Performance of ARPEGE and AROME/ALADIN

Philippe Marguinaud CNRM/GMAP

- Our model
- Scalability issues
- GPUs
- ESCAPE & HPC market
- Possible improvements

My experience

- Scalability is a misleading concept
- The model only needs to be fast enough to deliver a forecast
- Performance = algorithms x implementation x optimisation
- The simplest method is most often the best

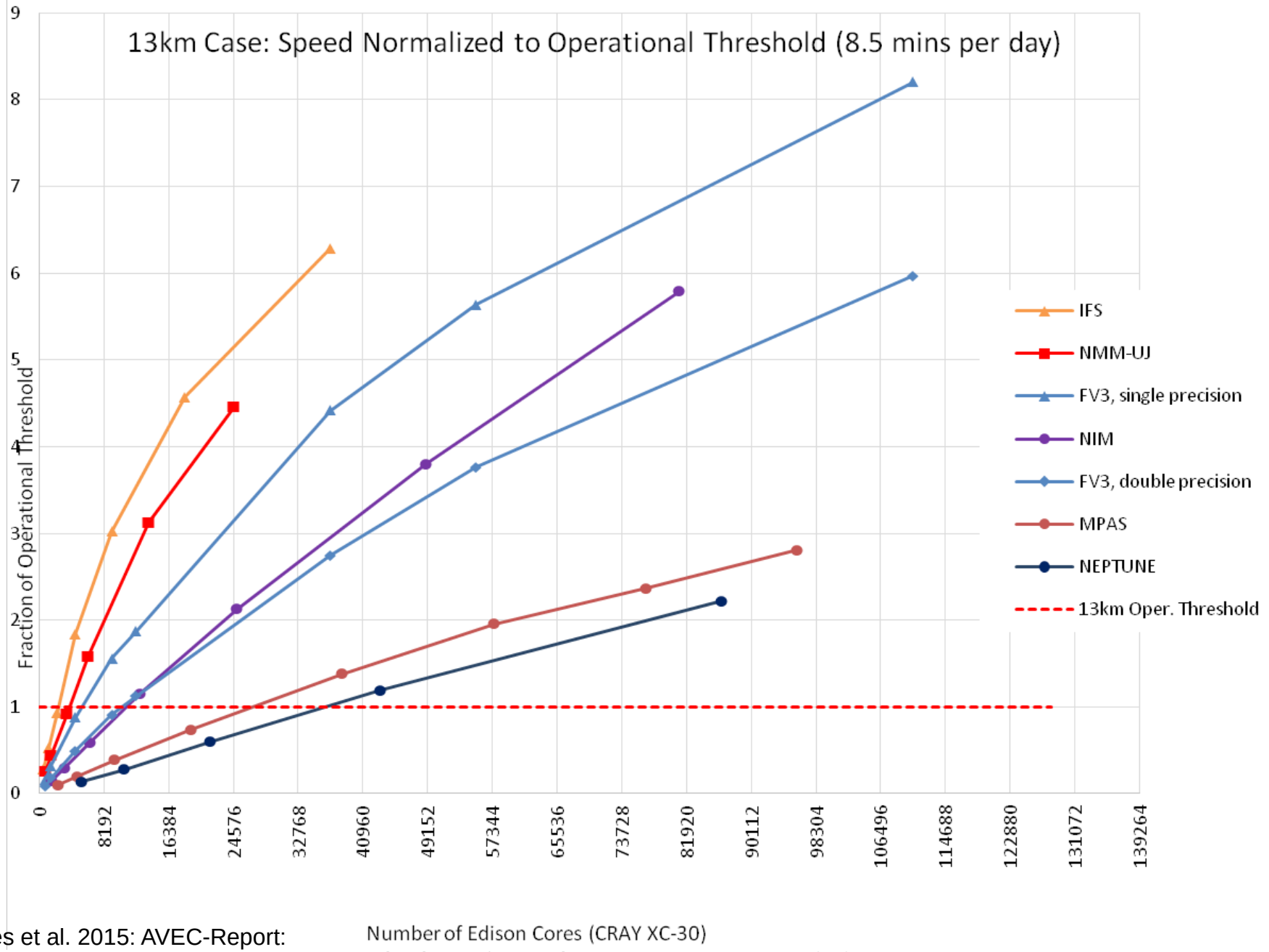
Our model

- Semi-Implicit + Semi-Lagrangian with the help of spectral transforms, allowing large time-steps
- Non scalable parts of the model integration = Semi-Lagrangian + Spectral transforms
- The model integration is a **sequential** process

