GYSELA5D

Adapting a GYrokinetic SEmi-LAgrangian code for current architectures and towards Exascale

J. Bigot[1], V. Grandgirard[2], G. Latu[2], C. Passeron[2], F. Rozar[1,2]

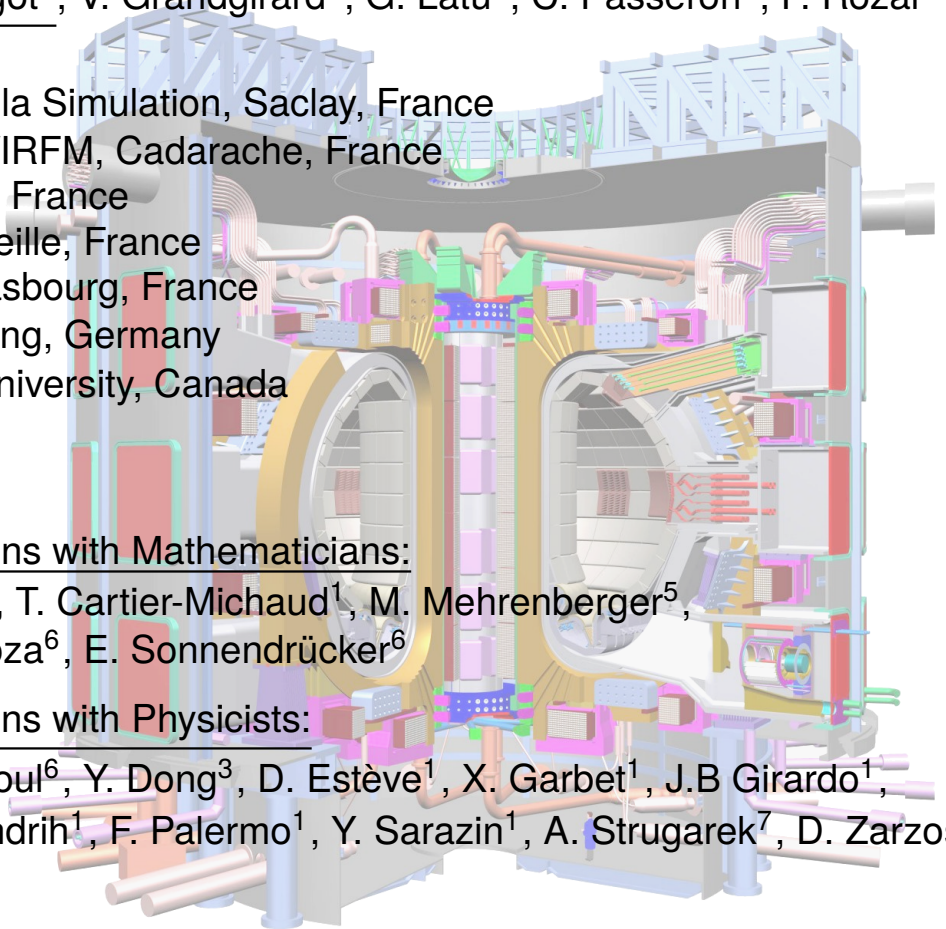
[1]Maison de la Simulation, Saclay, France
[2]CEA/DSM/IRFM, Cadarache, France
[3]LPP, Paris, France
[4]CPT, Marseille, France
[5]IRMA, Strasbourg, France
[6]IPP Garching, Germany
[7]Montreal university, Canada

Collaborations with Mathematicians:

A. Back[4], T. Cartier-Michaud[1], M. Mehrenberger[5], L. Mendoza[6], E. Sonnendrücker[6]

Collaborations with Physicists:

J. Abiteboul[6], Y. Dong[3], D. Estève[1], X. Garbet[1], J.B Girardo[1], Ph. Ghendrih[1], F. Palermo[1], Y. Sarazin[1], A. Strugarek[7], D. Zarzoso[6]

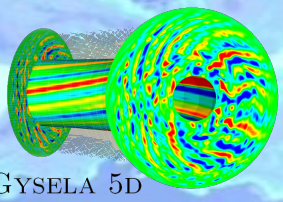


2014-04-01: CANUM 2014, À nouvelles machines, nouveaux algorithmes

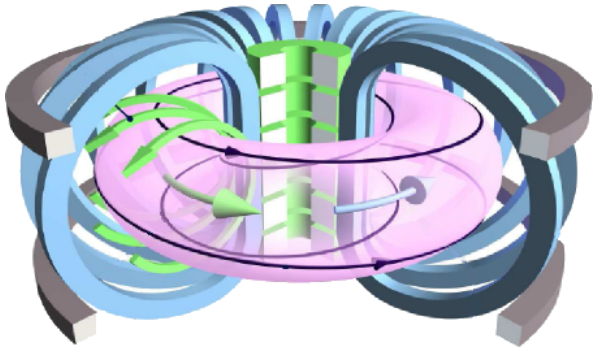
MAISON DE LA SIMULATION

Context

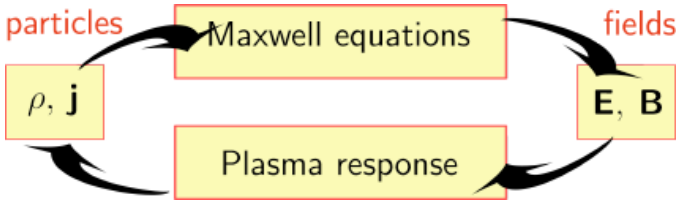
Magnetic confined fusion



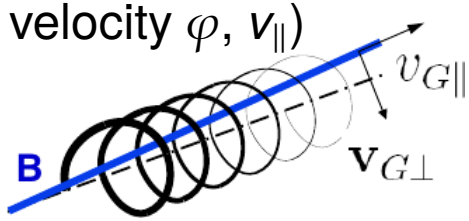
- ITER project (International project, construction in Cadarache):
  - Toroidal reactor for fusion reaction (tokamak)
  - Magnetic confinement of hot (10^8 K) plasma
  - Goals: $\approx$ 500 ms pulse, energy ratio $>$ 1
- Challenge: understand turbulence development
  - Loss of heat and matter, limits time of pulse



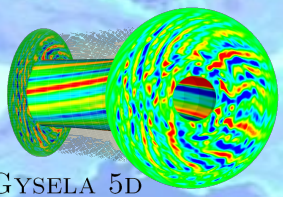
First Principle simulation



- Electromagnetic fields governed by charge density: **Poisson** solver
- Particle motion governed by electromagnetic fields: **Vlasov** solver
  - Kinetic theory $\Rightarrow$ 6D (3D space + 3D velocity)
  - Gyrokinetic approximation $\Rightarrow$ **5D** (3D space r, θ, μ + 2D velocity φ, $v_\parallel$)
    - \+ Much reduced Computation, Memory
    - \- Increased complexity



Outline

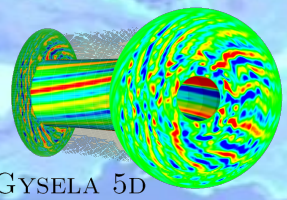


MAISON DE LA SIMULATION

MAISON DE LA SIMULATION

Code Structure

The algorithm



Input : Physics parameters
Input : Distribution function ($\bar{f}^0$)
Output : Distribution function ($\bar{f}^N$) + Diagnostics

for *time step* $n \in 0 \rightarrow N$ **do**
  Integrals: $\mathcal{N}_i^n(r,\theta,\varphi) = \int\int \bar{f}^n \quad B(r,\theta)\, \mathcal{J}(k_\perp \rho_C)\, dv_\parallel d\mu$
  Compute fields (Poisson): $\mathcal{N}_i^n(r,\theta,\varphi) \rightarrow \Phi^n(r,\theta,\varphi)$
  Diagnostics for time step n
  Advect particles (Maxwell): $\Phi^n(r,\theta,\varphi), \bar{f}^n \rightarrow \bar{f}^{n+1}$
end

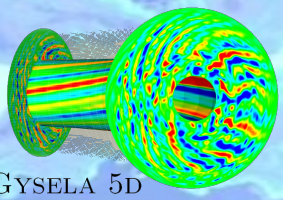
Algorithm 1: Overall Gysela algorithm

- Main unknown: $\bar{f}^n(r,\theta,\varphi,v_\parallel,\mu)$
- Practically: RK2 time integration scheme $O(\Delta t^2)$

