2. Automatically takes care of low-level memory management between host and accelerators.
  3. StarPU efficiently maps tasks to available computing units.
  4. Offers scope for development/tuning of scheduling policies.

Task-Based Parallelization & StarPU

- Heterogeneous clusters entail:
  1. Several multiprocessor architectures coexisting on a single node (CPU cores, specialized coprocessors, GPGPUs).
  2. Need for using different hardware specific API's to interact with them (CUDA, OpenMP, OpenACC, openCL).
  3. Developing new applications/ porting legacy codes etc. requires substantial effort.
  4. Poor portability across different clusters.
  5. Performance gains depend on how memory management is achieved.
  6. Domain experts often need to double-up as HPC experts.
- The ideal dream of today's CFD developer:
  1. A framework that enables swift HPC code development.
  2. Minimal memory management efforts.
  3. Offers good scalability with little effort.
  4. Same code runs on CPU/GPU/COProcessors depending on what is available.
- StarPU: A dynamic runtime scheduler supporting task based parallelism.
  1. Offers a unified execution model (unifies all computing units and their embedded memory).
  2. Automatically takes care of low-level memory management between host and accelerators.
  3. StarPU efficiently maps tasks to available computing units.
  4. Offers scope for development/tuning of scheduling policies.

Task-Based Parallelization & StarPU

- Heterogeneous clusters entail:
  1. Several multiprocessor architectures coexisting on a single node (CPU cores, specialized coprocessors, GPGPUs).
  2. Need for using different hardware specific API's to interact with them (CUDA, OpenMP, OpenACC, openCL).
  3. Developing new applications/ porting legacy codes etc. requires substantial effort.
  4. Poor portability across different clusters.
  5. Performance gains depend on how memory management is achieved.
  6. Domain experts often need to double-up as HPC experts.
- The ideal dream of today's CFD developer:
  1. A framework that enables swift HPC code development.
  2. Minimal memory management efforts.
  3. Offers good scalability with little effort.
  4. Same code runs on CPU/GPU/COProcessors depending on what is available.
- StarPU: A dynamic runtime scheduler supporting task based parallelism.
  1. Offers a unified execution model (unifies all computing units and their embedded memory).
  2. Automatically takes care of low-level memory management between host and accelerators.
  3. StarPU efficiently maps tasks to available computing units.
  4. Offers scope for development/tuning of scheduling policies.

Task-Based Parallelization & StarPU

- Heterogeneous clusters entail:
  1. Several multiprocessor architectures coexisting on a single node (CPU cores, specialized coprocessors, GPGPUs).
  2. Need for using different hardware specific API's to interact with them (CUDA, OpenMP, OpenACC, openCL).
  3. Developing new applications/ porting legacy codes etc. requires substantial effort.
  4. Poor portability across different clusters.
  5. Performance gains depend on how memory management is achieved.
  6. Domain experts often need to double-up as HPC experts.
- The ideal dream of today's CFD developer:
  1. A framework that enables swift HPC code development.
  2. Minimal memory management efforts.
  3. Offers good scalability with little effort.
  4. Same code runs on CPU/GPU/COProcessors depending on what is available.
- StarPU: A dynamic runtime scheduler supporting task based parallelism.
  1. Offers a unified execution model (unifies all computing units and their embedded memory).
  2. Automatically takes care of low-level memory management between host and accelerators.
  3. StarPU efficiently maps tasks to available computing units.
  4. Offers scope for development/tuning of scheduling policies.

Task-Based Parallelization & StarPU

- Heterogeneous clusters entail:
  1. Several multiprocessor architectures coexisting on a single node (CPU cores, specialized coprocessors, GPGPUs).
  2. Need for using different hardware specific API's to interact with them (CUDA, OpenMP, OpenACC, openCL).
  3. Developing new applications/ porting legacy codes etc. requires substantial effort.
  4. Poor portability across different clusters.
  5. Performance gains depend on how memory management is achieved.
  6. Domain experts often need to double-up as HPC experts.
- The ideal dream of today's CFD developer:
  1. A framework that enables swift HPC code development.
  2. Minimal memory management efforts.
  3. Offers good scalability with little effort.
  4. Same code runs on CPU/GPU/COProcessors depending on what is available.
- StarPU: A dynamic runtime scheduler supporting task based parallelism.
  1. Offers a unified execution model (unifies all computing units and their embedded memory).
  2. Automatically takes care of low-level memory management between host and accelerators.
  3. StarPU efficiently maps tasks to available computing units.
  4. Offers scope for development/tuning of scheduling policies.

