Parallel strategy and roofline analysis for a 3D code modelling edge plasma physics

M. Kuhn, G. Latu, N. Crouseilles, S. Genaud
matthieu.kuhn@inria.fr

April 6, 2016

Context

My PhD

- *Parallel computing and numerical methods for plasma boundary simulations*

Goal

- Reduce execution times of an existing code Emedge3D
  - simulating plasma behavior at the edges of tokamaks

Challenges

- Very large problems at both spatial and temporal scales:
  - $T_{\text{max}} >> 1$ and $\Delta t << 1$ (stability condition)
  - $N_d >> 1$ and $\Delta d << 1$, $d \in \{x, y, z\}$

Solutions

- Semi-implicit time integration to $\nearrow \Delta t$
- HPC to reach high resolutions in space

HPC for Emedge3D

Goal

- Goal: give a viable strategy for hybrid MPI/OpenMP parallelization in Emedge3D

Setup

- Study on a (3D) reduced model of Emedge3D, containing the two most challenging and time consuming spatial operators
  - the 3D diffusion $\nabla \cdot (A(x)\nabla .)$, $\nabla = (\partial_x, \partial_y, \partial_z)$
  - the 2D advection $\{f, g\} = \partial_x f \partial_y g - \partial_y f \partial_x g$

Outline

Outline

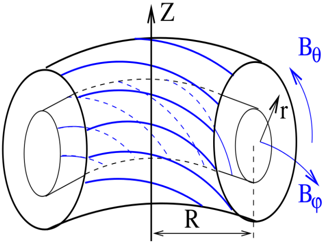
1. The advection-diffusion equation and numerical settings

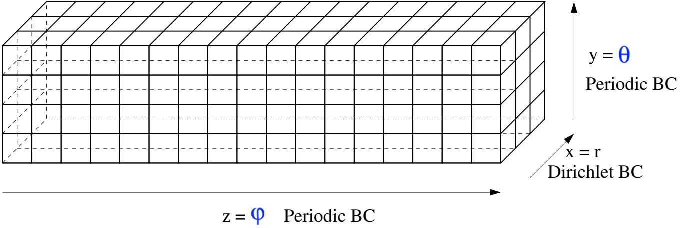
2. Parallelization strategy and implementation

3. Performance analysis
   - OpenMP only and roofline analysis
   - Hybrid MPI/OpenMP

4. Conclusion and perspectives

SLAB geometry





Equation: reduced model of Emedge3D

$$\partial_t T + \{\phi, T\} + Q_{adv} = \nabla \cdot (A(x) \nabla T) + Q_{diff},$$

with

- $T = T(t, x, y, z)$ the unknown
- 2D advection: $\{f, g\} = \partial_x f \partial_y g - \partial_y f \partial_x g$
- 3D diffusion: 3×3 matrix $A(x)$, $\nabla = (\partial_x, \partial_y, \partial_z)$
- $\phi = \phi(x, y, z)$ the electric potential (constant in time)
- $Q_* = Q_*(t, x, y, z), * \in \{adv, diff\}$ the source terms
- $+$ analytical test case

Discretization

- As in Emedge3D, two discretizations in space:
  - real (x) and Fourier basis (y,z) for the diffusion $\nabla \cdot (A(x)\nabla .)$
    - $\rightarrow$ finite difference order 2 (x) and spectral differentiation (y,z)
  - full real basis for the advection $\{f, g\} = \partial_x f \partial_y g - \partial_y f \partial_x g$
    - $\rightarrow$ Arakawa method order 2
    - $\rightarrow$ to avoid a convolution ($\mathcal{O}(N_x \times N_y^2 \times N_z^2)$)

$\Rightarrow$ 2D DFT between operators

Discretization

- As in Emedge3D, two discretizations in space:
  - real (x) and Fourier basis (y,z) for the diffusion $\nabla \cdot (A(x)\nabla .)$
    - $\rightarrow$ finite difference order 2 (x) and spectral differentiation (y,z)
  - full real basis for the advection $\{f, g\} = \partial_x f \partial_y g - \partial_y f \partial_x g$
    - $\rightarrow$ Arakawa method order 2
    - $\rightarrow$ to avoid a convolution $(\mathcal{O}(N_x \times N_y^2 \times N_z^2))$