MAISON DE LA SIMULATION

Advections

Vlasov Equation



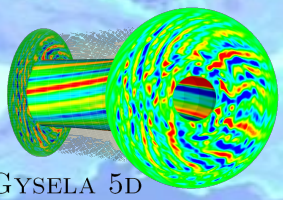
- Simplified view of Maxwell equation: Vlasov (collisionless, no RHS)
  - $\frac{\partial \bar{f}}{\partial t} + \frac{dr}{dt}\frac{\partial \bar{f}}{\partial r} + \frac{d\theta}{dt}\frac{\partial \bar{f}}{\partial \theta} + \frac{d\varphi}{dt}\frac{\partial \bar{f}}{\partial \varphi} + \frac{dv_{\|}}{dt}\frac{\partial \bar{f}}{\partial v_{\|}} = 0$

- Solved through Splitting of Strang:
  - $\partial_t \bar{f} + v_{\|}\partial_{\varphi}\bar{f} = 0$ ($\frac{\hat{\varphi}}{2}$ operator)
  - $\partial_t \bar{f} + \dot{v}_{\|}\partial_{v_{\|}}\bar{f} = 0$ ($\frac{\hat{v_{\|}}}{2}$ operator)
  - $\partial_t \bar{f} + \overrightarrow{v_{GC}} \cdot \overrightarrow{\nabla}_{\perp}\bar{f} = 0$ ($\hat{r\theta}$ operator)
  - $\partial_t \bar{f} + \dot{v}_{\|}\partial_{v_{\|}}\bar{f} = 0$ ($\frac{\hat{v_{\|}}}{2}$ operator)
  - $\partial_t \bar{f} + v_{\|}\partial_{\varphi}\bar{f} = 0$ ($\frac{\hat{\varphi}}{2}$ operator)

MAISON DE LA SIMULATION

Advections

Vlasov Equation

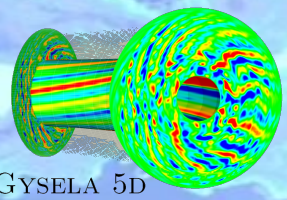


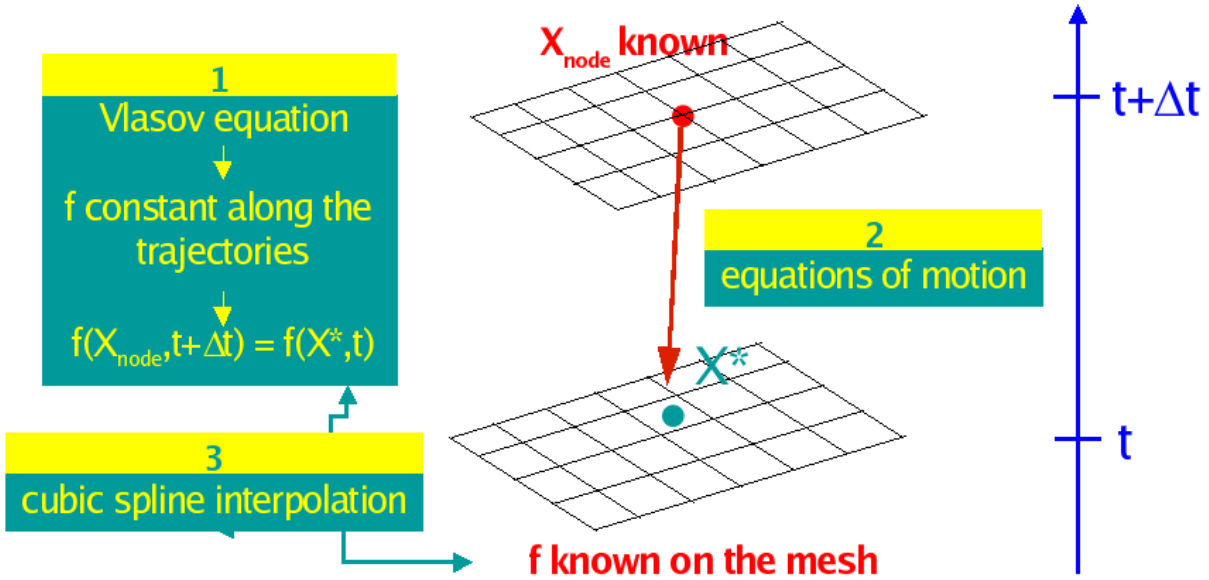
- Simplified view of Maxwell equation: Vlasov (collisionless, no RHS)
  - $\frac{\partial \bar{f}}{\partial t} + \frac{dr}{dt}\frac{\partial \bar{f}}{\partial r} + \frac{d\theta}{dt}\frac{\partial \bar{f}}{\partial \theta} + \frac{d\varphi}{dt}\frac{\partial \bar{f}}{\partial \varphi} + \frac{dv_{\|}}{dt}\frac{\partial \bar{f}}{\partial v_{\|}} = 0$
- Solved through Splitting of Strang:
  - $\partial_t \bar{f} + v_{\|}\partial_{\varphi}\bar{f} = 0$ ($\frac{\hat{\varphi}}{2}$ operator)
  - $\partial_t \bar{f} + \dot{v}_{\|}\partial_{v_{\|}}\bar{f} = 0$ ($\frac{\hat{v_{\|}}}{2}$ operator)
  - $\partial_t \bar{f} + \overrightarrow{v_{GC}} \cdot \overrightarrow{\nabla}_{\perp}\bar{f} = 0$ ($\hat{r\theta}$ operator)
  - $\partial_t \bar{f} + \dot{v}_{\|}\partial_{v_{\|}}\bar{f} = 0$ ($\frac{\hat{v_{\|}}}{2}$ operator)
  - $\partial_t \bar{f} + v_{\|}\partial_{\varphi}\bar{f} = 0$ ($\frac{\hat{\varphi}}{2}$ **operator**)
- Let's focus on one advection: The semi-lagrangian scheme

MAISON DE LA SIMULATION

Advections

Semi-Lagrangian scheme





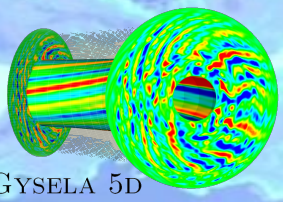
- $\bar{f}$ **conserved along characteristics**
- Find the origin of the characteristics ending at the grid points
- Interpolate value at origin $X^\star$ from known grid values: **Cubic spline interpolation**

