Rewrite of the graupel microphysical parameterisation of COSMO using STELLA

Exa2Green
energy-aware numerics

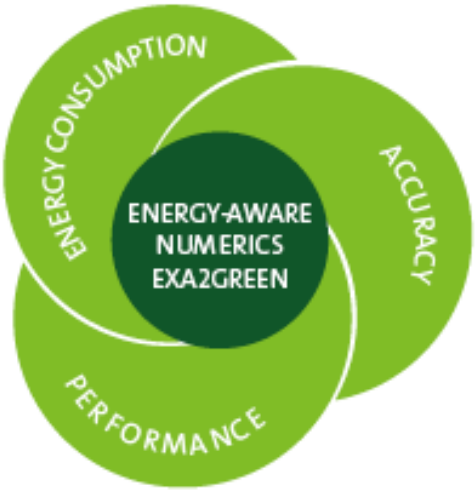
Joseph Charles
COSMO User Workshop
November 27, 2013

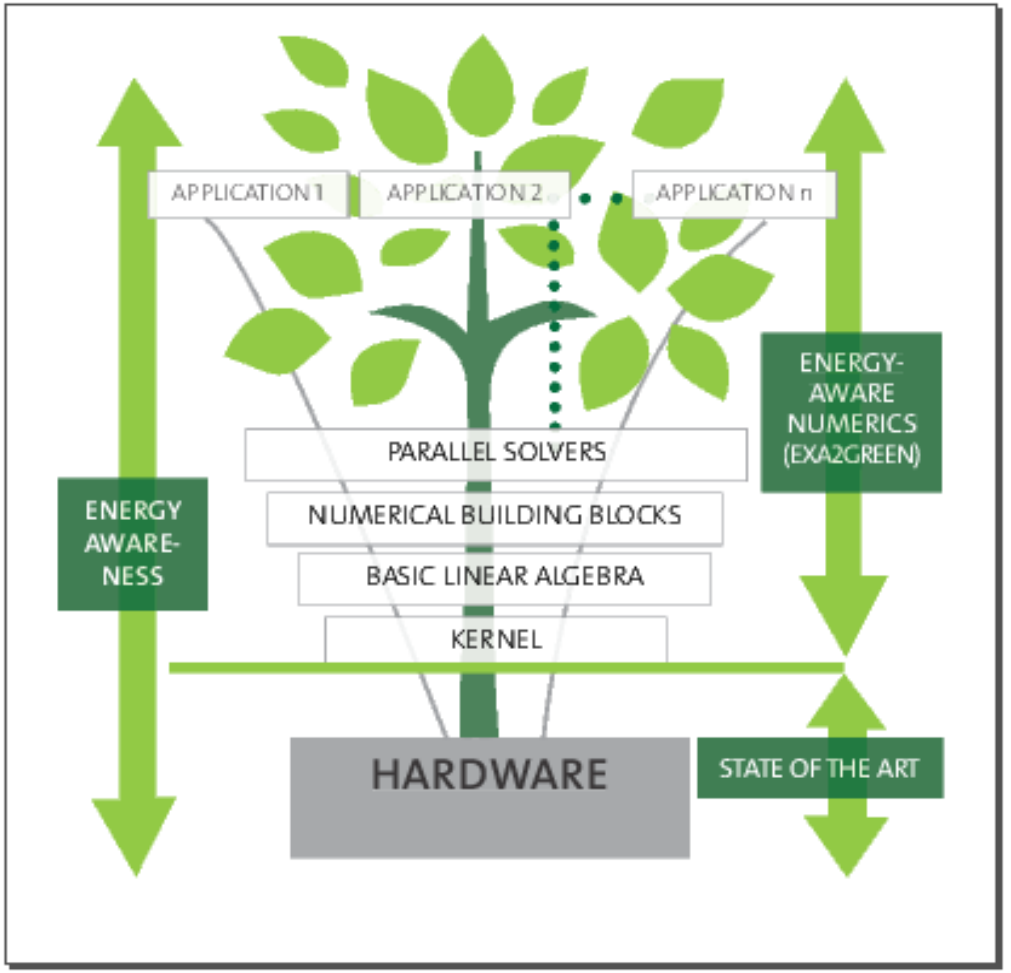
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