  → Impossible to distribute/parallelize along time dimension

  → The large time-step approach looks very interesting

Comparison with other models



[Michalakes et al. 2015: AVEC-Report: NGGPS level-1 benchmarks and software evaluation]

But...

Performance = Algorithms x Code design x Optimization

- Who designed the code and optimized it ?
- How much effort was spent into adapting it to the software and hardware environments ?

→ We are not comparing algorithms, but software packages

Making the model more scalable

- G. Mozdzynski showed that running the radiation scheme outside the model improves scalability

  → Cut the model in two halves : run the physics separately from the dynamics ?

- Remove all unnecessary stuff from the main time-stepping data-flow (IO, post-processing, DDH, etc…)
- Port to GPUs (less MPI tasks would be necessary)
- Increase resolution moderately

A performance wall ?

- Explicit time-stepping grid-point models have no global communications **but** much smaller (ten times) time-steps
- A grid-point model with higher time-steps is possible **but** it requires at least one Helmholtz equation solver (iterative or spectral) for fast waves
- A radiation calculation server makes the model more scalable, **but** the model does not like the radiation to be run asynchronously.
- Weak constraint 4DVAR more scalable **but** convergence is not so fast
- 4D-EnVar is more scalable, **but** generates a lot of IOs, and still requires a minimizer or a solver

We may be attacking something very powerful

Wall time ↔ Energy

GPUs

- The programming model is still evolving:

  CUDA → OpenCL/ACC/MP

  coherent memory model

- Porting the model as it is today would be very difficult
- The expected gain is still bounded:

  MeteoSwiss has reported an overall total gain of 3 (in terms of energy used, GPUs vs CPUs)

GPUs vs CPUs

	NVIDIA TESLA K80	2-socket INTEL BROADWELL node
Peak (GFlops)	2910	2534
Memory bandwidth (Gb/s)	480	153
Release date	November 2014	June 2015
Power (W)	300	2x135

CPU-bound code → no gain
Memory bound code → gain = 3x at best

What to expect from ESCAPE

- Initiated by ECMWF last year
- Will provide a set of “dwarves“, running on multiple platforms

  → Comparing algorithms should be possible

  → Comparing hardware should be possible
- May provide a framework for writing code on (very) different architectures
- Will provide simulations of dwarves on very high number of nodes

HPC market

- Hardware efficiency is still making progress

  For AROME, 150 Broadwell nodes ↔ 330 Ivybridge nodes

- Today, we use commodity hardware, because specialized hardware (ie vector machines) is not competitive, but this may change
- Intel manufactures the most appropriate processors, but other still exist: IBM (OpenPower), Fujitsu (Sparc64), ARM, NEC, AMD
- NWP is a niche in HPC business, which is a niche in IT business

Possible improvements and advice

- Port the model to simple precision (expected gain = 20 to 30%) and make it the default
- Rewrite AROME/MesoNH physics in Fortran 77, with a few bits of Fortran 90 (expected gain = 10% ?) ; solve the problem of APL_AROME (most costly routine of the physics)
- Keep **only** physics and dynamics inside the model; set aside IO, post-processing, LBC reading and processing, etc...

Possible improvements and advice

- Increase usage of the system:
  - Do not waste time in scripts, copying data, etc...
  - Use advanced tools such as OLIVE, Vortex, mtool
  - Try to exploit NVRAM for downstream applications
- Wait and see what comes out of ESCAPE
- Do not break vectorization !