- Main cost: Cubic spline interpolation

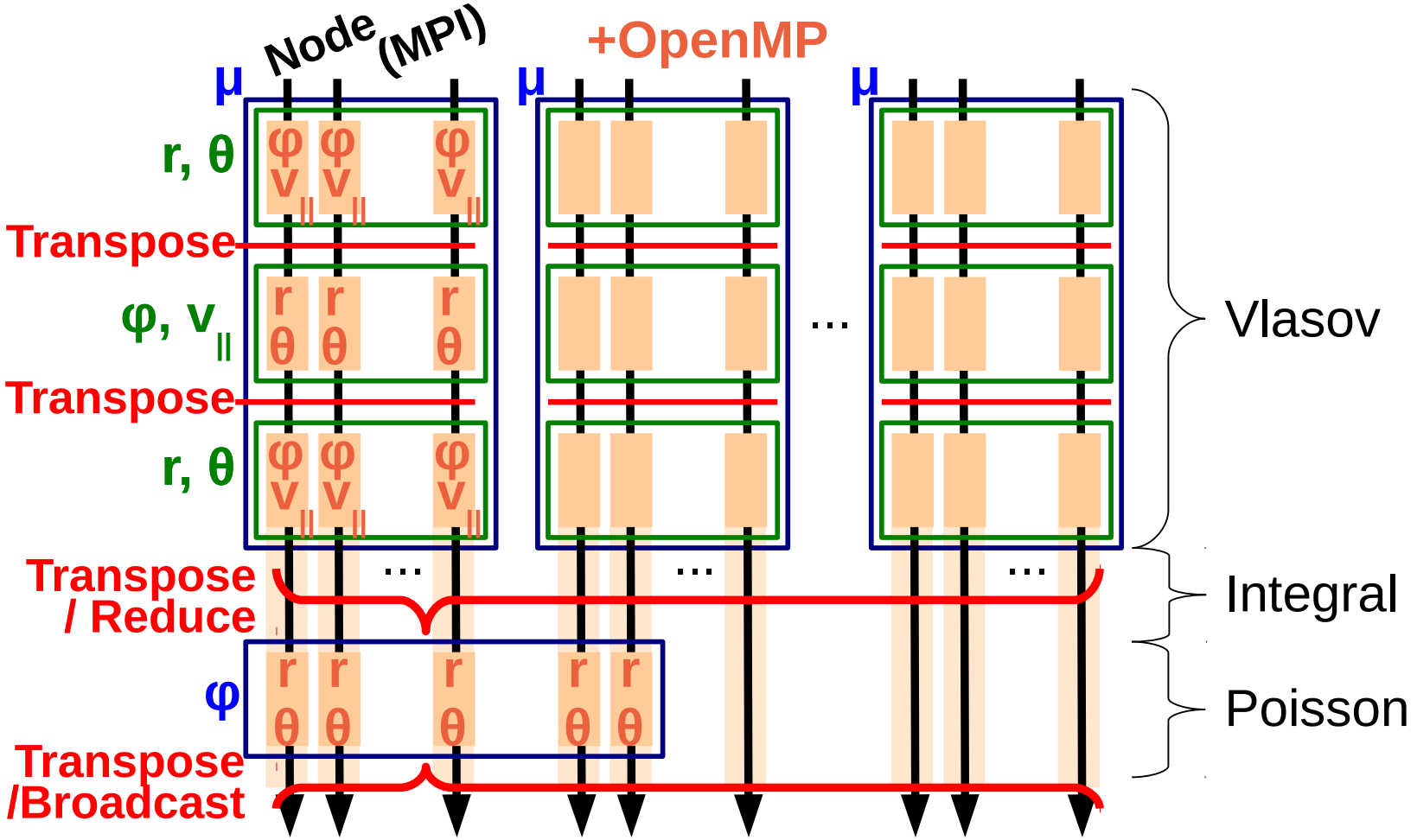
MAISON DE LA SIMULATION

Code Structure

Parallelization (Data Decomposition)

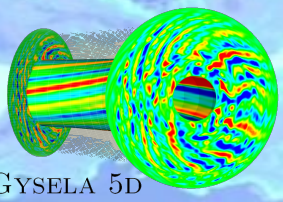


Main data: 5D array ((r, θ, φ) space, $(v_{\parallel}, \mu)$ velocity)



MAISON DE LA SIMULATION

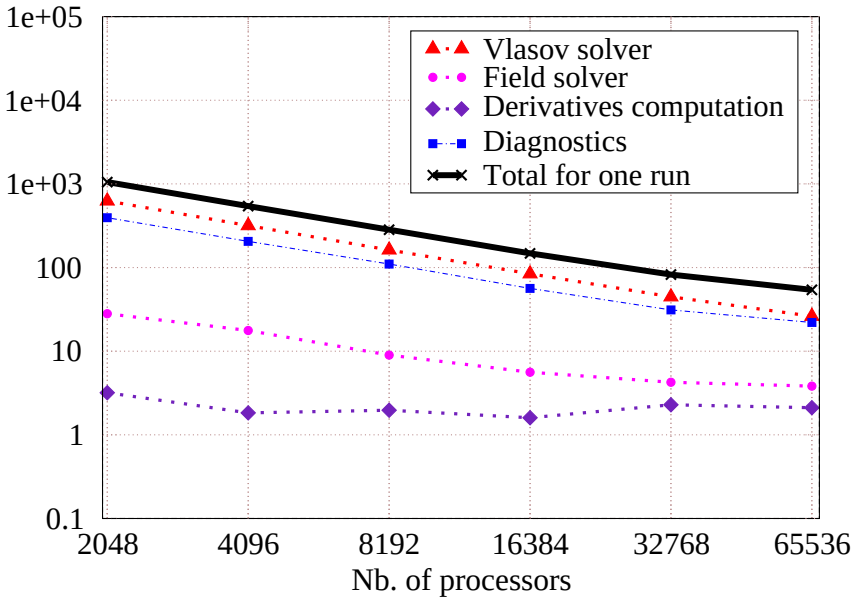
Scalability and bottlenecks



Strong scaling: $N_r = 512,\ N_\theta = 512,\ N_\varphi = 128,\ N_{v\|} = 128$

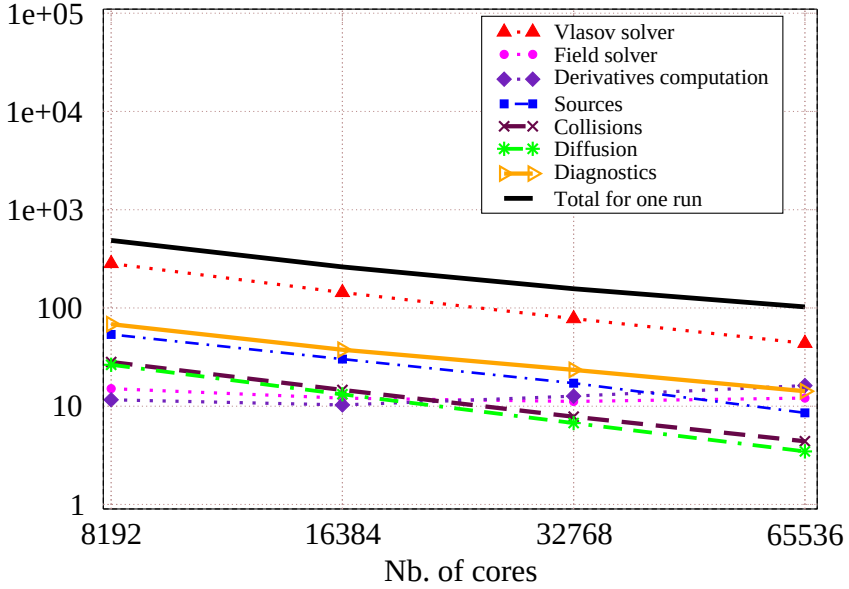
$M_\mu = 32$, main data=1 TiB

Execution time, one run (Curie)



$M_\mu = 16$, main data=512 GiB

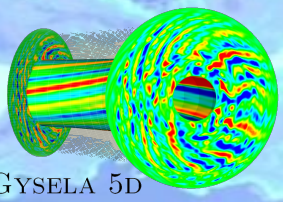
Execution time, one run (Turing)