  $\Rightarrow$ 2D DFT between operators

- Explicit discretization in time with Lie splitting:
  - with $t^k = k\Delta t$ and $T^k = T(t^k)$
  - $T^* = T^k - \Delta t\left(\left\{\phi, T^k\right\} + Q(t^k)\right)$
  - $\hat{T}^{k+1} = \hat{T}^* + \Delta t \nabla \cdot (A_x \nabla \hat{T}^*)$

Outline

Sequential algorithm

$$\partial_t T + \{\phi, T\} = \nabla \cdot (A(x) \nabla T)$$

Time loop

for all time iteration n **do**

$T^* \leftarrow T^n - \Delta t \{\phi, T^n\}$ — Advection operator

$\widehat{T^*} \leftarrow DFT2D^{+1}_{x,y}(T^*)$ — Discretization switch real to semi-spectral

$\widehat{T^{n+1}} \leftarrow \widehat{T^*} + \Delta t \nabla \cdot (A_x \nabla \widehat{T^*})$ — Diffusion operator

$T^{n+1} \leftarrow DFT2D^{-1}_{x,y}(\widehat{T^{n+1}})$ — Discretization switch semi-spectral to real

end for

Sequential algorithm

$$\partial_t T + \{\phi, T\} = \nabla \cdot (A(x)\nabla T)$$

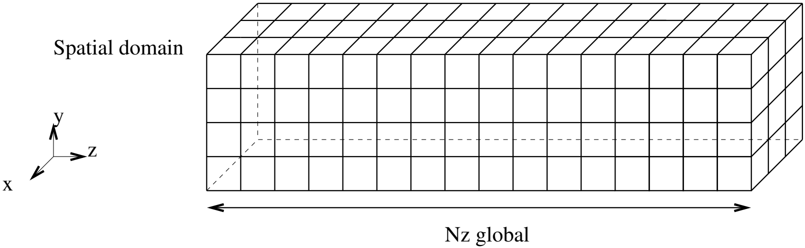
Time loop

for all time iteration n **do**
- $T^* \leftarrow T^n - \Delta t\,\{\phi, T^n\}$ — Advection operator
- $\widehat{T^*} \leftarrow DFT2D^{+1}_{x,y}(T^*)$ — Discretization switch real to semi-spectral
- $\widehat{T^{n+1}} \leftarrow \widehat{T^*} + \Delta t \nabla \cdot (A_x \nabla \widehat{T^*})$ — Diffusion operator
- $T^{n+1} \leftarrow DFT2D^{-1}_{x,y}(\widehat{T^{n+1}})$ — Discretization switch semi-spectral to real

end for

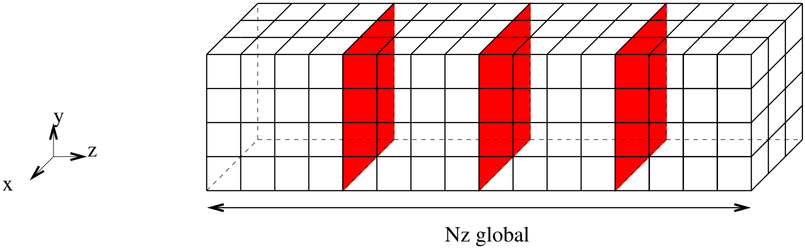
- Hotspot: 2D DFT computation
  - $\mathcal{O}(N_x \times N_y \log(N_y) \times N_z \log(N_z))$ vs $\mathcal{O}(N_x \times N_y \times N_z)$

