PP POMPA (WG6)

News and Highlights

Oliver Fuhrer (MeteoSwiss) and the whole POMPA project team

COSMO GM13, Sibiu

Task Overview

Task 1 Performance analysis and documentation

Task 2 Redesign memory layout and data structures

Task 3 Improve current parallelization

Task 4 Parallel I/O

Task 5 Redesign implementation of dynamical core

Task 6 Explore GPU acceleration

Task 7 Implementation documentation

Task 8 Single precision

Task Overview

Task 1 Performance analysis and documentation

Task 2 Redesign memory layout and data structures

Task 3 Improve current parallelization

Task 4 Parallel I/O

Task 5 Redesign implementation of dynamical core

Task 6 Explore GPU acceleration

Task 7 Implementation documentation

Task 8 Single precision

Fundamental question

How to write a model code which…

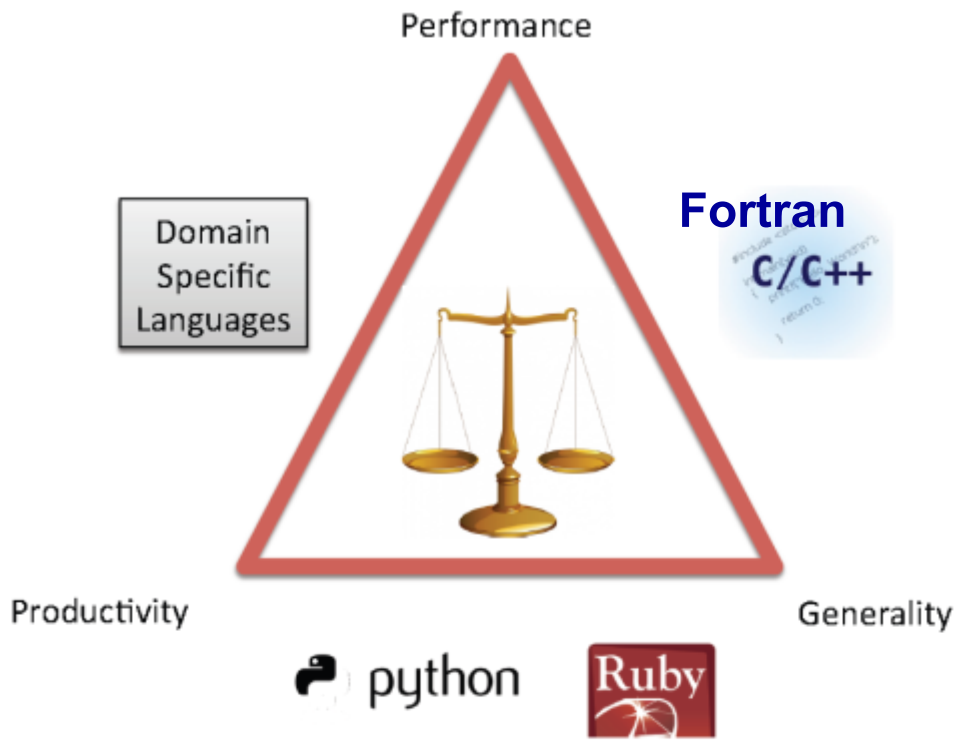
- **allows productive development by domain scientists**
- **runs efficiently on different HPC architectures**
- **continues to do so in the future**

- Clear trend in HPC architectures to become heterogeneous (GPUs, MIC, …)
- Programming models are not getting simpler (OpenMP, OpenACC, OpenCL, CUDA, …)
- Accelerators are a got fit for COSMO

Using a domain-specifc library is one way to solve this problem!

STELLA Library (DSL)

- Separate user code (algorithm) from hardware specific implementation (optimization)



This is a fundamental change for COSMO code with distinct pros / cons

STELLA usage

```
  DO k = 1, ke
!CDIR OUTERUNROLL=4
    DO   j  = 2 , je-1
!CDIR ON_ADB(lap)
!CDIR ON_ADB(s)
      DO i  = 2 , ie-1
        lap (i,j,k)   = s (i+1,j,k) + s (i-1,j,k) - 2.0_ireals*s(i,j,k) &
                      + crlato(j)*(s(i,j+1,k) - s(i,j  ,k))          &
                      - crlatu(j)*(s(i,j  ,k) - s(i,j-1,k))
      ENDDO
    ENDDO
  ENDDO
```

- Remove explicit data structure (i,j,k)
- Remove explicit loops and loop order
- Remove directives (e.g. NEC, OpenMP, …)

STELLA usage

```
__ACC__
static T Do(Context ctx)
{
    ctx[data_out::Center()] = - (T)2.0 * ctx[data_in::Center()]
      + ctx[data_in::At(iplus1)] + ctx[data_in::At(iminus1)]
      + ctx[crlatvo::Center()] * ctx[Call<Delta>::With(jplus1 , data_in::Center())]
      + ctx[crlatvu::Center()] * ctx[Call<Delta>::With(jminus1, data_in::Center())];
}
```

```
// setup the tracer stencil
StencilCompiler::Build(
    stencil_,
    "HorizontalDiffusionTracers",
    dycoreRepository.calculationDomain(),
    StencilConfiguration<Real, HorizontalDiffusionTracersBlockSize>(),
    pack_parameters(
        Param<data_out, cInOut>(data_out_),
        Param<data_in, cIn>(data_in_),
    ),
    concatenate_sweeps(
        define_sweep<cKIncrement>(
            define_stages(
                StencilStage<LapStage, IJRange<cComplete,-2,2,-2,2>,
                                       KRange<FullDomain,0,0> >(),
            )
        )
    )
);
```

Dynamical core based on STELLA

- **Fully functional RK dynamical core** (all features for COSMO-7 and COSMO-2, and some extras…)