- time dominated by Vlasov solver

MAISON DE LA SIMULATION

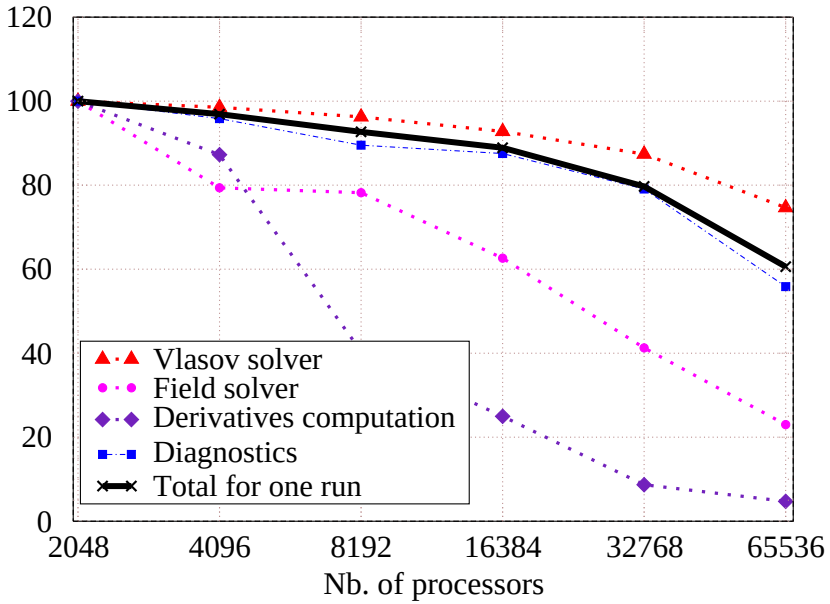
Scalability and bottlenecks



Strong scaling: $N_r = 512,\ N_\theta = 512,\ N_\varphi = 128,\ N_{v\|} = 128$

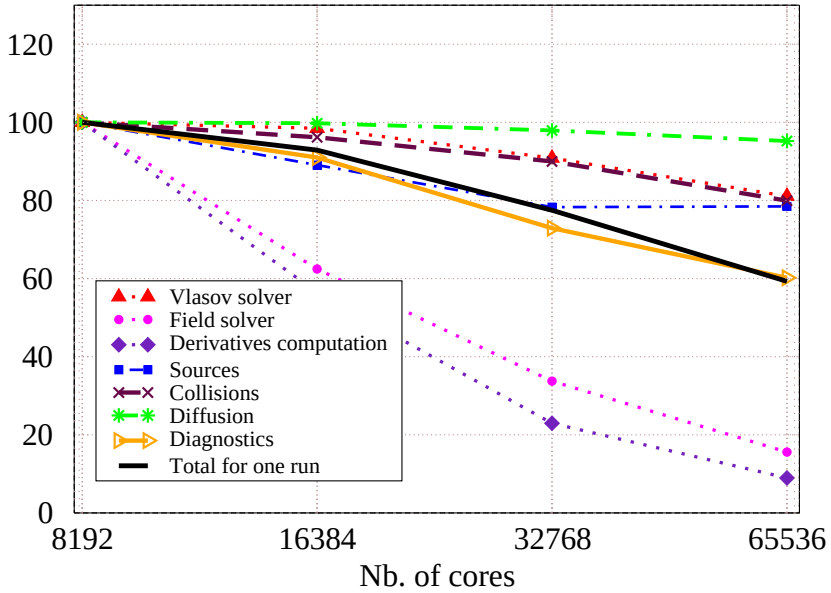
$M_\mu = 32$, main data=1 TiB

Relative efficiency, one run (Curie)



$M_\mu = 16$, main data=512 GiB

Relative efficiency, one run (Turing)



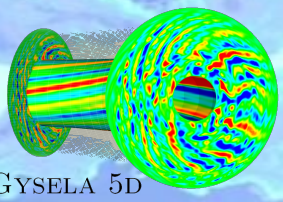
≈ 60% efficiency at 64 ki core on both

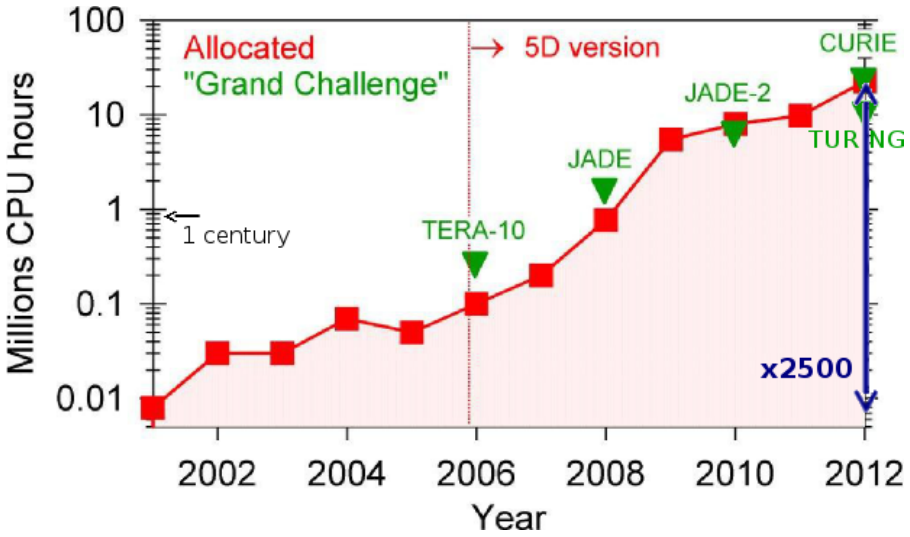
- time dominated by Vlasov solver
- scaling bottleneck: Poisson solver

MAISON DE LA SIMULATION

Computational needs

Exascale is a requirement





	Number of Points (ρ*=ρ/a)	Time / Ω_c	Number of cores	Number of days of simulation
Gd Challenge CINES 2010	272 billions (ρ*=1/512)	147 840	8192	31
Gd Challenge CURIE 2012	33 billions (ρ*=1/150)	678 510	16 384	15
	=> Adding of tritium		32768	6
Comparison with experiment (in progress)	87 billions (ρ*=1/300)	2 000 000	5520	46

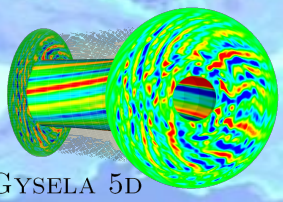
Some big simulations

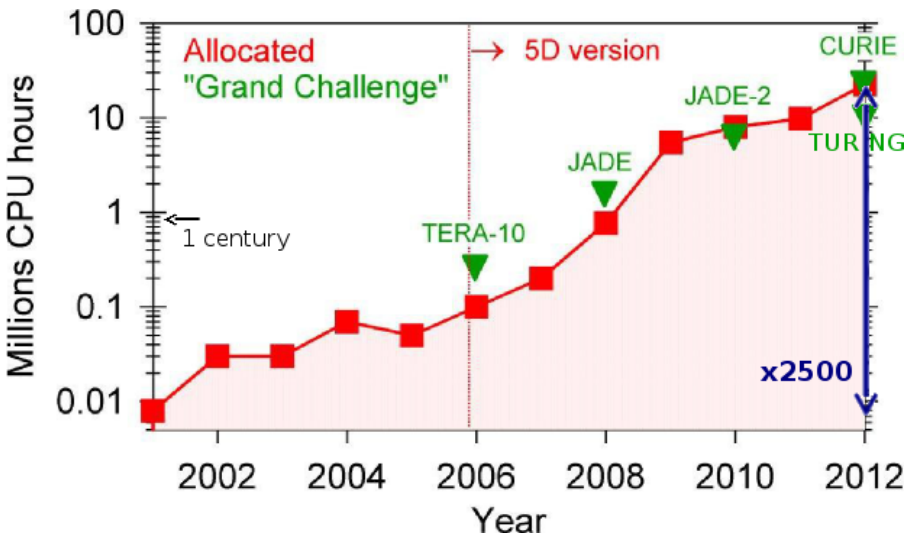
- ITER-like size:
  - 272.10^6 points
- Longest simulated time:
  - ~ 1/2 energy confinement time

MAISON DE LA SIMULATION

Computational needs

Exascale is a requirement





	Number of Points (ρ*=ρ/a)	Time / Ω_c	Number of cores	Number of days of simulation
Gd Challenge CINES 2010	272 billions (ρ*=1/512)	147 840	8192	31
Gd Challenge CURIE 2012	33 billions (ρ*=1/150)	678 510	16 384	15
	=> Adding of tritium		32768	6
Comparison with experiment (in progress)	87 billions (ρ*=1/300)	2 000 000	5520	46

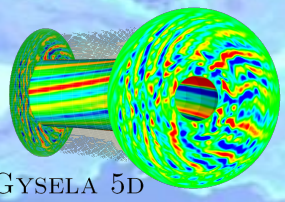
Compromises required

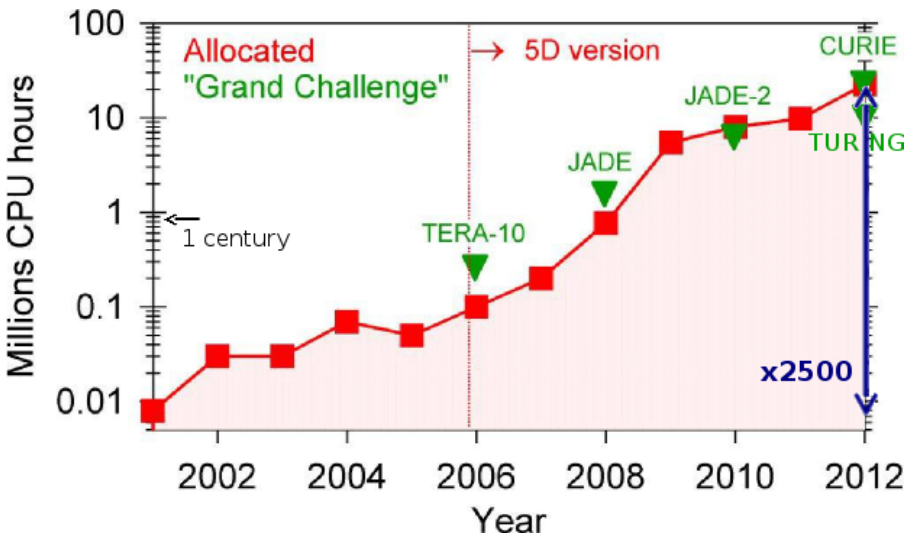
- ITER-like size:
  - 272.10^6 points
  - ☹ $N = 9$ instead of 18 for ripple effects
- Longest simulated time:
  - $\sim 1/2$ energy confinement time
  - ☹ Several τ_E times not accessible