Task-Based Parallelization & StarPU

- Heterogeneous clusters entail:
  1. Several multiprocessor architectures coexisting on a single node (CPU cores, specialized coprocessors, GPGPUs).
  2. Need for using different hardware specific API's to interact with them (CUDA, OpenMP, OpenACC, openCL).
  3. Developing new applications/ porting legacy codes etc. requires substantial effort.
  4. Poor portability across different clusters.
  5. Performance gains depend on how memory management is achieved.
  6. Domain experts often need to double-up as HPC experts.
- The ideal dream of today's CFD developer:
  1. A framework that enables swift HPC code development.
  2. Minimal memory management efforts.
  3. Offers good scalability with little effort.
  4. Same code runs on CPU/GPU/COProcessors depending on what is available.
- StarPU: A dynamic runtime scheduler supporting task based parallelism.
  1. Offers a unified execution model (unifies all computing units and their embedded memory).
  2. Automatically takes care of low-level memory management between host and accelerators.
  3. StarPU efficiently maps tasks to available computing units.
  4. Offers scope for development/tuning of scheduling policies.

Task-Based Parallelization & StarPU

- Heterogeneous clusters entail:
  1. Several multiprocessor architectures coexisting on a single node (CPU cores, specialized coprocessors, GPGPUs).
  2. Need for using different hardware specific API's to interact with them (CUDA, OpenMP, OpenACC, openCL).
  3. Developing new applications/ porting legacy codes etc. requires substantial effort.
  4. Poor portability across different clusters.
  5. Performance gains depend on how memory management is achieved.
  6. Domain experts often need to double-up as HPC experts.
- The ideal dream of today's CFD developer:
  1. A framework that enables swift HPC code development.
  2. Minimal memory management efforts.
  3. Offers good scalability with little effort.
  4. Same code runs on CPU/GPU/COProcessors depending on what is available.
- StarPU: A dynamic runtime scheduler supporting task based parallelism.
  1. Offers a unified execution model (unifies all computing units and their embedded memory).
  2. Automatically takes care of low-level memory management between host and accelerators.
  3. StarPU efficiently maps tasks to available computing units.
  4. Offers scope for development/tuning of scheduling policies.

Task-Based Parallelization & StarPU

- Heterogeneous clusters entail:
  1. Several multiprocessor architectures coexisting on a single node (CPU cores, specialized coprocessors, GPGPUs).
  2. Need for using different hardware specific API's to interact with them (CUDA, OpenMP, OpenACC, openCL).
  3. Developing new applications/ porting legacy codes etc. requires substantial effort.
  4. Poor portability across different clusters.
  5. Performance gains depend on how memory management is achieved.
  6. Domain experts often need to double-up as HPC experts.
- The ideal dream of today's CFD developer:
  1. A framework that enables swift HPC code development.
  2. Minimal memory management efforts.
  3. Offers good scalability with little effort.
  4. Same code runs on CPU/GPU/COProcessors depending on what is available.
- StarPU: A dynamic runtime scheduler supporting task based parallelism.
  1. Offers a unified execution model (unifies all computing units and their embedded memory).
  2. Automatically takes care of low-level memory management between host and accelerators.
  3. StarPU efficiently maps tasks to available computing units.
  4. Offers scope for development/tuning of scheduling policies.