- **Easy switch from CPU to GPU**
  - CPU = (k,j,i), OpenMP
  - GPU = (i,j,k), CUDA, software managed caching

- **CPU (Fortran) → CPU (STELLA) = ~1.6 x**
- **CPU (STELLA) → GPU (STELLA) = ~3.3 x**

* CPU = Intel Xeon 2670 GPU = NVIDIA K20x

Recent Developments

- **Functionality and performance improvements** in STELLA (no bug since > 1 year)
- **Support now by Ben Cumming (CSCS)**
- **Additional dycore features**
  - Relaxation (Carlos)
  - Saturation Adjustments (Carlos)
  - Strang splitting for tracer advection (Tobias)
  - Moisture divergence (Xavier)
- **Missing features**
  - Only RK-core considered
  - new FW-solver
  - Several options (PD-advection, advection order, …)
  - …

Documentation & Publications

- Documentation
  - Stencil library (implementation)
  - Communication framework (user guide + implementation)
  - Wrapper (user guide + implementation)
  - Serialization framework (user guide)
  - Style-guide

- Stencil library workshop material (“users guide”)

- See http://hpcforge.org/

- Publications
  - Gysi et al. 2013 (in preparation)

Next Steps…

- **Dynamical core**
  - Implement important missing parts

- **STELLA library**
  - Continuous development
    - Usability
    - Performance
  - Next generation
    - Generalization
    - Block-structured / unstructured grids

- **Integrate into official version**

Task Overview

Task 1 Performance analysis and documentation

Task 2 Redesign memory layout and data structures

Task 3 Improve current parallelization

Task 4 Parallel I/O

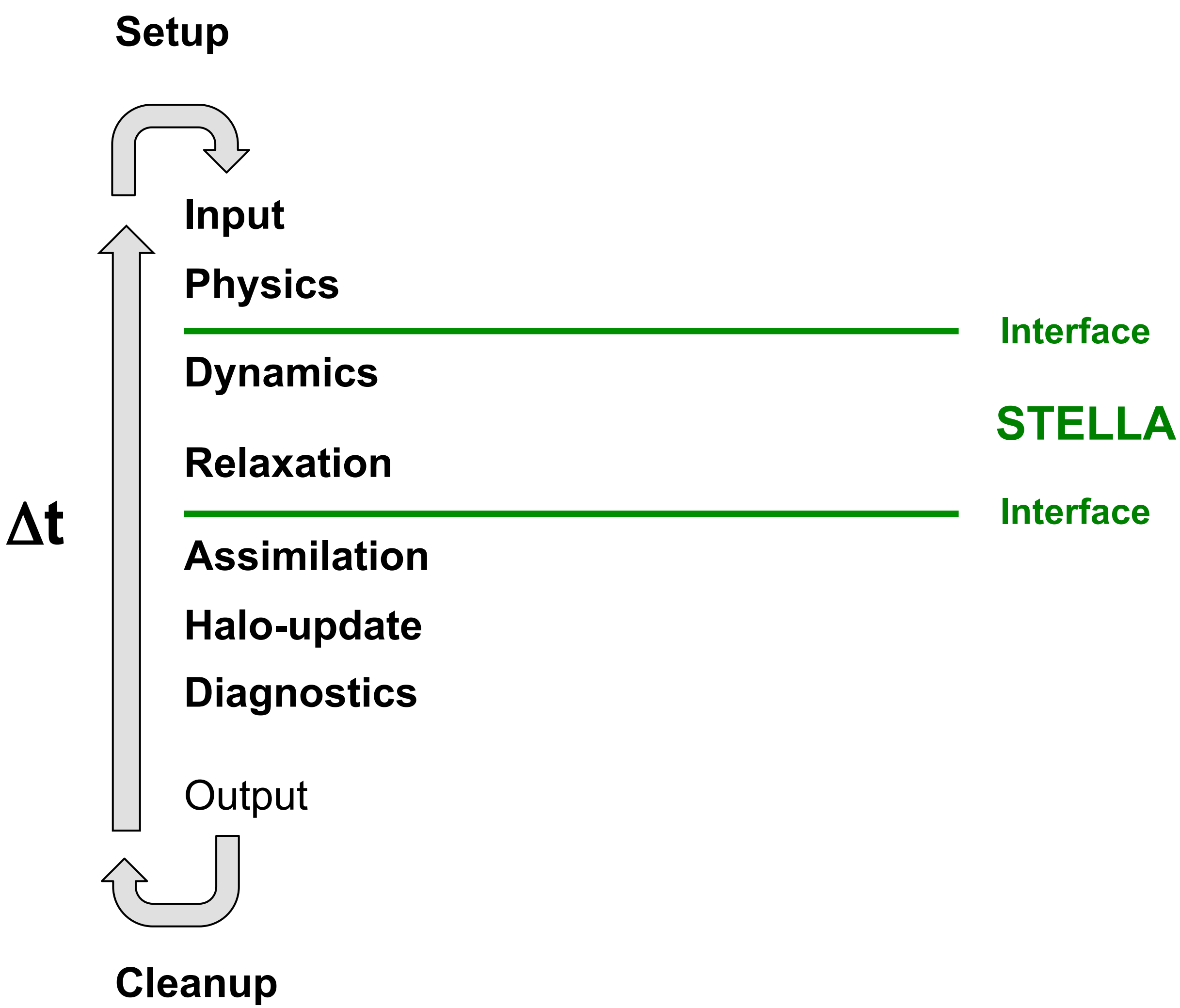
Task 5 Redesign implementation of dynamical core

Task 6 Explore GPU acceleration

Task 7 Implementation documentation

Task 8 Single precision

Implementation



Implementation

Setup

Copy to GPU

Input

Physics

Compiler directives (OpenACC)