EU FP7-funded Exa2Green Project

"Energy-aware Sustainable Computing on Future Technology"





http://exa2green-project.eu

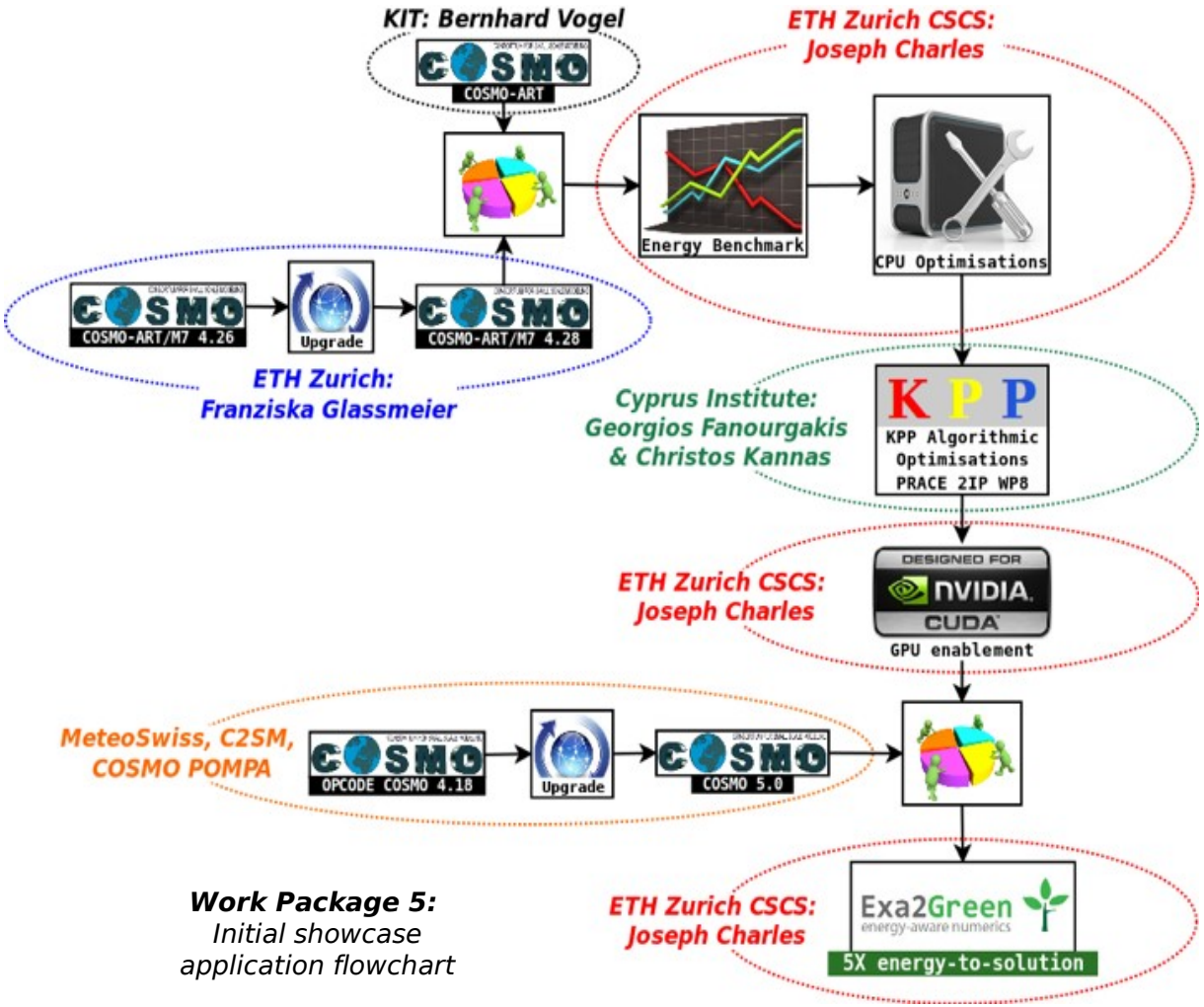
Exa2Green
energy-aware numerics

SEVENTH FRAMEWORK PROGRAMME

EU FP7-funded Exa2Green Project

"Energy-aware Sustainable Computing on Future Technology"

- **Seven European partners:**
  - University of Hamburg,
  - University of Jaume,
  - University of Heidelberg,
  - ETH Zurich CSCS,
  - IBM Rueschlikon,
  - Karlsruhe Inst. of Tech.,
  - Steinbeis Innovation gGmbH
- **Human resources:**
  36 PMs for CSCS, 261.6 PMs overall
- **Framework:**
  Covers all essential fields of expertise in energy-efficient computing
- **Showcase application:**
  COSMO-ART and/or COSMO-ART/M7
- **Ultimate goal:**
  5x improvement in energy-to-solution over baseline
- **Boundary conditions:**
  leverage off of HP2C COSMO work, maximize benefit for Swiss climate community