Task-Based Parallelization & StarPU

- Heterogeneous clusters entail:
  1. Several multiprocessor architectures coexisting on a single node (CPU cores, specialized coprocessors, GPGPUs).
  2. Need for using different hardware specific API's to interact with them (CUDA, OpenMP, OpenACC, openCL).
  3. Developing new applications/ porting legacy codes etc. requires substantial effort.
  4. Poor portability across different clusters.
  5. Performance gains depend on how memory management is achieved.
  6. Domain experts often need to double-up as HPC experts.
- The ideal dream of today's CFD developer:
  1. A framework that enables swift HPC code development.
  2. Minimal memory management efforts.
  3. Offers good scalability with little effort.
  4. Same code runs on CPU/GPU/COProcessors depending on what is available.
- StarPU: A dynamic runtime scheduler supporting task based parallelism.
  1. Offers a unified execution model (unifies all computing units and their embedded memory).
  2. Automatically takes care of low-level memory management between host and accelerators.
  3. StarPU efficiently maps tasks to available computing units.
  4. Offers scope for development/tuning of scheduling policies.

Task-Based Parallelization & StarPU

- Heterogeneous clusters entail:
  1. Several multiprocessor architectures coexisting on a single node (CPU cores, specialized coprocessors, GPGPUs).
  2. Need for using different hardware specific API's to interact with them (CUDA, OpenMP, OpenACC, openCL).
  3. Developing new applications/ porting legacy codes etc. requires substantial effort.
  4. Poor portability across different clusters.
  5. Performance gains depend on how memory management is achieved.
  6. Domain experts often need to double-up as HPC experts.
- The ideal dream of today's CFD developer:
  1. A framework that enables swift HPC code development.
  2. Minimal memory management efforts.
  3. Offers good scalability with little effort.
  4. Same code runs on CPU/GPU/COProcessors depending on what is available.
- StarPU: A dynamic runtime scheduler supporting task based parallelism.
  1. Offers a unified execution model (unifies all computing units and their embedded memory).
  2. Automatically takes care of low-level memory management between host and accelerators.
  3. StarPU efficiently maps tasks to available computing units.
  4. Offers scope for development/tuning of scheduling policies.

Task-Based Parallelization & StarPU

- Heterogeneous clusters entail:
  1. Several multiprocessor architectures coexisting on a single node (CPU cores, specialized coprocessors, GPGPUs).
  2. Need for using different hardware specific API's to interact with them (CUDA, OpenMP, OpenACC, openCL).
  3. Developing new applications/ porting legacy codes etc. requires substantial effort.
  4. Poor portability across different clusters.
  5. Performance gains depend on how memory management is achieved.
  6. Domain experts often need to double-up as HPC experts.
- The ideal dream of today's CFD developer:
  1. A framework that enables swift HPC code development.
  2. Minimal memory management efforts.
  3. Offers good scalability with little effort.
  4. Same code runs on CPU/GPU/COProcessors depending on what is available.
- StarPU: A dynamic runtime scheduler supporting task based parallelism.
  1. Offers a unified execution model (unifies all computing units and their embedded memory).
  2. Automatically takes care of low-level memory management between host and accelerators.
  3. StarPU efficiently maps tasks to available computing units.
  4. Offers scope for development/tuning of scheduling policies.

HODINS-2D

- HODINS→ **H**igh **O**rder **DI**scontinuous methods with ru**N**time **S**cheduler.

HODINS-2D

- HODINS-2D → a first order accurate Finite Volume code for the Euler equations with source terms using StarPU. [1]