Interface

Dynamics

Relaxation

STELLA

Interface

Assimilation

Halo-update

Diagnostics

Compiler directives (OpenACC)

Copy from GPU

Output

Δt

Cleanup

Current status of Physics

Scheme	Status
microphysics - hydci_pp (ice scheme) - hydci_pp_gr (graupel)	 done ready
sub grid scale oro. (sso)	done
radiation	done
turbulence	done
soil model - terra_multlay - terra1 - terra2 - seaice - flake_interface	 done - - - -
convection - conv_tiedtke - organize_conv_kainfri - conv_shallow	 work in progress done

Work by X. Lapillonne, D. Leutwyler, C. Padrin , A. Roches, S. Schaffner

Current status of Physics

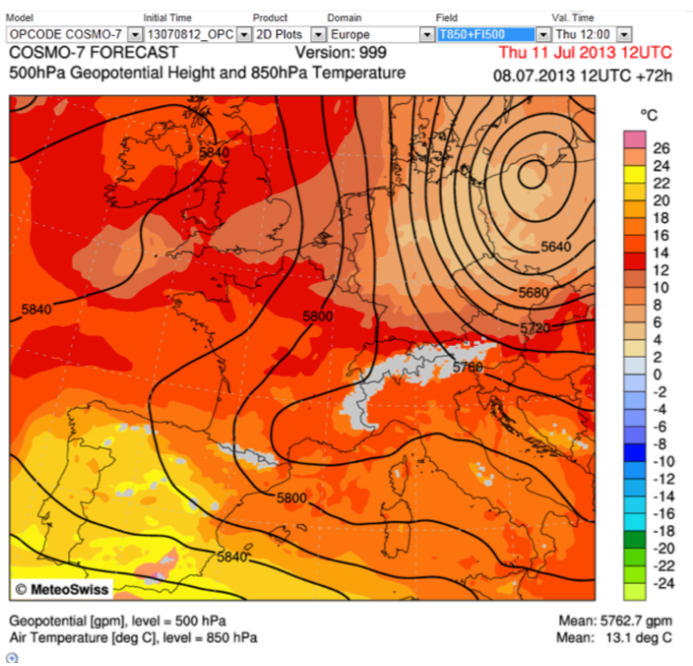
new

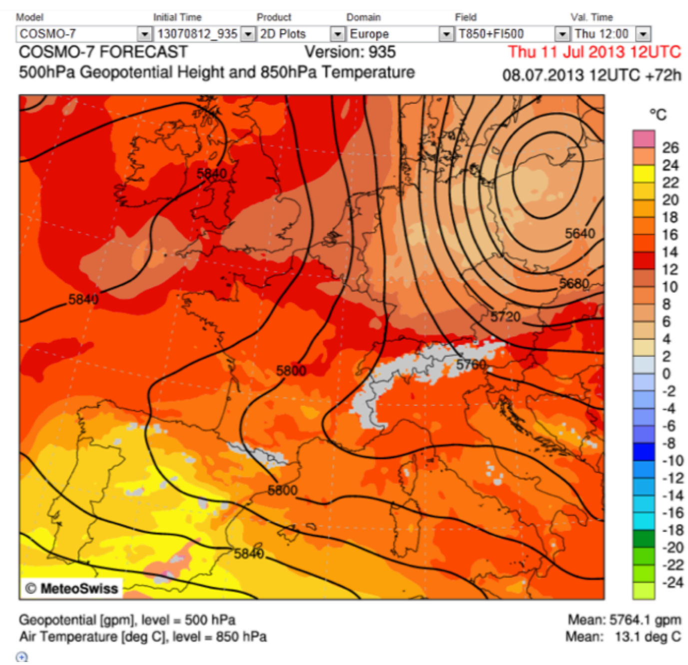
Scheme	Status
microphysics - hydci_pp (ice scheme) - hydci_pp_gr (graupel)	 done ready
sub grid scale oro. (sso)	done
radiation	done
turbulence	done
soil model - terra_multlay - terra1 - terra2 - seaice - flake_interface	 done - - - -
convection - conv_tiedtke - organize_conv_kainfri - conv_shallow	 work in progress done

Work by X. Lapillonne, D. Leutwyler, C. Padrin , A. Roches, S. Schaffner

It works!

- COSMO (v4.19) running on GPU-hardware
- Regular runs
  (00 UTC and 12 UTC of COSMO-7 and COSMO-2)
- Full operational chain
  (plots are delivered into visualization software)
- Almost full featured, but certainly physically reasonable





Overall speedup?

- **Depends on use-case and on hardware** compared
- Benchmark without assimilation and I/O
- Using latest CPU (Intel Xeon 2670) and latest GPU (NVIDIA K20x)

- **CPU (Fortran) → CPU (STELLA) = ~1.3 x**
- **CPU (STELLA) → GPU (STELLA) = ~3 x**

- Larger factor for power savings

Task Overview

Task 1 Performance analysis and documentation

Task 2 Redesign memory layout and data structures

Task 3 Improve current parallelization

Task 4 Parallel I/O

Task 5 Redesign implementation of dynamical core

Task 6 Explore GPU acceleration

Task 7 Implementation documentation

Task 8 Single precision

Motivation & Goal

- Do we really need 15 digits?