MAISON DE LA SIMULATION

Computational needs

Exascale is a requirement

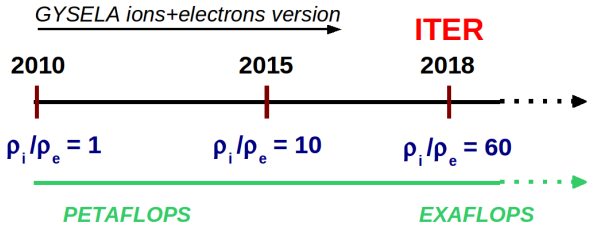




	Number of Points (ρ*=ρ/a)	Time / Ω_c	Number of cores	Number of days of simulation
Gd Challenge CINES 2010	272 billions (ρ*=1/512)	147 840	8192	31
Gd Challenge CURIE 2012	33 billions (ρ*=1/150)	678 510	16 384	15
	=> Adding of tritium		32768	6
Comparison with experiment (in progress)	87 billions (ρ*=1/300)	2 000 000	5520	46

More physics to come

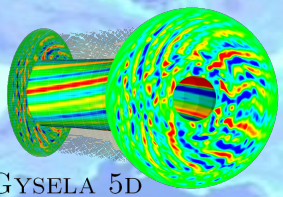
- ITER-like size:
  - 272.10^6 points
  - ☹ $N = 9$ instead of 18 for ripple effects
- Longest simulated time:
  - $\sim 1/2$ energy confinement time
  - ☹ Several τ_E times not accessible
- Kinetic electrons ($\rho_{ions}/\rho_{elec} = 60$)
  - mesh size $\times\ 60^3$
  - time step / 60



- **Exascale required !**

MAISON DE LA SIMULATION

Let's go to Exascale



Nobody really knows what Exascale computers will look like...
... but we already have some insight

Multilevel parallelism

Expensive communications

Expensive I/O

Let's go to Exascale

High fault rate

High core count

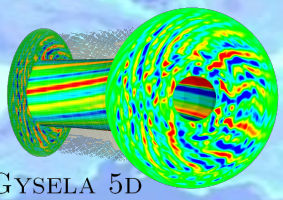
Low memory amount

Small cores

MAISON DE LA SIMULATION

Poisson Communications

Non Optimized Algorithm



GYSELA 5D

- The guiding-center density n_{Gi} is computed by

$$n_{Gi}(r,\theta,\varphi) = 2\pi \int B(r,\theta)d\mu \int dv_{//} J_0(\mu)\bar{f}(r,\theta,\varphi,v_{//},\mu)$$

- Quasi Neutral Poisson equation can be written

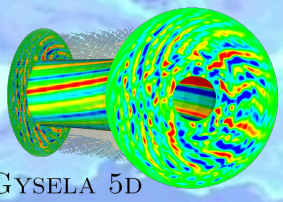
$$-\frac{1}{n_0(r)}\nabla_\perp \cdot \left[\frac{n_0(r)}{B_0}\nabla_\perp \Phi(r,\theta,\varphi)\right] + \frac{1}{T_e(r)}\left[\Phi(r,\theta,\varphi) - \langle\Phi\rangle_{\theta,\varphi}(r)\right] = \frac{n_{Gi}(r,\theta,\varphi)}{n_0(r)}$$

- Strategy to solve this Poisson equation:
  1. Solve separatly the average $\langle\Phi\rangle_{\theta,\varphi}(r)$
  2. Parallel loop in φ:
     - 2.1 FFT 1D in θ
     - 2.2 Finite differences with respect to variable r
     - 2.3 Get $\Phi(r=*,\theta=*,\varphi)$
  3. MPI Gather and Broadcast of all $\Phi(r=*,\theta=*,\varphi=*)$

MAISON DE LA SIMULATION

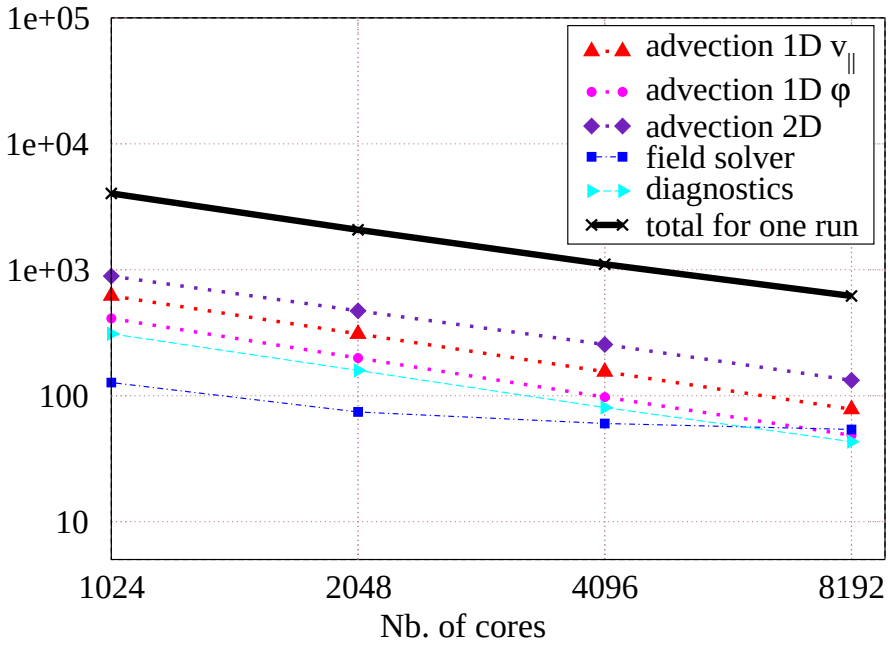
Poisson Communications

The Problem

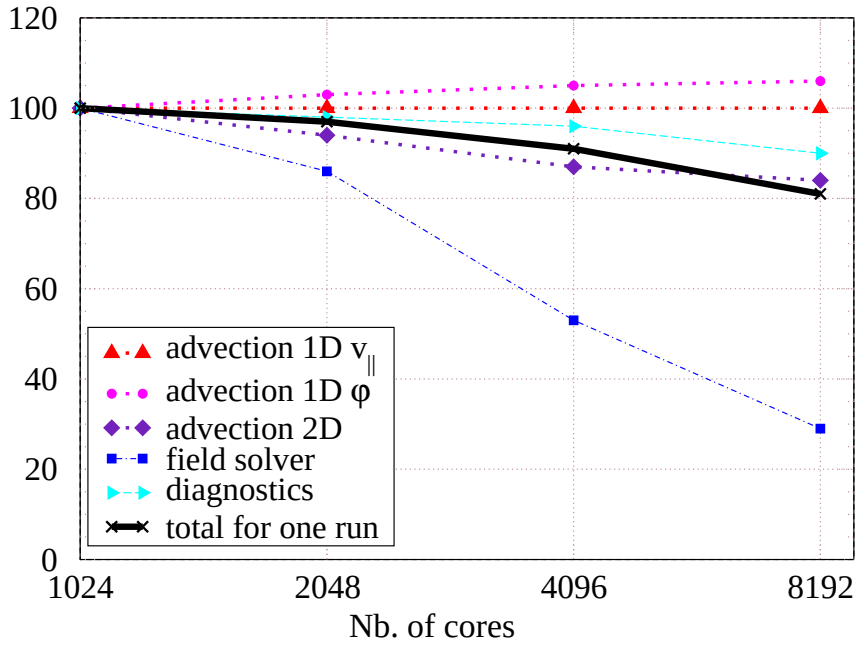


main data = 750 GB ($N_r = 512$, $N_\theta = 512$, $N_\varphi = 256$, $N_{v_\parallel} = 47$, $N_\mu = 32$)