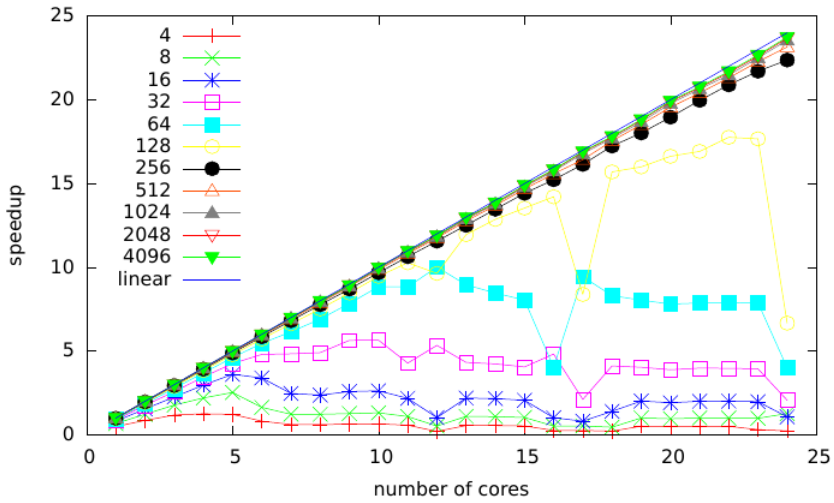
[1] Essadki, M. Jung, J. Larat, A. Pelletier, M. Perrier, V. A Task-Driven Implementation of a Simple Numerical Solver for Hyperbolic Conservation Laws. *ESAIM: ProcS* (2018) **63**:228–247.

HODINS-2D

- Experiments performed on a 2 Dodeca-core Haswell Intel Xeon E5-2680, a Xeon Phi KNL and Nvidia K40-M GPU.

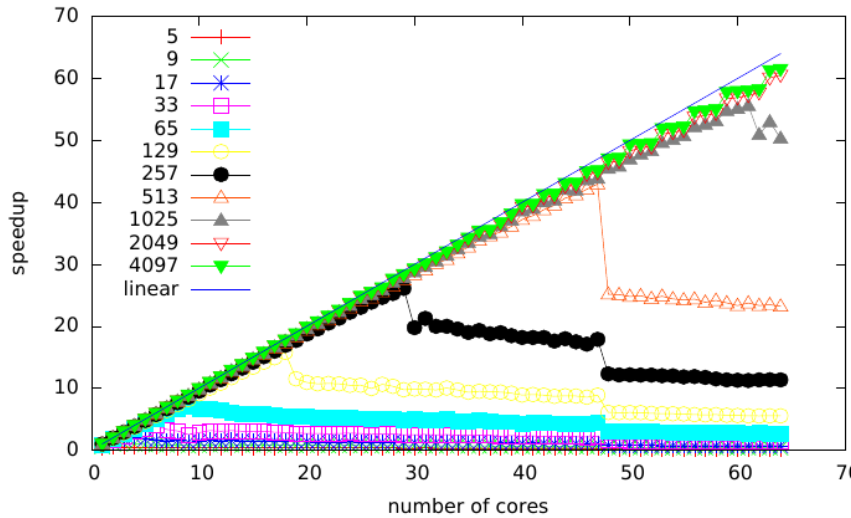
HODINS-2D

Importance of tasksizeOverhead [1]



(A) 2 dodeca-core Haswell Intel Xeon E5-2680.

(a) Haswell



(B) Xeon Phi KNL.

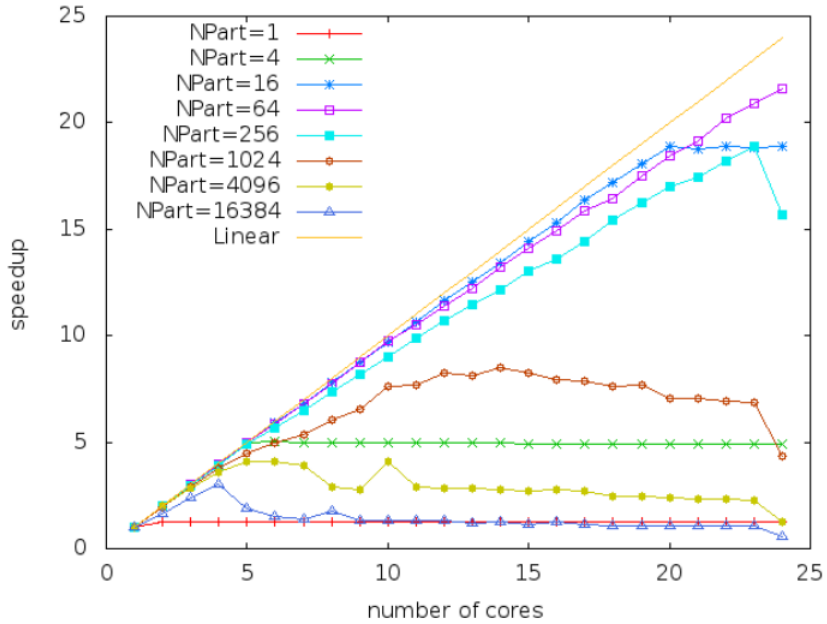
(b) KNL

Tasks must have a minimum execution time (~1ms) for scalability from StarPU!