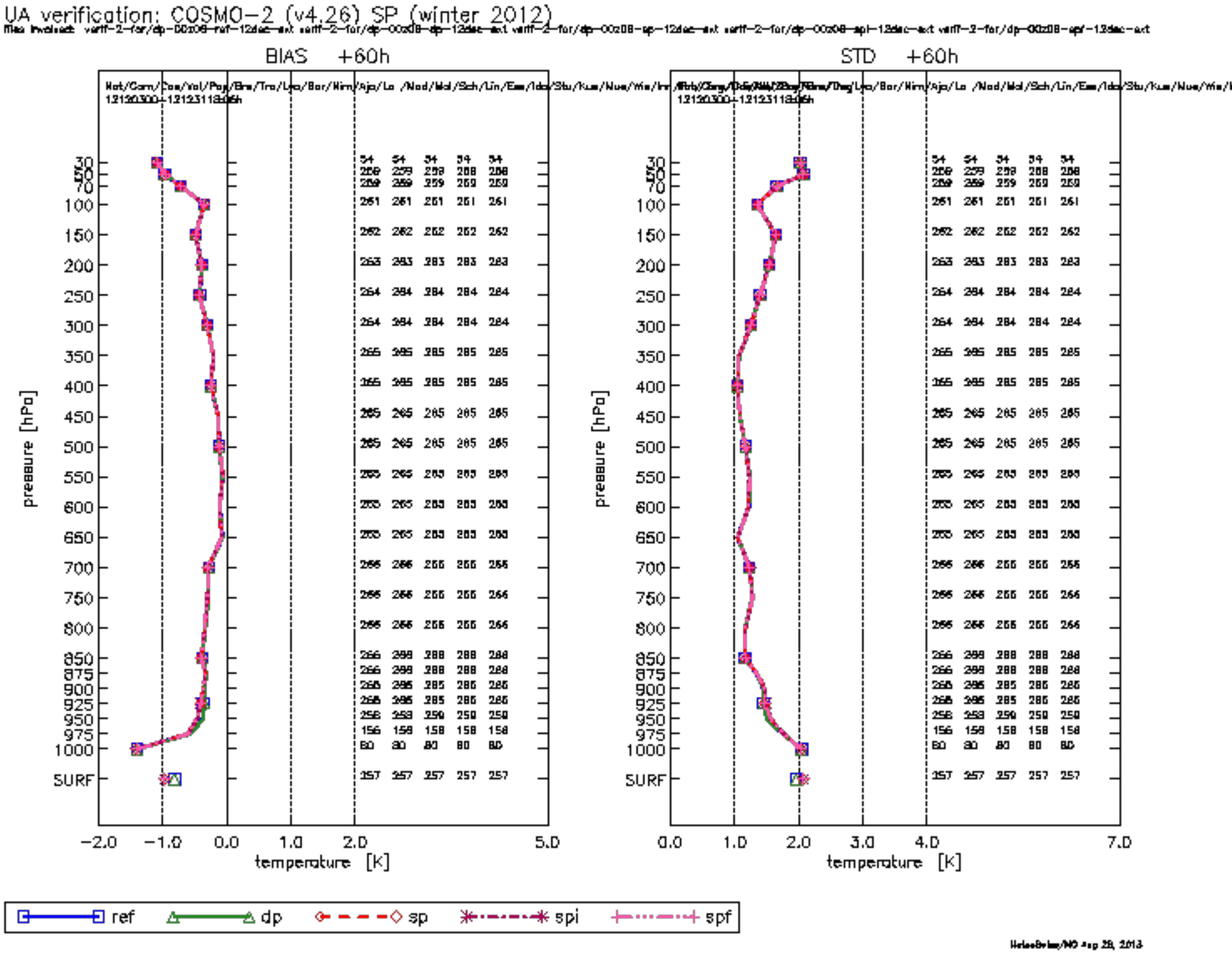
- The advantages of single precision computing:

```
real(kind=8) :: a    ! I am 8 Bytes
real(kind=4) :: b    ! I am 4 Bytes
```

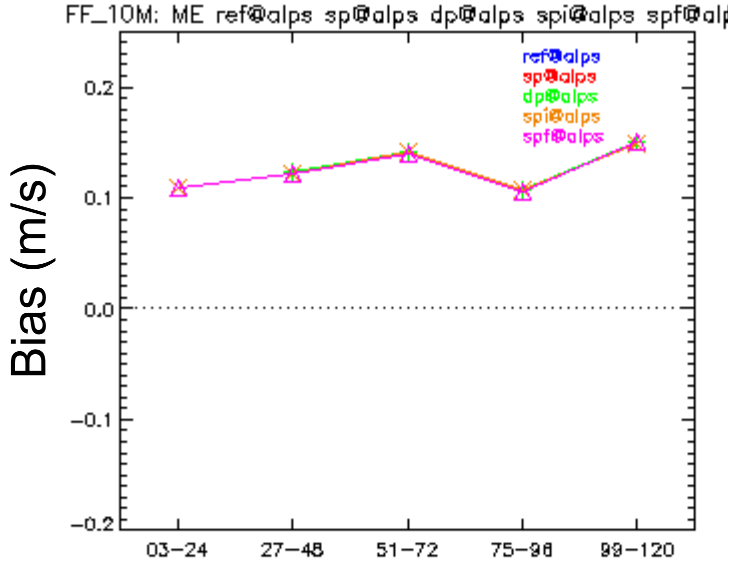
  - Move less information
  - Keep more numbers in cache
  - Lower precision arithmetic is faster