Execution time for one Gysela run



Relative efficiency for one Gysela run

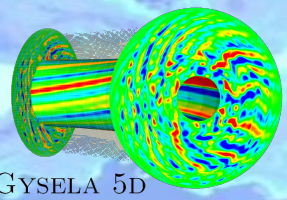


- Good result: 82% relative efficiency at 8192 cores
- **BUT**: 1) lack of memory scalability,
  2) communication cost grows quickly - field solver

MAISON DE LA SIMULATION

Poisson Communications

Revising the Algorithm



- Old algorithm
  1. Broadcast of 3D data Φ to all nodes
     $\rightarrow$ comm. cost: $\Theta(N_r\, N_\theta\, N_\varphi\, Nb\_cores)$
  2. Redondant computing of Φ derivatives (each node)

- New algorithm: **Avoid Φ broadcast**
  1. Send Φ to N_μ communicators
     $\rightarrow$ comm. cost: $\Theta(N_r\, N_\theta\, N_\varphi\, N_\mu)$
  2. Parallel computing of Φ derivatives
     $\rightarrow$ comm. cost: $\Theta(N_r\, N_\theta\, N_\varphi\, N_\mu)$

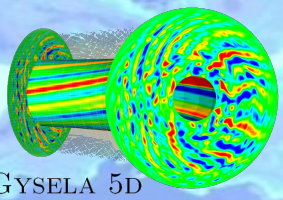
- Reduce the asymptotic communication cost

[G. Latu]

MAISON DE LA SIMULATION

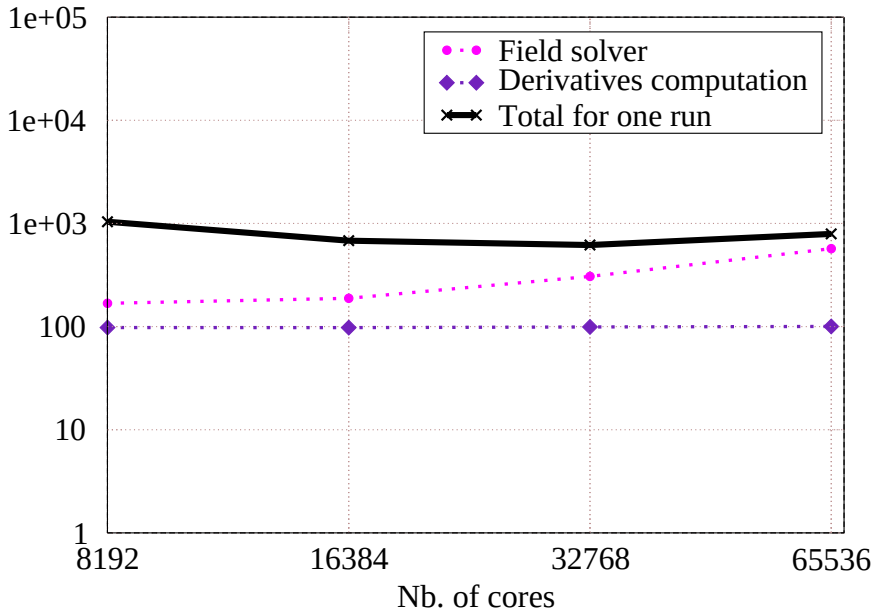
Poisson Communications

Result

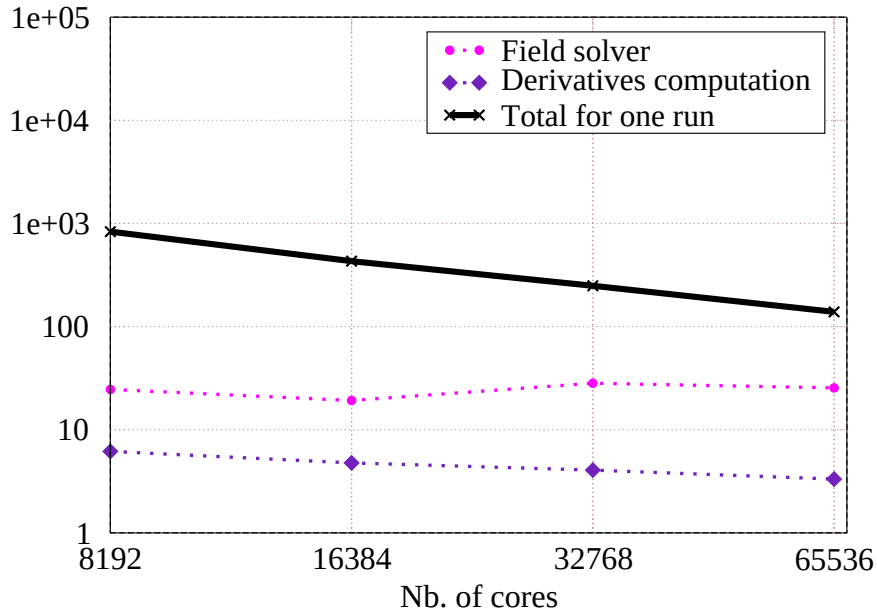


main data = 1 TB $(N_r = 512,\ N_\theta = 1024,\ N_\varphi = 128,\ N_{v_\parallel} = 64,\ N_\mu = 32)$

Old version - Execution time for one run



New version - Execution time for one run



Strong scaling (Jaguar @ Oak Ridge 2011)

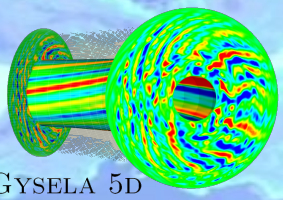
- More computation, but...
  - QN solver & derivatives computation **scale**
  - Good performance at 64k cores

[G. Latu]

MAISON DE LA SIMULATION

Communications

Data compression



- Sent data: electromagnetic field ⇒ smooth 3D field ⇒ low entropy
  - MPI comm. data **easily compressible**
- Compression time vs. ratio compromise
  - Typical algorithms too slow
  - Designed a **dedicated library (GCL)**: predict and send difference
  - Tried **1-pass wavelets** and **stream prediction**, both O(N)

	Processing (ms)	Compr. (%)	Comm (ms)	Total (s)	
Base	0	0	1180	1.18	
Wavelet	235	22	925	1.16	98%
Stream	270	30	848	1.12	96%

Jade, 128 nodes

	Processing (ms)	Compr. (%)	Comm (ms)	Total (s)	
Base	0	0	1460	1.46	
Wavelet	231	21	1158	1.39	95%
Stream	271	29	1021	1.29	93%

Jade, 512 nodes

Weak scaling, uncompressed data sent: 512Mio/node

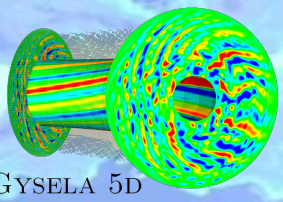
- Bad compression ratio but very fast
- Already good @ 128 nodes & **scales perfectly** ⇒ usefulness ↗ with size

[O. Thomine]

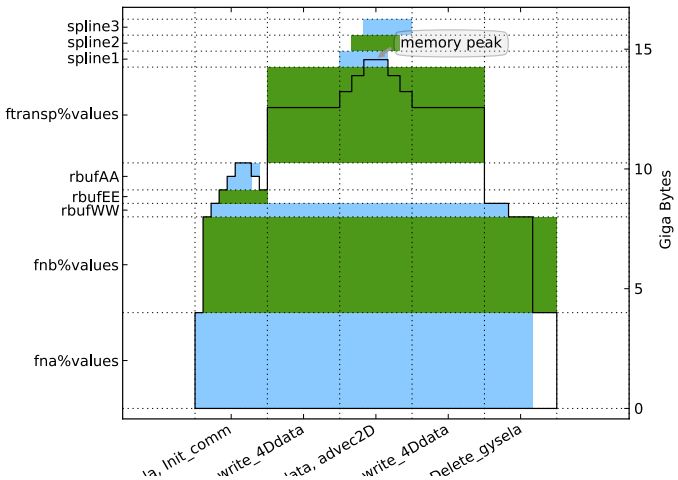
MAISON DE LA SIMULATION

Memory

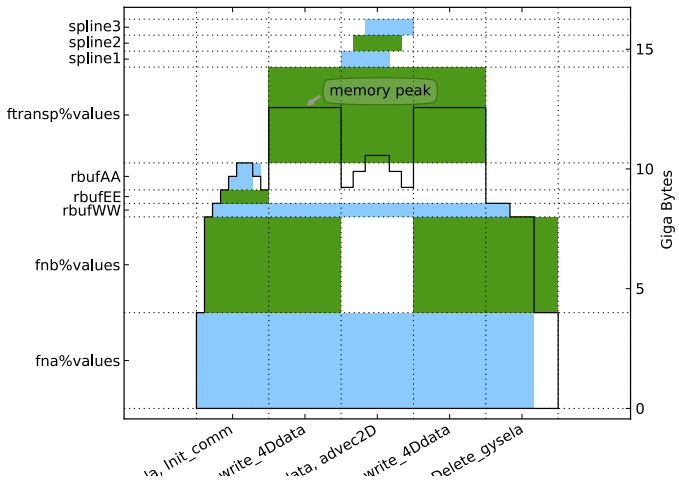
Scalability improvement



- GYSELA is global ⇒ Huge meshes ⇒ **Constrained by memory per node**
- Development of the **MTM library** (Memory Trace & Modeling)
  - Instrument memory allocations: `allocate`→`mtm_allocate`
  - Trace file generated ⇒ vizualization of memory peak
    - Deallocate unneeded variables at peak