[1] Taken from Essadki et al, 2018. Legends are in (μ s)

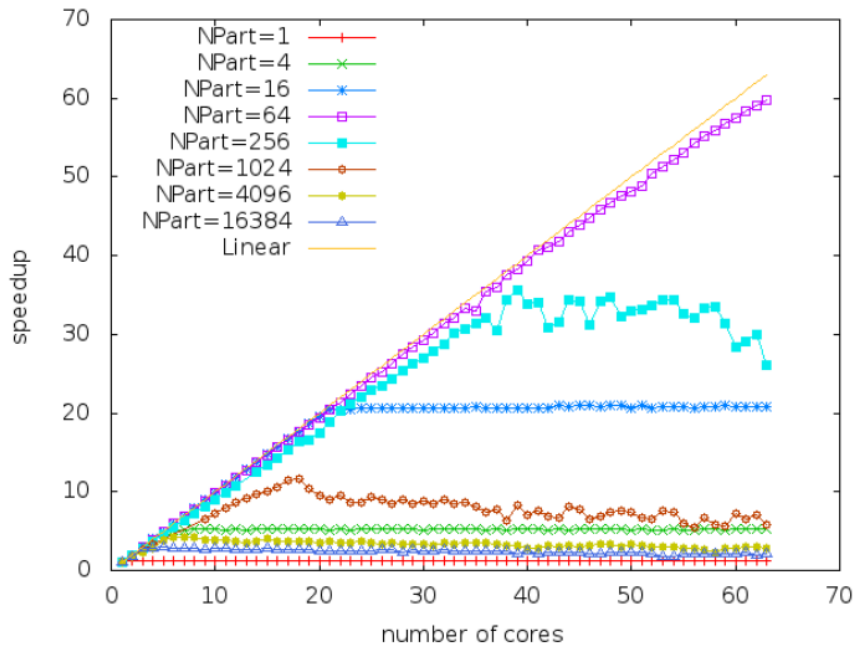
HODINS-2D

Scalability results[1]



(A) 2 dodeca-core Haswell Intel Xeon E5-2680.

(a) Haswell



(B) Xeon Phi KNL.

(b) KNL

For scalability from StarPU: Numerous tasks ($\mathcal{O}$(No of Cores)) that are individually large enough (>tasksizeoverhead)