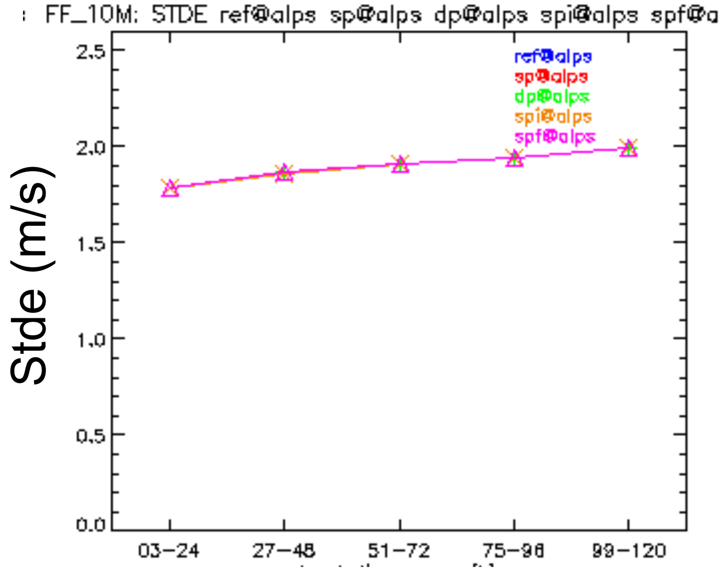
- Goal: one single place in code to define working precision

Upper air verification (+60h, summer)

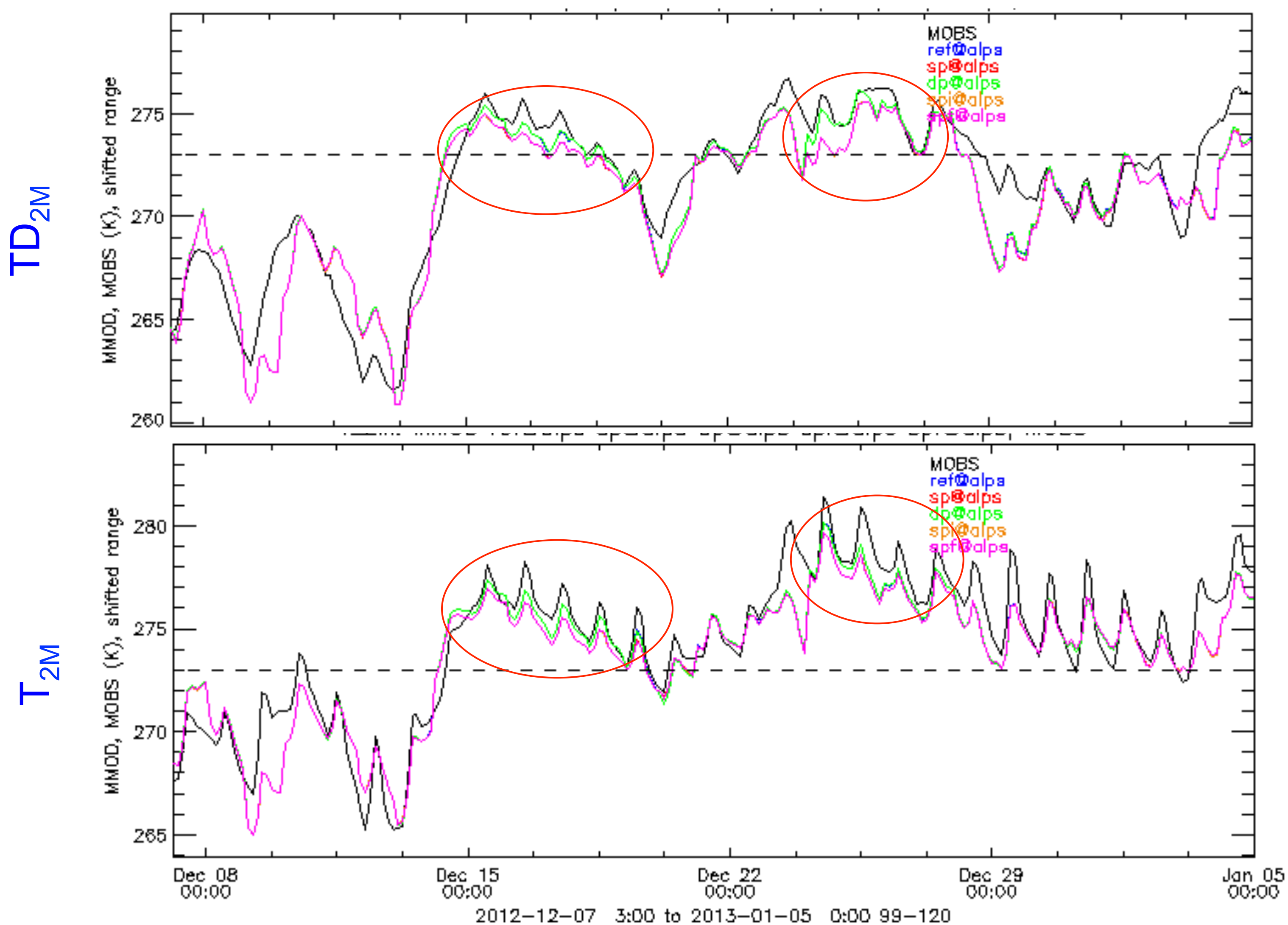


SYNOP Verification (FF10M, summer)