Before optimisation



After optimisation

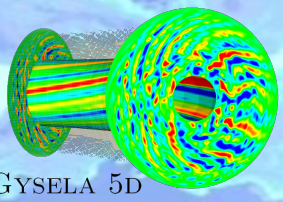
- Static to dynamic memory alloc. + improvement of algorithms
  ⇒ Memory footprint **divided by 2** on 32*k* cores

[F. Rozar et al., PPAM2013]

MAISON DE LA SIMULATION

Memory

Behavior prediction



- Static memory allocation had some advantages
  - Predictable memory consumption (and exhaustion)
  - No memory allocation overhead
- Additional code instrumentation to extend trace file

```
MTM_param(nthetha) // parameters whose value can be changed
MTM_param(nphi)
MTM_expr(dimx = nthetha*nphi) // a derived value
MTM_alloc(tabx, dimx) // an actual allocation
```

- Enables replay with changed parameters value ⇒ **prediction**

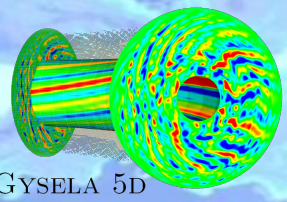
Number of MPI processes	128	256	512	1024	2048
4D structures	209.2	107.1	56.5	28.4	14.4
3D structures	62.7	36.0	22.6	19.7	18.3
2D structures	33.1	33.1	33.1	33.1	33.1
1D structures	6.6	3.4	2.0	1.7	1.6
Total per MPI process in GBytes	311.5	179.6	114.2	83.0	67.5

[F. Rozar et al.]

MAISON DE LA SIMULATION

Memory

Behavior prediction



- Static memory allocation had some advantages
  - Predictable memory consumption (and exhaustion)
  - No memory allocation overhead
- Additional code instrumentation to extend trace file

```
MTM_param(nthetha) // parameters whose value can be changed
MTM_param(nphi)
MTM_expr(dimx = nthetha*nphi) // a derived value
MTM_alloc(tabx, dimx) // an actual allocation
```

- Enables replay with changed parameters value ⇒ **prediction**

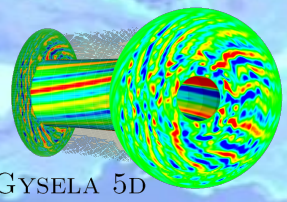
Number of MPI processes	128	256	512	1024	2048
4D structures	209.2	107.1	56.5	28.4	14.4
3D structures	62.7	36.0	22.6	19.7	18.3
2D structures	**33.1**	**33.1**	**33.1**	**33.1**	**33.1**
1D structures	6.6	3.4	2.0	1.7	1.6
Total per MPI process in GBytes	311.5	179.6	114.2	83.0	**67.5**

[F. Rozar et al.]

MAISON DE LA SIMULATION

Memory

Behavior prediction



- Static memory allocation had some advantages
  - Predictable memory consumption (and exhaustion)
  - No memory allocation overhead
- Additional code instrumentation to extend trace file

```
MTM_param(nthetha) // parameters whose value can be changed
MTM_param(nphi)
MTM_expr(dimx = nthetha*nphi) // a derived value
MTM_alloc(tabx, dimx) // an actual allocation
```

- Enables replay with changed parameters value ⇒ **prediction**

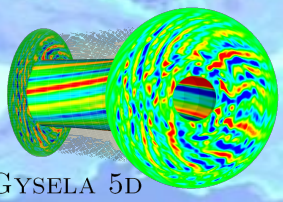
Number of MPI processes	128	256	512	1024	2048
4D structures	209.2	107.1	56.5	28.4	14.4
3D structures	62.7	36.0	22.6	19.7	18.3
2D structures	**7.1**	**7.1**	**7.1**	**7.1**	**7.1**
1D structures	6.6	3.4	2.0	1.7	1.6
Total per MPI process in GBytes	311.5	179.6	114.2	83.0	67.5

[F. Rozar et al.]

MAISON DE LA SIMULATION

Handling I/O

Diagnostics



Diagnostics: **Numerical counterpart to probes in reactors** (0D...3D values)

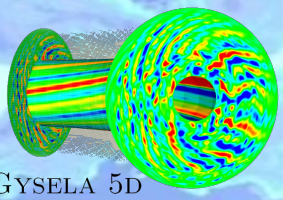
- Current implementation
  1. Pre-process in parallel
  2. Reduce all data on one node per diagnostic ($\approx$ 6 in code)
  3. Post-process and write to HDF5 file
- Limitations
  - Each **data has to fit in 1 node** memory (problem for 3D)
  - **Computation stopped** for few nodes computation & I/O
- Development of the **LCHD library** performed by HLST-IPP Garching
  - Parallelization of 3D diagnostics (domain decomposition)
  - Fast dedicated file format
  - Data compression
  - Post-processing script for compatibility
- **I/O bandwidth** $\times 26$ with **parallel efficiency of** 95% (256 $\rightarrow$ 1280 cores)
  - ➠ Lossless: 8% compression
  - ➠ Lossy: from 50% to 70% achieved without altering physics

[S. Espinoza, HLST Report 2013]

MAISON DE LA SIMULATION

Handling I/O

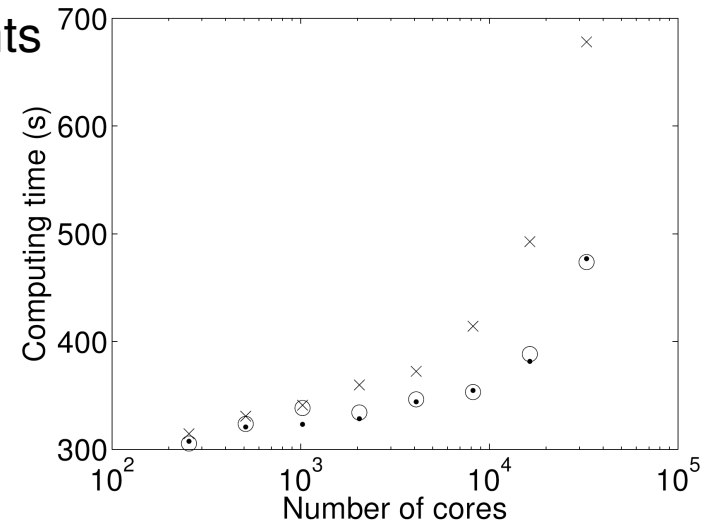
Checkpointing



- Checkpoints
  - Saved every ≈ 10h
  - Full 5D ion distribution + 3D fields saved, 1 file/node
- ↗ core count
  - ↗ failure probability, more frequent checkpoints
  - Fixed PFS BW, bad scaling property
- New implementation
  - copy data to memory
  - restart computation
  - Dedicated thread copy to PFS

☺ ⇒ Overlap computation & I/O

☹ ⇒ Memory overhead



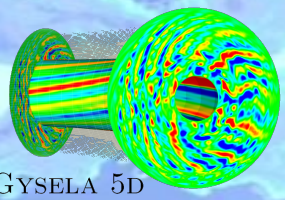
Synchronous I/O (cross)
vs. Asynchronous I/O (circles)
and no I/O (points)

[O. Thomine et al., ESAIM proceedings 2013]