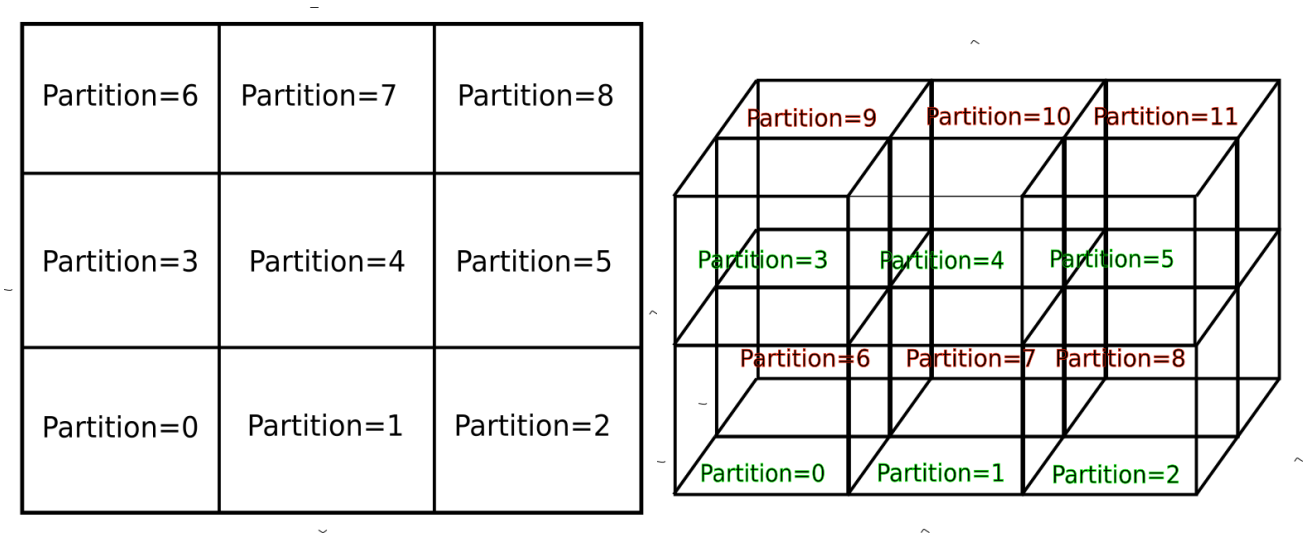
[1] Taken from Essadki et al, 2018. Results for 1024×1024 grid.

HODINS-2D

- How do we improve arithmetic intensity of kernels?

HODINS-3D

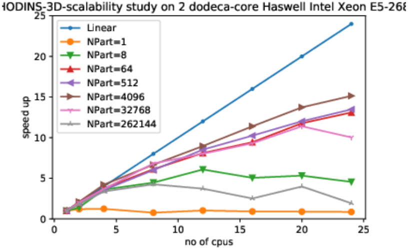
- Extend HODINS-2D to HODINS-3D.

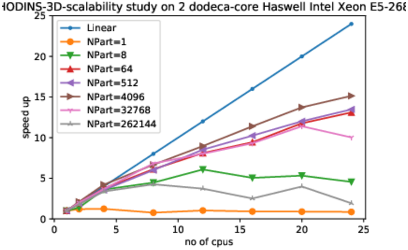


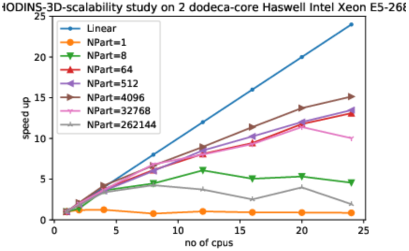
(a) Haswell (b) KNL

HODINS-3D

Scalability results[1]







- Same pattern as 2D → Too many/Too few tasks not good for scalability.
- NParts = 64, 512 and 4096 offers some scaling.
- HODINS-3D scalability → Not as great as reported for HODINS-2D.
- Need to investigate the reason for poor scaling.

[1] Results for 512^3 grid.

HODINS-Higher order

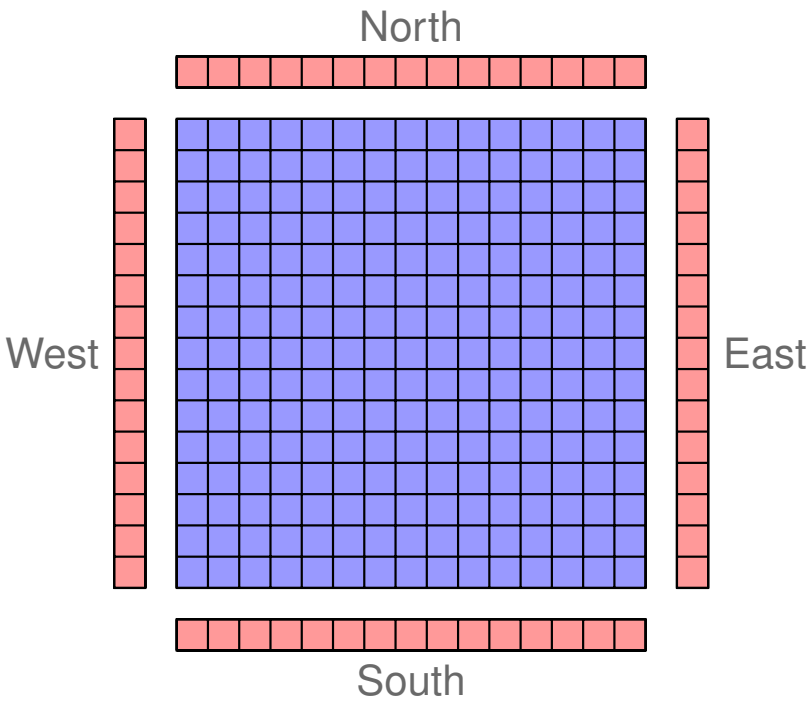
- Higher order increases arthmetic intensity per kernel due to additional operations like gradient computation and reconstruction.