MPI parallelization strategy



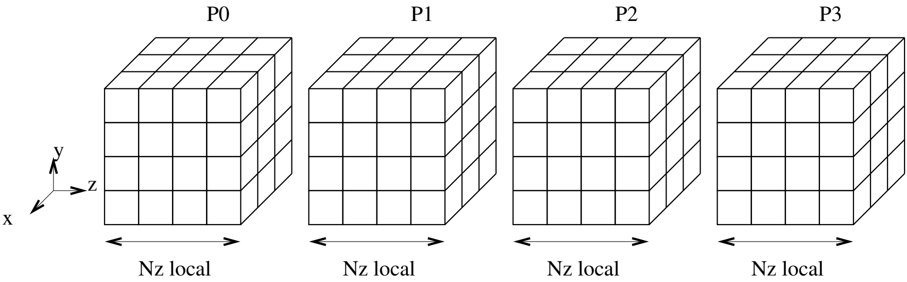
- domain decomposition along one spatial dimension
- no overlap

MPI parallelization strategy



- domain decomposition along one spatial dimension
- no overlap

MPI parallelization strategy



- domain decomposition along one spatial dimension
- no overlap

$\Rightarrow$ read dependencies along spatial dimensions considered as blocking

MPI parallelization: decomposition dimensions

MPI parallelization: decomposition dimensions

Advection (real space)

for all $k \in 1..N_z$ **do**
  for all $j \in 1..N_y, i \in 1..N_x$ **do**
    $Adv[i,j,k] \leftarrow Stencil_{x,y}(T[i,j,k])$
  end for
end for

MPI parallelization: decomposition dimensions

Advection (real space)

for all $k \in 1..N_z$ **do**
  for all $j \in 1..N_y, i \in 1..N_x$ **do**
    $Adv[i,j,k] \leftarrow Stencil_{x,y}(T[i,j,k])$
  end for
end for

Diffusion (semi-spectral space)

for all $k \in 1..N_z, j \in 1..N_y$ **do**
  for all $i \in 1..N_x$ **do**
    $Diff[i,j,k] \leftarrow Stencil_x(\widehat{T}[i,j,k])$
  end for
end for

MPI parallelization: decomposition dimensions

Advection (real space)

for all $k \in 1..N_z$ **do**
  for all $j \in 1..N_y, i \in 1..N_x$ **do**
    $Adv[i,j,k] \leftarrow Stencil_{x,y}(T[i,j,k])$
  end for
end for

Diffusion (semi-spectral space)

for all $k \in 1..N_z, j \in 1..N_y$ **do**
  for all $i \in 1..N_x$ **do**
    $Diff[i,j,k] \leftarrow Stencil_x(\widehat{T}[i,j,k])$
  end for
end for

1D DFT dimension y

for all $i \in 1..N_x, k \in 1..N_z$ **do**
  $hatT[i,:,k] \leftarrow DFT1D_y(T[i,:,k])$
end for

1D DFT dimension z

for all $i \in 1..N_x, j \in 1..N_y$ **do**
  $\widehat{T}[i,j,:] \leftarrow DFT1D_z(\hat{T}[i,j,:])$
end for

MPI parallelization: decomposition dimensions

Advection (real space)

for all $k \in 1..N_z$ **do**
  for all $j \in 1..N_y, i \in 1..N_x$ **do**
    $Adv[i,j,k] \leftarrow Stencil_{x,y}(T[i,j,k])$
  end for
end for

Diffusion (semi-spectral space)

for all $k \in 1..N_z, j \in 1..N_y$ **do**
  for all $i \in 1..N_x$ **do**
    $Diff[i,j,k] \leftarrow Stencil_x(\widehat{T}[i,j,k])$
  end for
end for

1D DFT dimension y

for all $i \in 1..N_x, k \in 1..N_z$ **do**
  $hatT[i,:,k] \leftarrow DFT1D_y(T[i,:,k])$
end for

1D DFT dimension z

for all $i \in 1..N_x, j \in 1..N_y$ **do**
  $\widehat{T}[i,j,:] \leftarrow DFT1D_z(\hat{T}[i,j,:])$
end for

$\Rightarrow$ need to redistribute between DFT

MPI algorithm with 1D × 1D DFT functions

Time loop

for all time iteration n **do**

$T^* \leftarrow T^n - \Delta t \{\phi, T^n\}$ — z distributed

$\hat{T}^* \leftarrow DFT_y^{+1}(T^*)$ — z distributed

Redistribute $\hat{T}^*$ along y

$\widehat{\hat{T}^*} \leftarrow DFT_z^{+1}(\hat{T}^*)$ — y distributed

$\widehat{\widehat{T^{n+1}}} \leftarrow \widehat{\hat{T}^*} + \Delta t \nabla \cdot (A_x \nabla \widehat{\hat{T}^*})$ — y distributed