MAISON DE LA SIMULATION

Handling I/O

Checkpointing



- Next step : FTI a Fault Tolerance Interface library
  - Take advantage of node-local SSDs (Cf. Curie)
  - PFS-free checkpoints

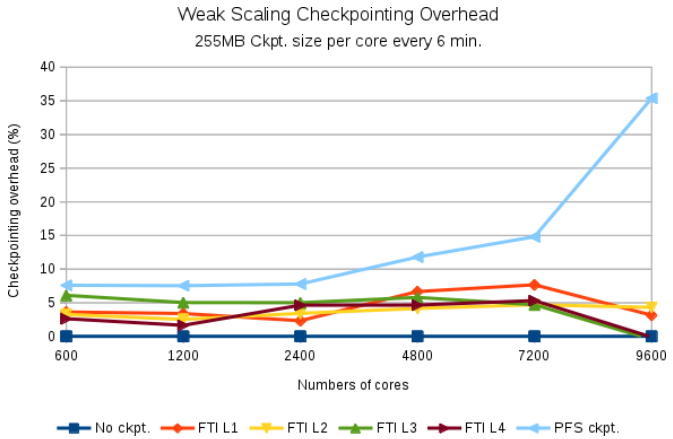
- The approach: Multilevel Checkpointing

Level 1 checkpoint to local SSD only (transient errors)
Level 2 checkpoint to SSD + Exchange with neighbor (node-local failure)
Level 3 checkpoint to SSD + Group Reed-Solomon encoding (limited failure)
Level 3 checkpoint to SSD + Asynchronous copy to PFS (catastrophic failure)

- Cheaper checkpoints, better scaling
- But do not handle all kind of failures

  ⇒ a combination of each level required

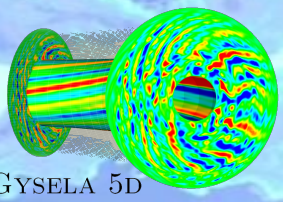
- Full scale evaluation in progress



[J. Bigot, L. Bautista et al.]

MAISON DE LA SIMULATION

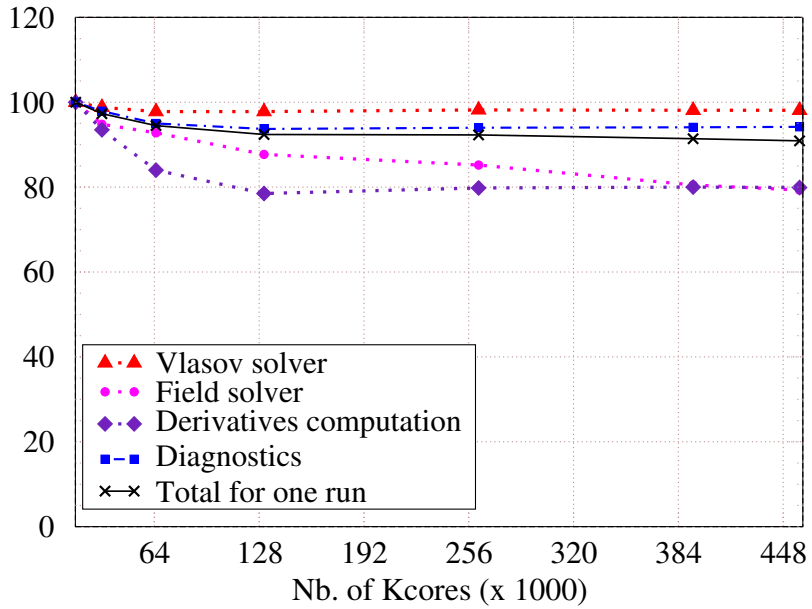
Performance Scaling



Weak scaling: $N_r = 512,\ N_\theta = 1024,\ N_\varphi = 128,\ N_{v\|} = 128,\ N_\mu = [2-56]$

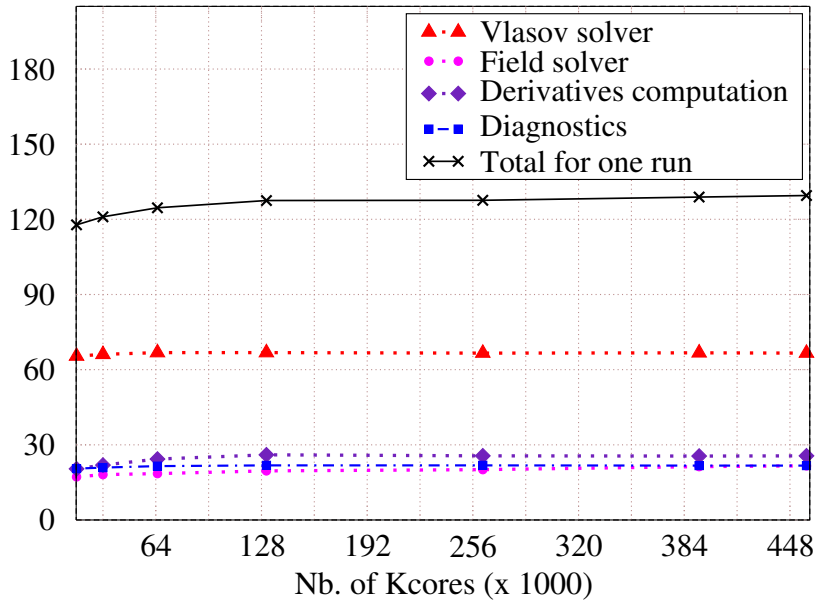
- 16 → 448 ki cores, 65 → 1 835 k threads (**Full Juqueen !**)
- No comm compression included

Relative efficiency, one run (Weak scaling - Juqueen)



- **91% efficiency at 458 752 cores**

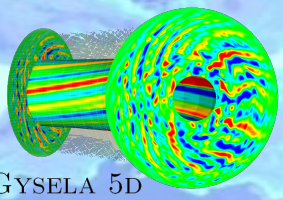
Execution time, one Gysela (Weak Scaling - Juqueen)



- Time still dominated by Vlasov
- Current bottleneck: still Poisson

MAISON DE LA SIMULATION

Work in Progress & Projects

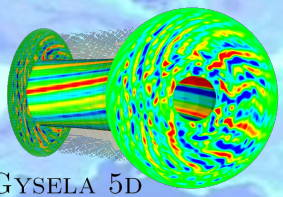


- More I/O optimizations
  - Use of dedicated I/O nodes for diagnostics
    - → Intricate in code ☹
- Node-level optimizations
  - MIC porting
    - → New 4D scheme instead of splitting
  - 3-level parallelism: MPI + OpenMP + vectorization
- Code modularization (component-inspired approach: L^2C, ...)
  - Ease choice of algorithm
  - Ease changes that impact the code structure (Cf. diagnostics)
  - Get ready for more invasive changes
- ...

\+ Lots of new physics !

MAISON DE LA SIMULATION

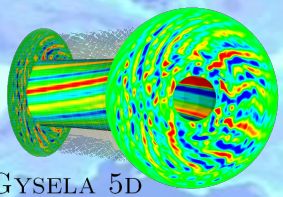
Conclusion



- Going to Exascale is a step by step process
  - Fighting Amdahl's law in a never ending job
- Flops are not the only metric to watch anymore
  - communications, memory, IOs, ...
- Reusable libraries for peripheral optimizations:
  - Communications compression (GCL)
  - Memory scalability (MTM)
  - Diagnostics IOs (LCHD)
  - Checkpointing (FTI)
- Other optimizations not reusable yet
  - Communication scheme optimizations, Vectorization optimizations, ...
  - **Programming models have to be improved**
- Parallelization has to be taken into accound at every level:
  - physics, maths, computer science

MAISON DE LA SIMULATION

Conclusion



- Going to Exascale is a step by step process
  - Fighting Amdahl's law in a never ending job
- Flops are not the only metric to watch anymore
  - communications, memory, IOs, ...
- Reusable libraries for peripheral optimizations:
  - Communications compression (GCL)
  - Memory scalability (MTM)
  - Diagnostics IOs (LCHD)
  - Checkpointing (FTI)
- Other optimizations not reusable yet
  - Communication scheme optimizations, Vectorization optimizations, ...
  - **Programming models have to be improved**
- Parallelization has to be taken into accound at every level:
  - physics, maths, computer science
  - **Luckily, MdlS is here** ☺