HODINS-Higher order

- Higher order achieved using SSPRK-MUSCL combination.

HODINS-Higher order

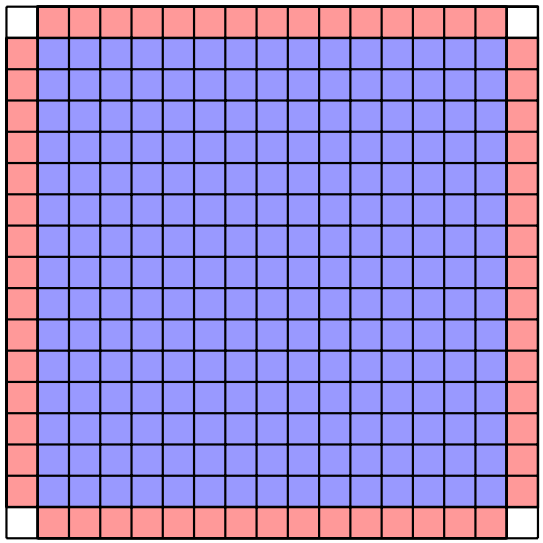
Old overlaps model



- 5 pointers
- 5 `starpu_data_handle`
- 1st step: launch concurrently
  - Internal residual computation[R]
  - copy inside overlaps[W]
- 2nd step: compute the border residual, involving both internal[R] and borders[R] data handles
  - But matching indices between internal and overlaps is not straightforward to implement

HODINS-Higher order

New overlaps model



- 1 pointers
- 6 `starpu_data_handle`
  - One view with Internal and overlaps
  - One view with full pointer
- 1st step: launch concurrently
  - Internal residual computation
  - copy inside overlaps
- 2nd step: compute the border residual, involving the full view of the pointer
  - Use the same algorithm as for internal residual, but with different indices.

What lies ahead...

- Complete H.O implementation and do performance analyses (Gantt charts, GFLOPS, Memory bandwidth, time taken by codelets etc.).
- Assimilate learning to develop HODINS-MUSCL-3D. Performance analyses.
- Apply learning to DG through AEROSOL framework.

What lies ahead...

- Complete H.O implementation and do performance analyses (Gantt charts, GFLOPS, Memory bandwidth, time taken by codelets etc.).
- Assimilate learning to develop HODINS-MUSCL-3D. Performance analyses.
- Apply learning to DG through AEROSOL framework.

What lies ahead...

- Complete H.O implementation and do performance analyses (Gantt charts, GFLOPS, Memory bandwidth, time taken by codelets etc.).
- Assimilate learning to develop HODINS-MUSCL-3D. Performance analyses.
- Apply learning to DG through AEROSOL framework.

HODINS-Higher order

- We motivated the **need for Higher order methods for Non-Linear equations** (like Navier-Stokes eqns).
- We motivated why **Discontinuous Galerkin might be an ideal candidate** for achieving this.
- We motivated the **need for using task-based parallelism** in a world of hetergeneous architectures.
- We reviewed the work done under HODINS-2D on a first order Euler code with source terms: We realized that **saturating task heap with compute intense tasks is necessary for achieving scalability**.
- To improve compute intensity, we built HODINS-3D: **Decent scalability observed** for Nparts=64 for a grid of 512^3 cartesian cells on different architectures. Scope for improvement exist.
- Also **built HODINS-MUSCL-2D with new data management features**.