$T^{\hat{n}+1} \leftarrow DFT_z^{-1}(\widehat{\hat{T}}^{n+1})$ — y distributed

Redistribute $T^{\hat{n}+1}$ along z

$T^{n+1} \leftarrow DFT_y^{-1}(T^{\hat{n}+1})$ — z distributed

end for

OpenMP

- Parallelization of the outermost spatial loops

Advection

♯ pragma omp parallel for collapse(2)
for all $k \in 1..N_{z,local}$ **do**
  for all $j \in 1..N_y$ **do**
    for all $i \in 1..N_x$ **do**
      $Adv[i,j,k] \leftarrow Stencil_{x,y}(T[i,j,k])$
    end for
  end for
end for

! Data locality: to minimize memory bandwidth requirements

! Thread binding: to minimize negative NUMA effects

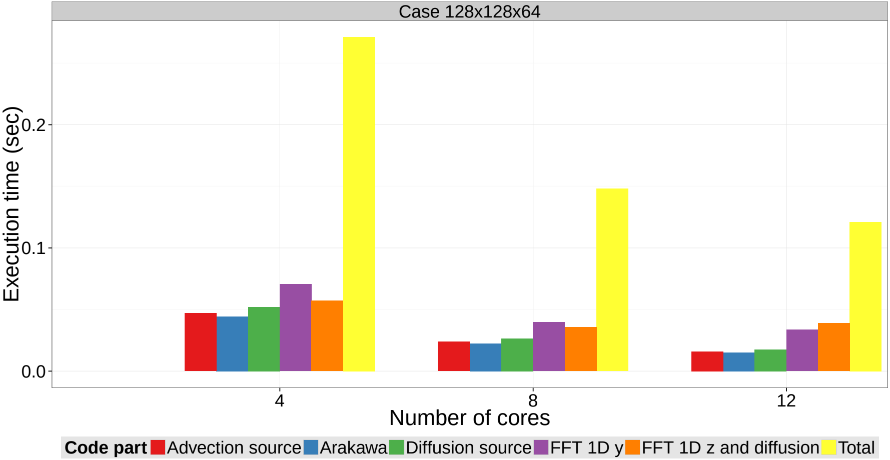
Outline

Architecture and compilation

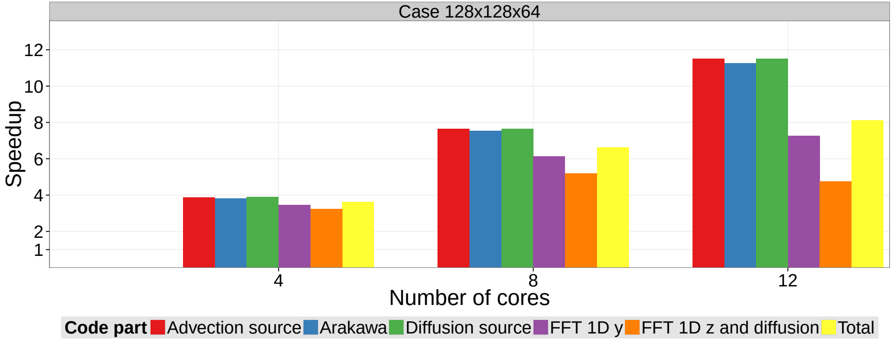
- Rheticus cluster of Aix-Marseille University
  - bi-socket nodes, 2×6 cores Intel Westmere processors
  - 24 GB RAM / node
- Libraries
  - FFTW3 version 3.3
  - Open MPI version 1.6.3 + mpiexec with thread binding
- Intel compiler
  - OpenMP
  - auto vectorization

