DE LA RECHERCHE À L'INDUSTRIE

High-performance tools to model instabilities in tokamak plasmas

JOREK & GYSELA

JOREK: M. Becoulet, G. Dif-Pradalier, A. Fil, V. Grandgirard, G. Latu, E. Nardon, F. Orain, C. Passeron, A. Ratnani, C. Reux

Collaborations with: INRIA, IPP, ITER Org., french Universities

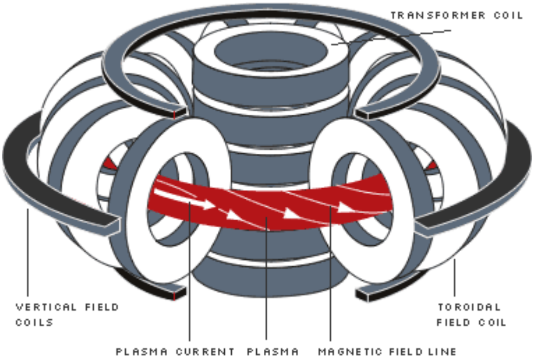
GYSELA: J. Bigot, G. Dif-Pradalier, D. Estève, V. Grandgirard, X. Garbet, Ph. Ghendrih, G. Latu, C. Passeron, Y. Sarazin, F. Rozar

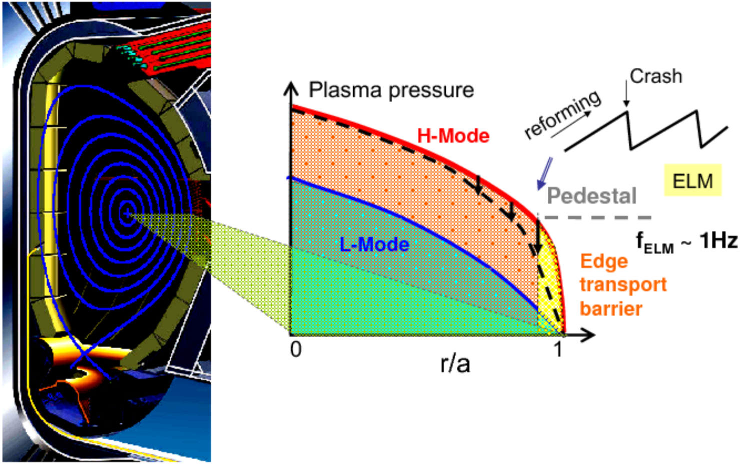
Collaborations with: INRIA, IPP, french Universities

www.cea.fr

Outline

- JOREK code (modelling non linear MHD)
  - Context, Results & work in progress
- GYSELA code (modelling transport & ITG turbulence)
  - Context, Results & work in progress
- Perspectives



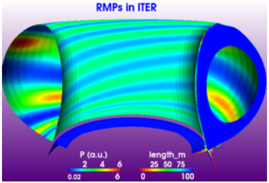


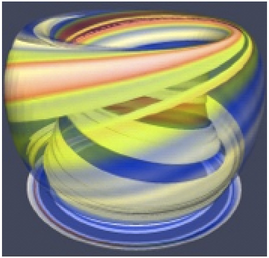
Extracted from [Liang Yunfeng, 2010]

JOREK: main characteristics

- Physics: ELM cycle & control, Disruptions
  - ELMs *[G. Dif-Pradalier, M. Bécoulet, S. Pamela]*
  - Resonant Magnetic Perturbations (RMPs) *[M. Bécoulet, F. Orain]*
  - Pellets injection *[G. Huijsmans, S. Futatani]*
  - VDE, β limit disruptions, density limit *[C. Reux, E. Nardon, A. Fil]*
- Realistic geometry (X-point)
  - Cubic finite elements, flux aligned **poloidal** grid
  - Fourier series in **toroidal** direction
  - Fluid description (3D), 6 to 8 variables per grid point
  - Numerical scheme: solve a large sparse linear system
- Challenges to improve handled Physics
  - exact geometry** & boundary conditions**
  - non-linear MHD equ. in toroidal geometry over long time scales* ($\mu s \rightarrow s$)
  - realistic physical variables*** [resistivity, parallel conductivity, collisionality]
  - large number of n–modes***, coupling with background turbulence$^{?}$