HODINS-Higher order

- We motivated the **need for Higher order methods for Non-Linear equations** (like Navier-Stokes eqns).
- We motivated why **Discontinuous Galerkin might be an ideal candidate** for achieving this.
- We motivated the **need for using task-based parallelism** in a world of hetergeneous architectures.
- We reviewed the work done under HODINS-2D on a first order Euler code with source terms: We realized that **saturating task heap with compute intense tasks is necessary for achieving scalability**.
- To improve compute intensity, we built HODINS-3D: **Decent scalability observed** for Nparts=64 for a grid of 512^3 cartesian cells on different architectures. Scope for improvement exist.
- Also **built HODINS-MUSCL-2D with new data management features**.

HODINS-Higher order

- We motivated the **need for Higher order methods for Non-Linear equations** (like Navier-Stokes eqns).
- We motivated why **Discontinuous Galerkin might be an ideal candidate** for achieving this.
- We motivated the **need for using task-based parallelism** in a world of hetergeneous architectures.
- We reviewed the work done under HODINS-2D on a first order Euler code with source terms: We realized that **saturating task heap with compute intense tasks is necessary for achieving scalability**.
- To improve compute intensity, we built HODINS-3D: **Decent scalability observed** for Nparts=64 for a grid of 512^3 cartesian cells on different architectures. Scope for improvement exist.
- Also **built HODINS-MUSCL-2D with new data management features**.

HODINS-Higher order

- We motivated the **need for Higher order methods for Non-Linear equations** (like Navier-Stokes eqns).
- We motivated why **Discontinuous Galerkin might be an ideal candidate** for achieving this.
- We motivated the **need for using task-based parallelism** in a world of hetergeneous architectures.
- We reviewed the work done under HODINS-2D on a first order Euler code with source terms: We realized that **saturating task heap with compute intense tasks is necessary for achieving scalability**.
- To improve compute intensity, we built HODINS-3D: **Decent scalability observed** for Nparts=64 for a grid of 512^3 cartesian cells on different architectures. Scope for improvement exist.
- Also **built HODINS-MUSCL-2D with new data management features**.

HODINS-Higher order

- We motivated the **need for Higher order methods for Non-Linear equations** (like Navier-Stokes eqns).
- We motivated why **Discontinuous Galerkin might be an ideal candidate** for achieving this.
- We motivated the **need for using task-based parallelism** in a world of hetergeneous architectures.
- We reviewed the work done under HODINS-2D on a first order Euler code with source terms: We realized that **saturating task heap with compute intense tasks is necessary for achieving scalability**.
- To improve compute intensity, we built HODINS-3D: **Decent scalability observed** for Nparts=64 for a grid of 512^3 cartesian cells on different architectures. Scope for improvement exist.
- Also **built HODINS-MUSCL-2D with new data management features.**

HODINS-Higher order

- We motivated the **need for Higher order methods for Non-Linear equations** (like Navier-Stokes eqns).
- We motivated why **Discontinuous Galerkin might be an ideal candidate** for achieving this.
- We motivated the **need for using task-based parallelism** in a world of hetergeneous architectures.
- We reviewed the work done under HODINS-2D on a first order Euler code with source terms: We realized that **saturating task heap with compute intense tasks is necessary for achieving scalability**.
- To improve compute intensity, we built HODINS-3D: **Decent scalability observed** for Nparts=64 for a grid of 512^3 cartesian cells on different architectures. Scope for improvement exist.
- Also **built HODINS-MUSCL-2D with new data management features**.

My sincere thanks to:

(1cm) **Inria & "Conseil Régional Nouvelle Aquitaine".**
Team CAGIRE (Vincent Perrier, Matthieu Hafele, Jonathan Jung)
ANR Solharis / Région Nouvelle Aquitaine hpc-scalable-ecosystem Organizers