Time-series TD_{2M} & T_{2M}



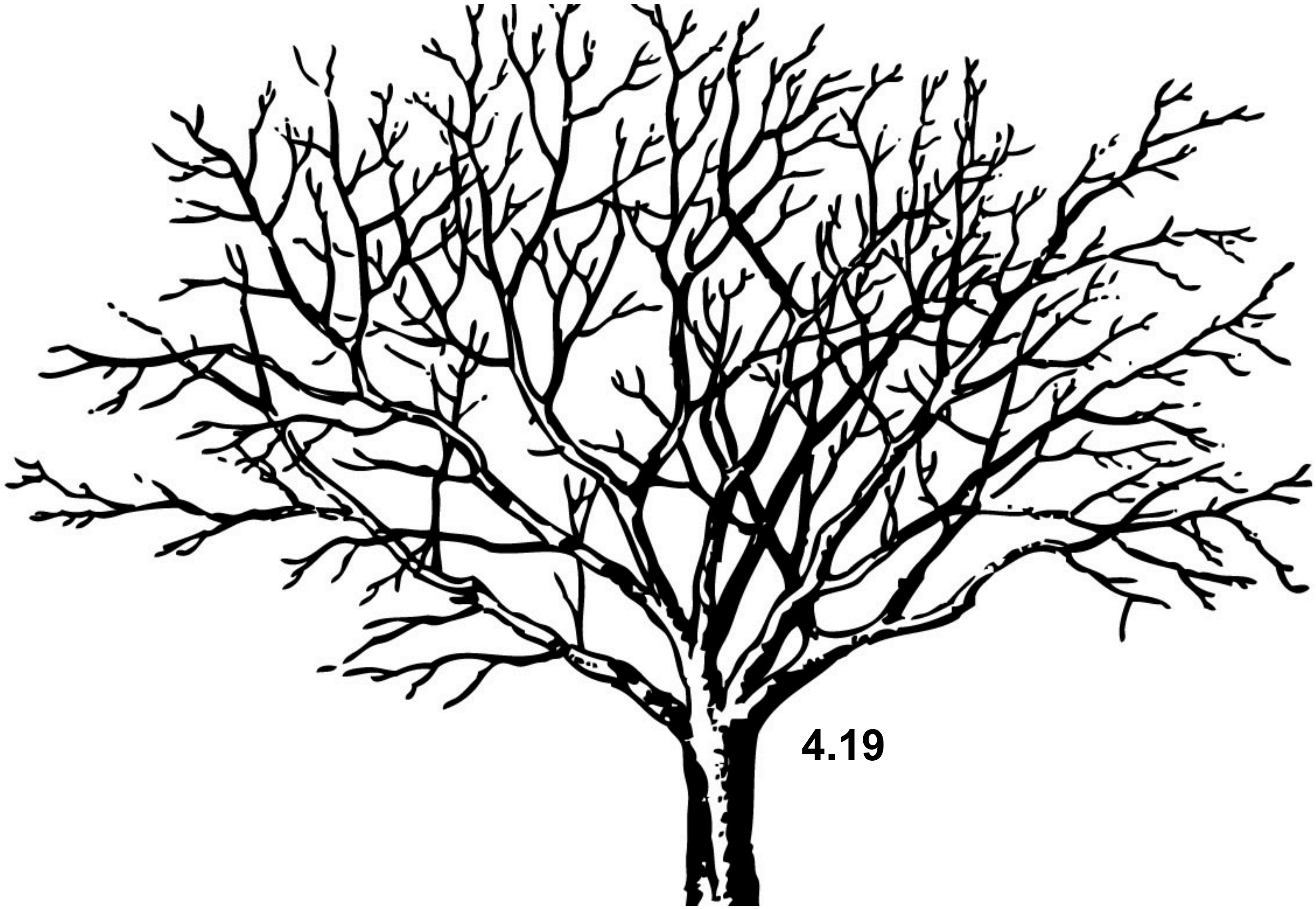
Summary & conclusions

- COSMO version 4.26 with support for user-defined working precision ready for re-integration
- single-precision mode for test purposes
- new code shows same skill with double precision as original code
- marginal degradation in skill found with single precision during a limited period, will be investigated
- reduction of elapsed time to 60% with single precision for COSMO-2 setup!