Outline

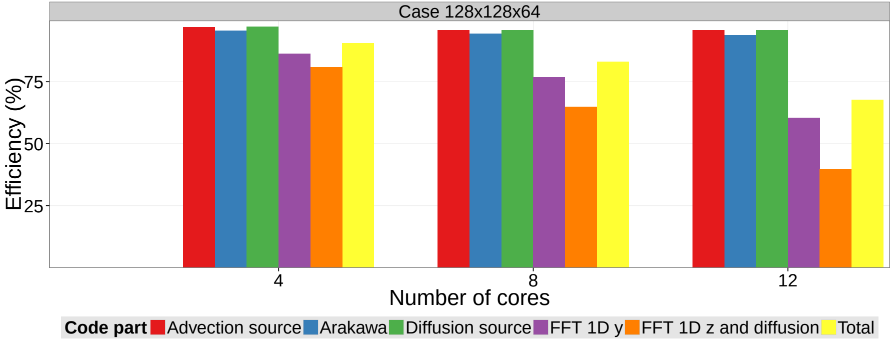
Small test case: times



Small test case: speedups



Small test case: efficiencies



Problem

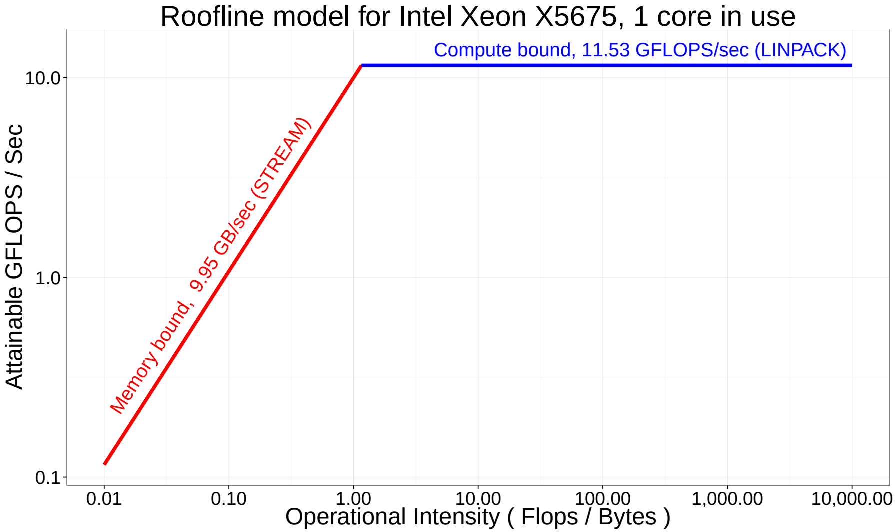
- With times, speedups and efficiencies
  - why DFT performances does not scale ?
  - how good are performances compared to hardware's peak performances ?

Roofline model [1]

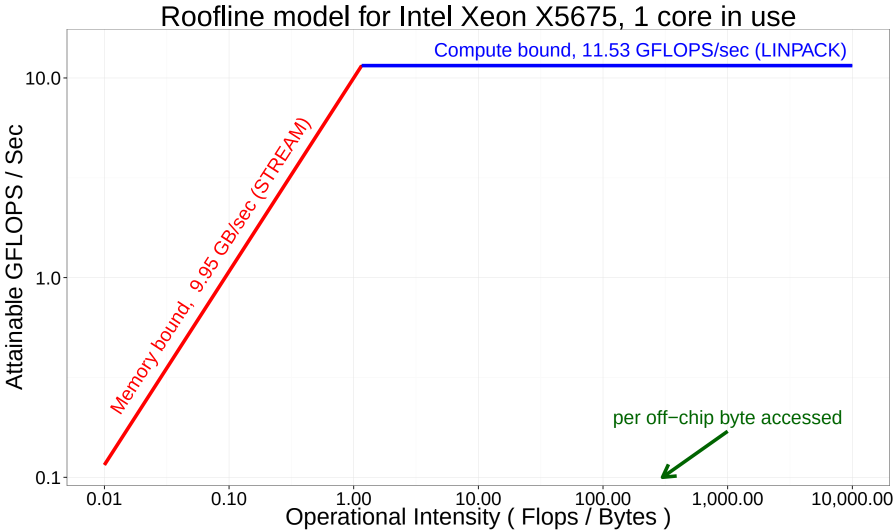
- More insightful than times, speedups and efficiencies
- Allows to compare code performances to
  - peak floating-point performance
  - peak memory bandwidth
- for a given architecture