JOREK: context, issues



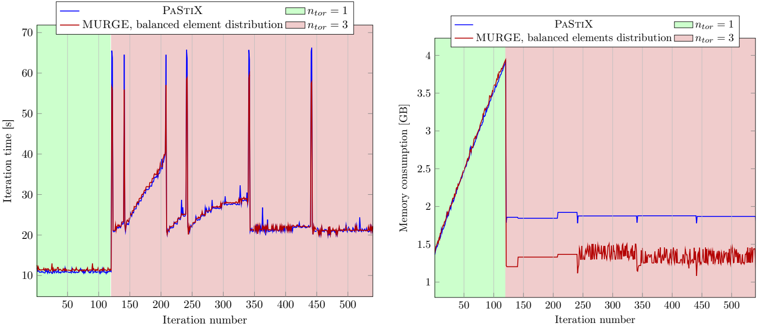


- Physics studies:
  Production code for *non-linear MHD*
  use Tier-2, Tier-1, Tier-0 facilities
- Mathematical issues:
  → Mesh, robustness, convergence
  → large costs (memory, computation)
- Parallel computing issues:
  → Rely on sparse linear solver
  depends on performance: PaStiX, MUMPS
  → Save memory space (larger cases)
- Collaborative issues:
  → Modify a single code, ensure correct results

JOREK: Work in progress (prod. runs 512 cores, test runs 1500 cores)

- Physics: CEA, ITER, IPP (Germany), JET(UK), ...
  Eurofusion funding (H2020)
- Mathematical bottlenecks: **convergence**, **large cases**
  - → IRFM, IPP Garching, INRIA TONUS, INRIA CASTOR:
    preconditioner, time scheme (convergence)
    other finite elem. (robustness/accuracy/stability)
    isogeometric analysis (reduced costs)
- Parallel computing bottlenecks: **large cases, new arch.**
  - → INRIA HIEPACS + IRFM (ANR ANEMOS):
    improve coupling Jorek/PaStiX, save memory space
  - → IPP + IRFM
    porting matrix construction on Xeon Phi
- Parallel code issues: **maintain a healthy code**
  - → automatize checks, regression tests on parallel machines

- **Aim: Reduce memory peak, improve mem. scalability**
  - Pb: Memory required to centralize/redistribute matrix
  - Solution: distributed API Murge (reduce memory peak)
  - Adapting/Improving Jorek&Murge [⇝ Hiepacs, X. Lacoste]



(a) Iteration time. (b) Memory consumption.

Figure : Runs of Jorek model 303, on 8 nodes of 8 cores (curie)

Jorek, Result 2: Strong scaling with MPI_THREAD_MULTIPLE - MdlS

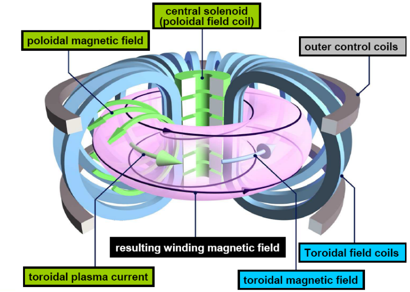
- Simulation model 302 - (ntor=15,nflux=32,ntheta=48) using MPI_THREAD_MULTIPLE mode:

Nb cores	**128**	**256**	**512**	**1024**
Nb nodes	8	16	32	64
Steps				
`construct_matrix`	15.3	7.7	4.1	2.2
`factorisation`	0.	0.	0.	0.
`gmres/solve`	3.0	1.7	1.2	0.86
`iteration time`	18.6	9.7	5.6	3.4
`rel. efficiency`	100%	96%	83%	68%

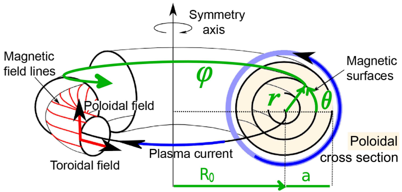
Table : one iteration - with no Factorization

- Good result: 68% rel. efficiency (whole code) at 1024 cores
- Pb: MPI_THREAD_MULTIPLE is not available on many machines
  $\longrightarrow$ Exec time **multiplied** by a factor 2 to 4

Turbulence governs plasma performance

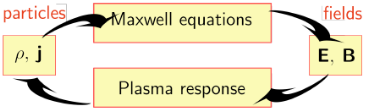


magnetic toroidal geometry (r, θ, φ)



- Turbulence limits the maximal value reachable for *n* and *T*
  - Generates loss of heat and particles
  - ↘ Confinement properties of the magnetic configuration
- To predict and control turbulence for optimizing experiments like ITER and future reactors is a subject of utmost importance.