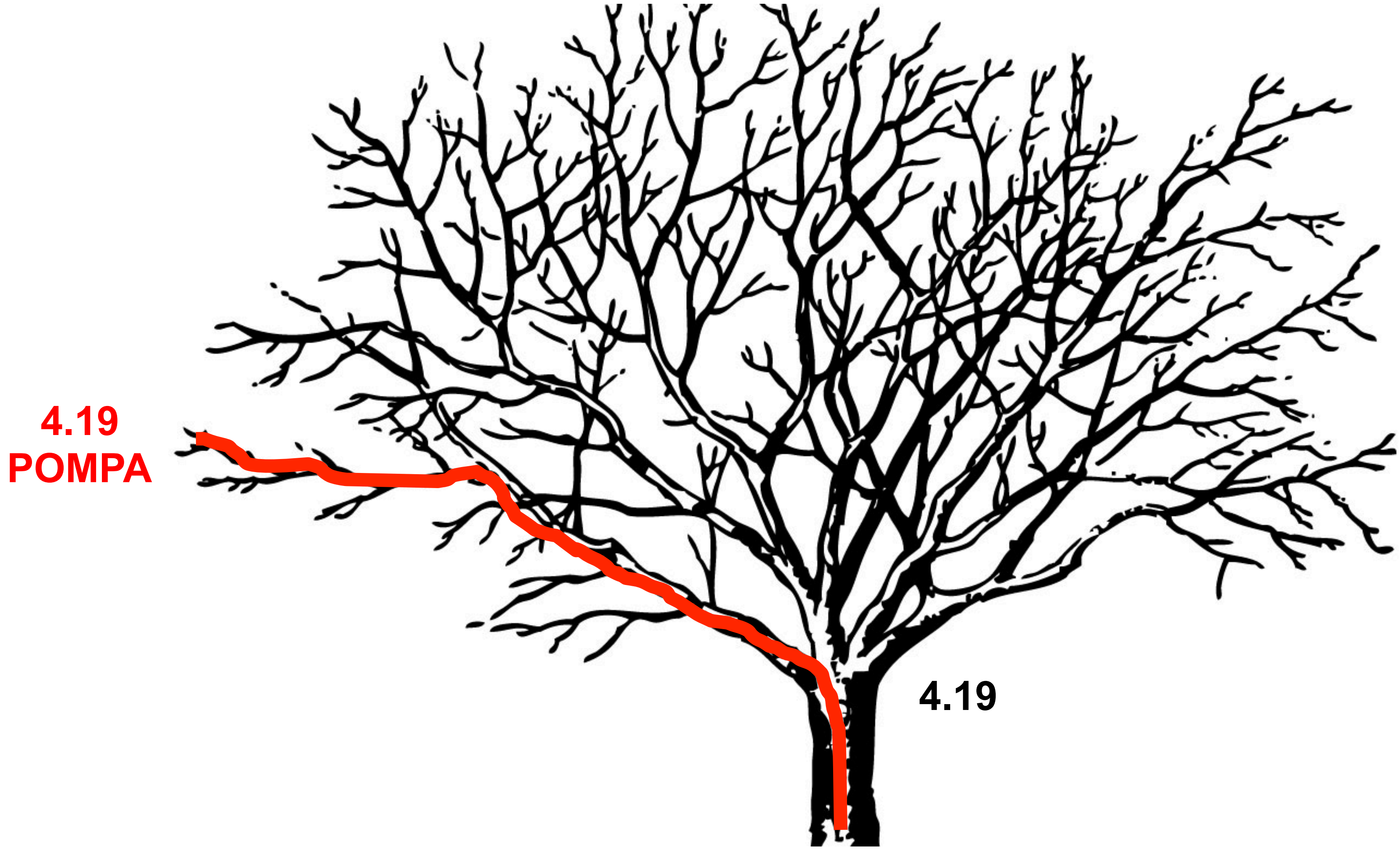
Summary

- There is a version of COSMO which runs on GPUs!
- Getting access to and large allocations on hybrid supercomputers for research projects is easy for early adopters!
- Get involved!

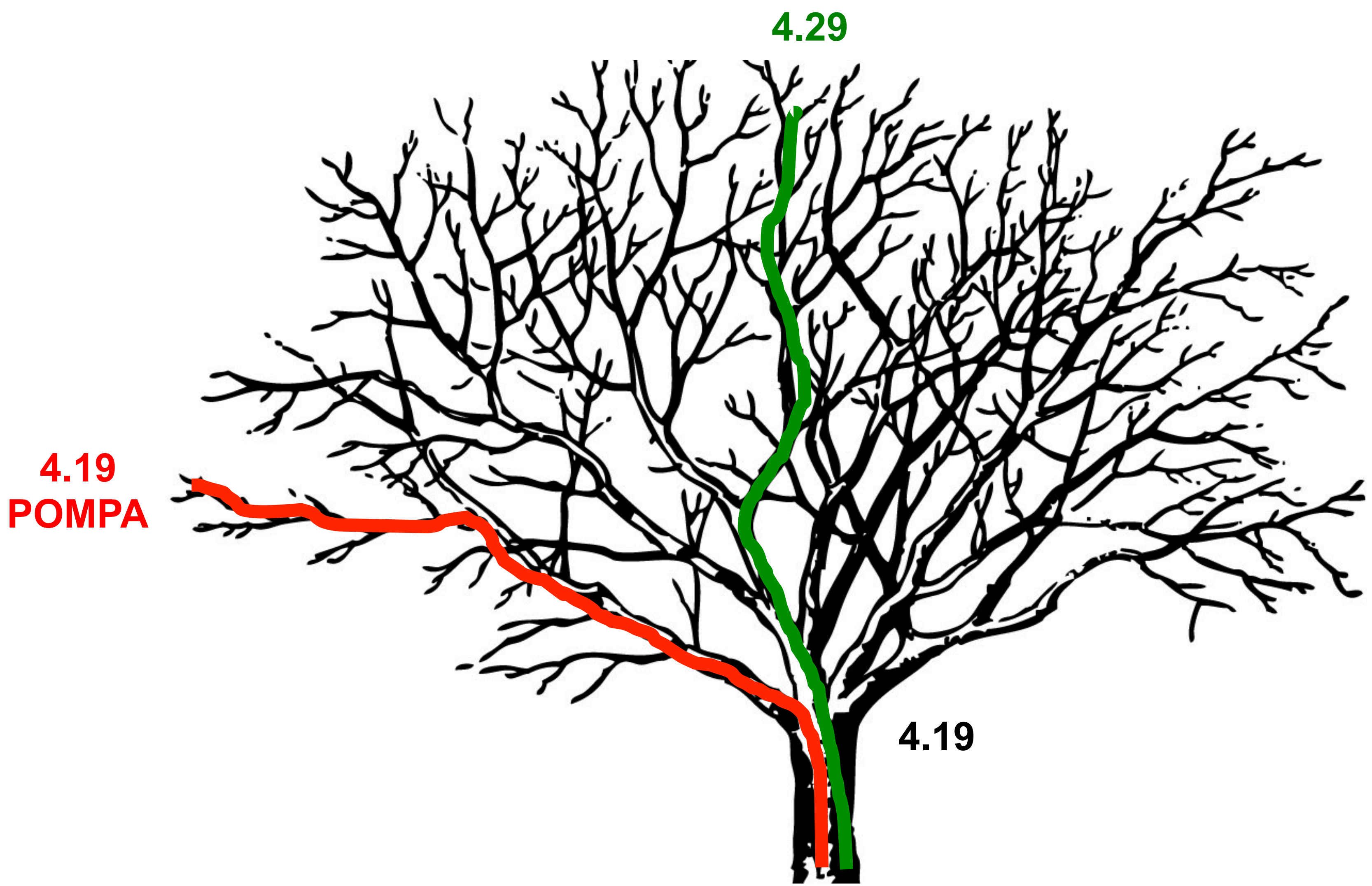
Goal for next COSMO year



Goal for next COSMO year



Goal for next COSMO year



Merge POMPA developments back to trunk until December 2014

Thank you!