[1] S. Williams et al, *Roofline: an insightful visual performance model for multicore architectures.* Commun. ACM, vol. 52, no. 4

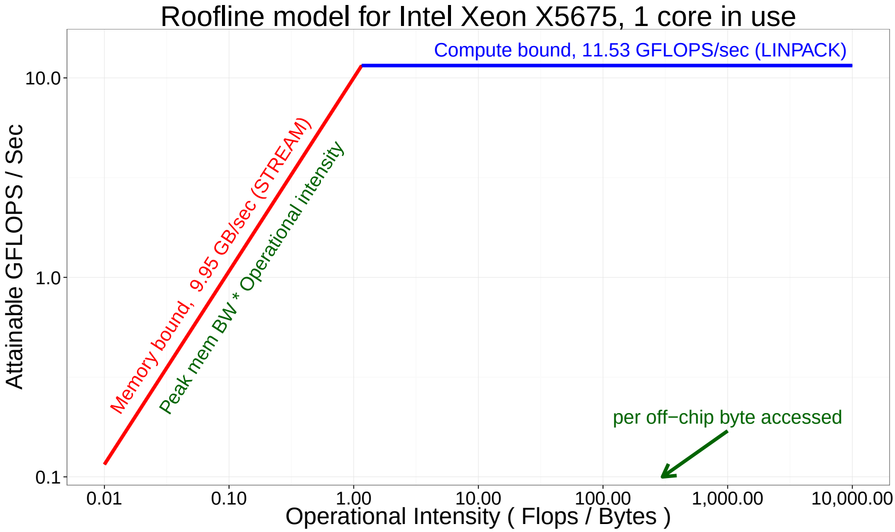
Roofline model: 1 core



Roofline model: 1 core



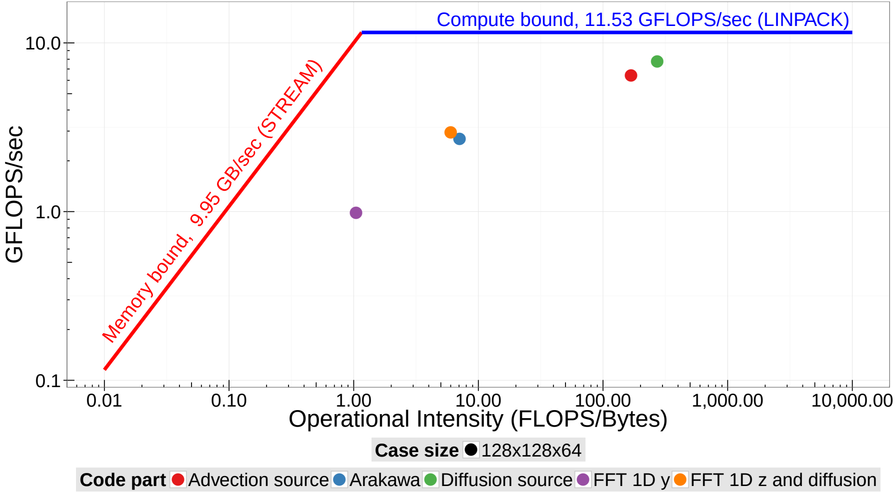
Roofline model: 1 core



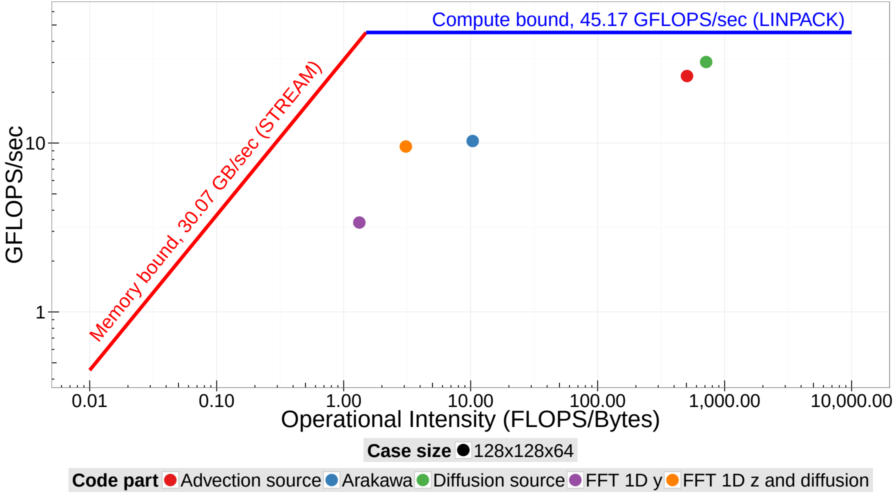
Measures

- For each desired code part
  - Time
- And estimations or hardware counter measures of
  - FLOPS
  - Off-chip memory traffic