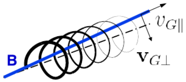
GYSELA code - context
Gyrokinetic theory



- Kinetic approach: 6D distribution function of particles $f(r, \theta, \varphi, v_{\|}, v_{\perp}, \alpha)$
- Gyrokinetic codes:
  - fusion plasma turbulence is low frequency

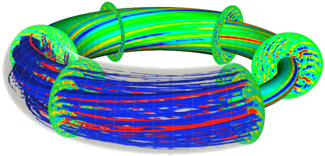
    $\omega_{\text{turb}} \sim 10^5 s^{-1} \ll \omega_{ci} \sim 10^8 s^{-1}$

  $\hookrightarrow$ Reduced to 5D distrib. function $f(r, \theta, \varphi, v_{\|}, \mu)$



☺ Reduced memory needs but ☹ increased complexity

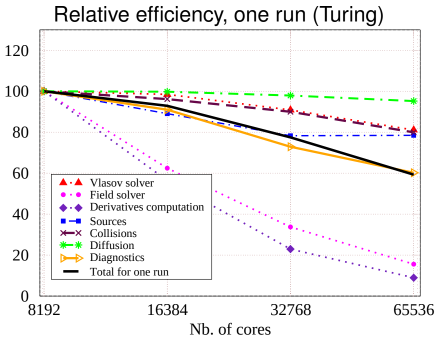
- Main features in GYSELA (Gyrokinetic Semi-Lagragian code):
  - Main equations: Vlasov 5D, Poisson 3D (quasineutrality)
  - Gyrokinetic setting (5D = 3D space + 2D velocity)
  - ITG driven turbulence
  - Heat & vorticity sources, flux driven
  - Collisional operator
  - Modelling fast particles
  - Adiabatic electron response



Result 3A: Scalability and bottlenecks

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

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

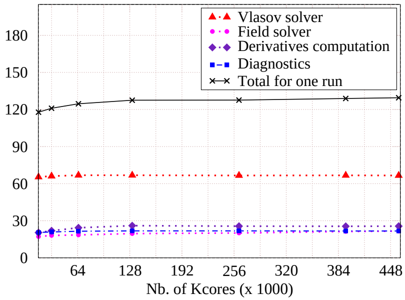


- Time dominated by Vlasov solver
- Bottleneck at large scale: Poisson solver, IO
  $\approx 60\%$ efficiency at 64 k cores on machines: Curie and Turing

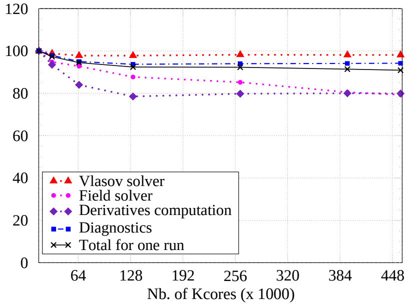
Result 3B: Weak scaling on Juqueen

- Many communication schemes rewritten (hierarchical gather/scatter)
- Tests performed on the whole Juqueen/Blue Gene machine (Juelich)

Execution time, one Gysela (Weak Scaling - Juqueen)



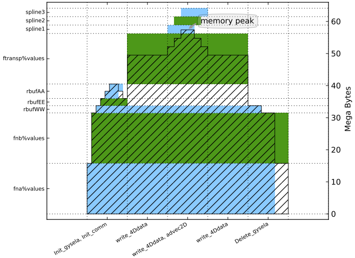
Relative efficiency, one run (Weak scaling - Juqueen)



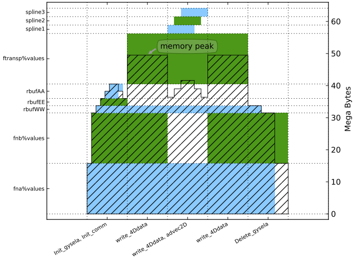
- Weak scaling: Relative efficiency of 91% on 458 752 cores (2013)
  - PRACE preparatory access *(April 2012 - Nov 2012)*: 250 000 hours
  - Extra CPU allocation *(via P. Gibbon in 2013 / ANR G8-Exascale)*

Gysela
Result 4A: Model memory consumption

- GYSELA is global ➠ Huge meshes ➠ Constrained by memory per node
- Development of the **MTM library** (Modelization & Tracing Memory consumption)
  - Identify memory peak, look at memory scalability



Before optimisation



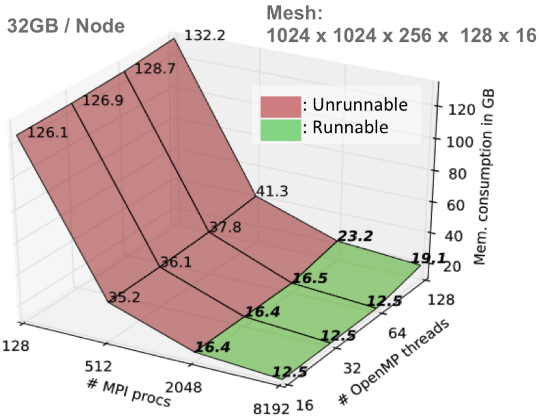
After optimisation

- Static to dynamic memory alloc. + improvement of algorithms

  ➠ **Gain of factor** 50% on 32*k* cores

Result 4B: Predict memory consumption

- Thanks to **MTM library** (Modelization & Tracing Memory consumption)
  - (a) Instrument your code with **MTM**, run a small case
  - (b) Run a *virtual* scaling (predict mem. required before submit)
  - (c) Never run out of memory !