…and thanks to the POMPA project team for their work in 2013!

Andre Walser
Andrea Arteaga
Anne Roches
Benjamin Cumming
Carlos Osuna
Cristiano Padrin
Daniel Leuenberger
David Leutwyler
Davide Cesari
Florian Dörfler
Jason Temple
Jean-Guillaume Piccinali
Jeremie Despraz
Joseph Charles
Katharina Riedinger
Kevin Wallimann
Matthew Cordery
Mauro Bianco
Men Muheim
Michael Baldauf
Neil Stringfellow
Nicolo Lardelli
Pablo Fernandez
Peter Messmer
Roberto Ansaloni
Sadaf Alam
Sander Schaffner
Stefan Rüdisühli
Stefano Zampini
Thomas Schulthess
Thomas Schönemeyer
Tim Schröder
Tiziano Diamanti
Tobias Gysi
Ugo Varetto
Ulrich Schättler
William Sawyer
Xavier Lapillonne

/dev/null

Benchmarks

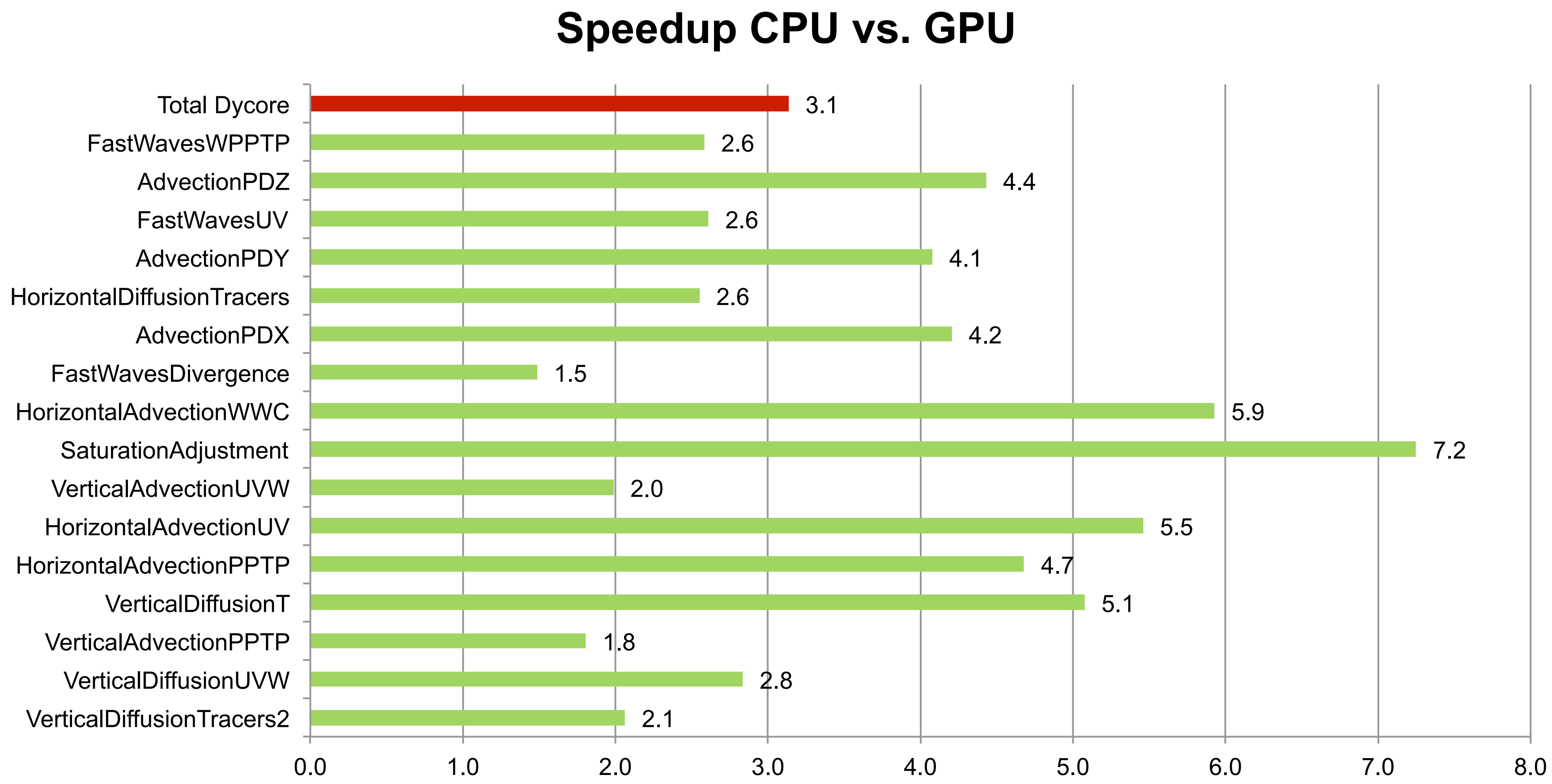
- **CPU performance** was measured on a single socket of Piz Daint
  - Sandy Bridge E5-2670 @ 2.60GHz
  - With hyperthreading

- **GPU performance** was measured on my Windows PC
  - Tesla K20c (roughly 10-20% slower than a K20x)
  - ECC on
  - CUDA 5.5

- Single node measurements on a 128 x 128 data set

Benchmark – CPU vs. GPU

A tesla K20c shows a 3.1x speedup over a Sandy Bridge socket



Approach(es) in POMPA

- How to achieve portable performance while retaining a single source code?

Dynamics
- ~60% of runtime
- few core developers
- many stencils
- very memory intense

Physics + Assimilation
- ~20% of runtime
- more developers
- plug-in / shared code
- "easy" to parallelize

Stencil library (STELLA)

Compiler directives (OpenACC)