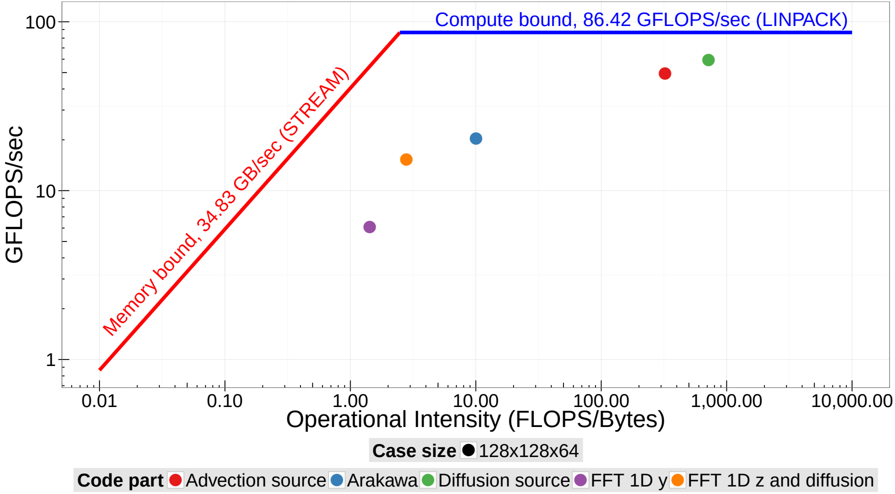
Roofline analysis: 1 core



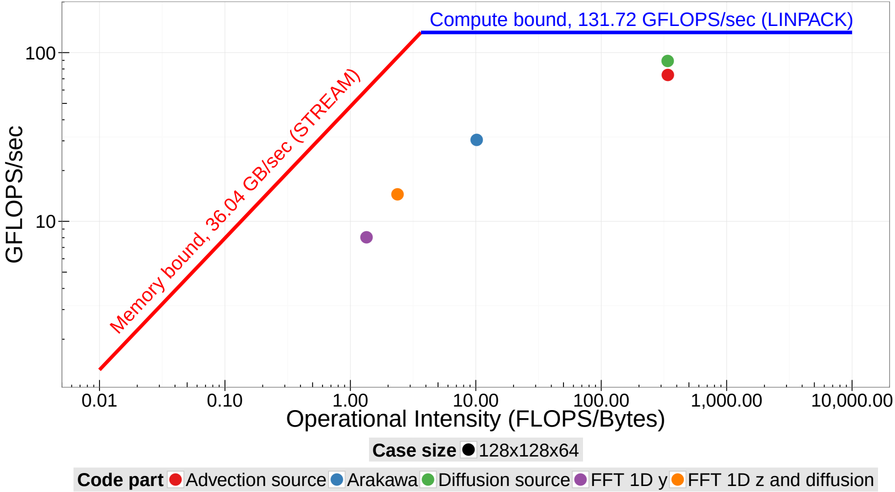
Roofline analysis: 4 cores



Roofline analysis: 8 cores

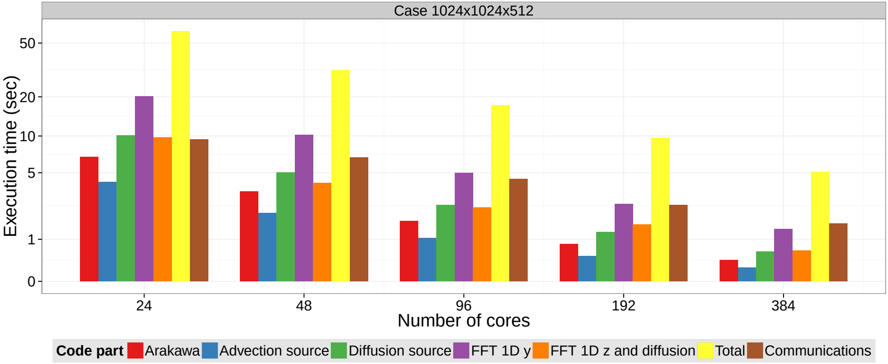


Roofline analysis: 12 cores



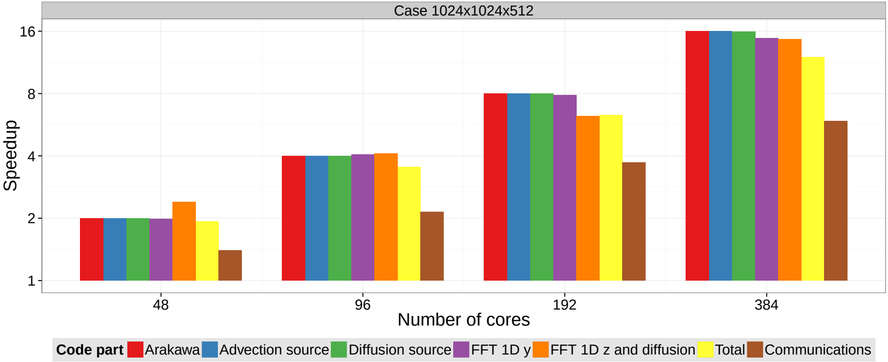
Outline

Multi-node performances: times



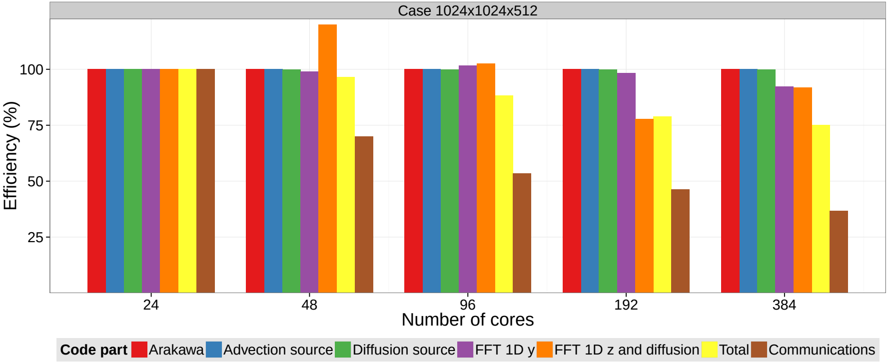
12 cores per node

Multi-node performances: speedups



12 cores per node

Multi-node performances: efficiencies



12 cores per node

Outline

Conclusion

- Viable hybrid MPI/OpenMP strategy for Emedge3D:
  - high resolution grids reachable (memory resources ++)
  - promising scalings, with 79% efficiency on 384 cores
- Perspectives:
  - perform tests on larger parallel systems
  - consider overlap in domain decomposition
  - couple parallelization strategy with semi-implicit time integration
  - integration of the parallelization strategy in Emedge3D

Thank you for listening!

1. *Parallelization of an Advection-Diffusion Problem Arising in Edge Plasma Physics Using Hybrid MPI/OpenMP Programming*, Euro-Par 2015: Parallel Processing, 2015

2. *Comparison of numerical solvers for anisotropic diffusion equations arising in plasma physics*, J. Sci. Comput., 2014

matthieu.kuhn@inria.fr