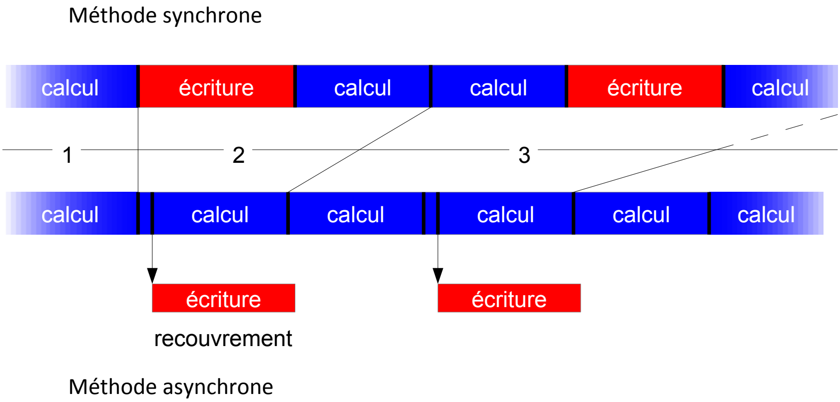


- Context
  - Number of cores in supercomputers increases very fast
  - Fault tolerance needed for long-running, large-scale codes
  - Current options: replication, or checkpoint/restart
- Cheaper checkpoints, reduced exec. time ➠ *actual* pb
- FTI: Fault Tolerance Interface (*INRIA & Argonne*)
  - register data part of the application state
  - notify the library when in a consistent state
- FTI strategy: take advantage of node-local SSDs (curie)

Level 1 checkpoint to local SSD only (transient errors)
Level ... ...
Level 4 checkpoint to SSD + Asynchronous copy to PFS

Result 5A: Asynchronous method

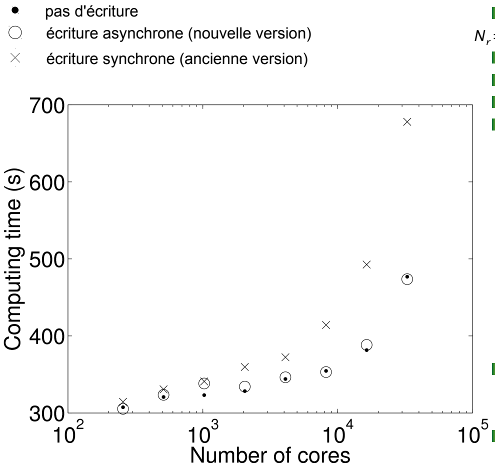
- Checkpoint writing is overlaped by computations



- FTI uses also asynchronous writing

Gysela
Result 5B: Weak scaling for checkpointing

- FTI has been integrated in Gysela ⇒ modularization of existing code
- Weak scaling benchs are currently running (FTI) ➠ pb of reproductibility



- Maillage par nœud : $N_r=512, N_\theta=512, N_\varphi=128, N_{v_{//}}=64, N_\mu=1$
- Taille des données : 2,3 To
- Temps d'écriture : ~ 1 minute
- Écriture toutes les 4 itérations
- Simulation : 10 itérations
- Temps calcul utilisé : 100 000 heures
- Helios machine (Rokkasho)

Pb 6: Reliability of parallel programs

- Should we trust our parallel codes ?
  - Many parallel computations, undeterminism of execution
  - Many possible input parameter sets, many parallel systems
  - Bugs introduced by developers
- Solution (using `http://ci.inria.fr`)
  - At every commit: compilation test
    - Compile every variant of the code
    - On multiple compilers/architectures
    - On multiple supercomputers
  - At every commit: launch cases (*work-in-progress*)
    - Run multiple test-cases on multiple supercomputers
    - Compare results with stored reference
  - Regularly: large-scale validation tests (*TODO*)
    - Run large test-cases
    - Analyze parallel performance
    - Validate results with known physical reference

- Improving Jorek parallel performances
  (INRIA HIEPACS)
- New preconditioners, time integration scheme for Jorek
  (INRIA TONUS, CASTOR)
- Parallelization of a new gyroaverage operator in Gysela
  (INRIA TONUS, HIEPACS)
- Software components and StarPU approaches for Gysela
  (INRIA AVALON, RUNTIME)
- Thread & mem. affinity for deploying complex parallel codes
  (INRIA HIEPACS, RUNTIME)
- Designing parallel kernels for 6D Vlasov equation
  (INRIA KALIFFE)
- Porting and developing new Gysela kernels for Xeon Phi
  (Japan collaboration, INTEL Exascale labs)
- Improved parallel I/O ...