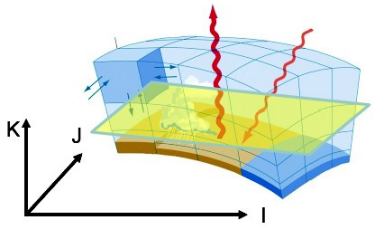
Work Package 5:
Initial showcase application flowchart

Exa2Green
energy-aware numerics

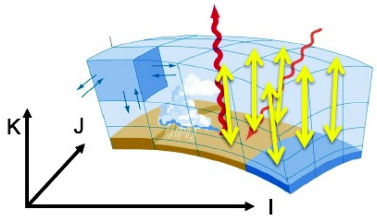
SEVENTH FRAMEWORK PROGRAMME

COSMO specific requirements

HP2C COSMO-CLM project: design of STELLA (STEncil Loop LAnguage)



Explicit solves in the horizontal IJ-planes
using finite difference stencils
No loop carried dependencies



Implicit solves in the vertical K-direction
using one tri-diagonal solve per column
Loop carried dependencies in K

Dynamical core

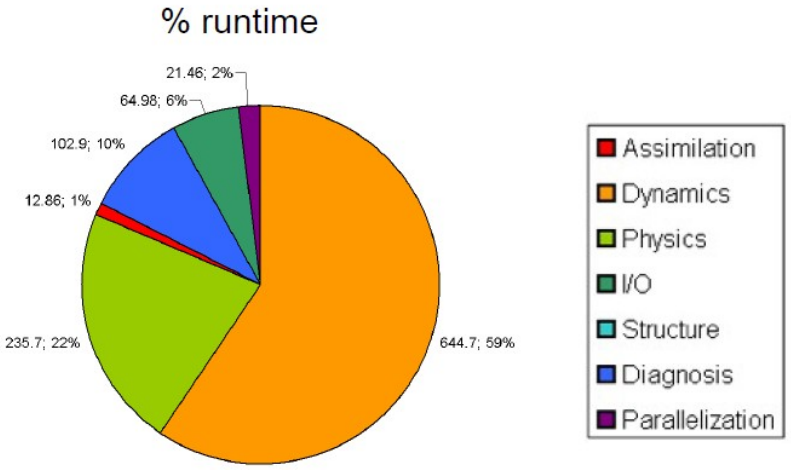
- Small group of developers
- Memory bandwidth bound
- Complex stencils (3D)
- 60% of runtime

→ **Complete rewrite in C++/CUDA**
→ Development of a stencil library
→ Development of new communication library (GCL)
→ Target architecture CPU (x86) and GPU.
→ Extendable to other architectures
→ Long term adaptation of the model

Physics and Data Assimilation

- Large group of developers
- Code may be shared with other models
- Less memory bandwidth bound
- Large part of code (50% of the lines)
- 20% of runtime

→ **GPU port with compiler directives (OpenACC)**
→ Little code optimization
→ Some parts stay on CPU
→ Most ported routines currently have CPU and GPU version



Exa2Green
energy-aware numerics

SEVENTH FRAMEWORK PROGRAMME

STELLA Library

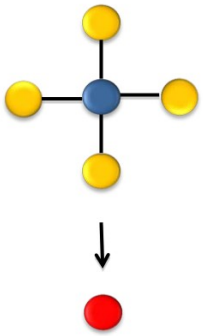
Design considerations for efficient parallelisation

- 2 fundamental concepts treated separately:
  - loop-logic: platform specific, implemented by the library
  - update-function: platform independent, implemented by the user

```
do k = kstart, kend
  do j = jstart, jend
    do i = istart, iend
      lap(i, j, k) = -4.0_ir * data(i, j, k) + &
        data(i+1, j  , k) + data(i-1, j  , k) + &
        data(i  , j+1, k) + data(i  , j-1, k)
    end do
  end do
end do
```

Loop-logic

Update-function



- Domain Specific Embedded Language using C++ template meta-programming to define:
  - loop-logic (sweeps: K-loops, stages: parallel IJ-loops)
  - temporary fields (stencil buffers, stage variables)

- 2 optimised backends:
  - X86 CPU: OpenMP, KIJ-storage
  - NVIDIA GPU: CUDA, IJK-storage

- Many features:

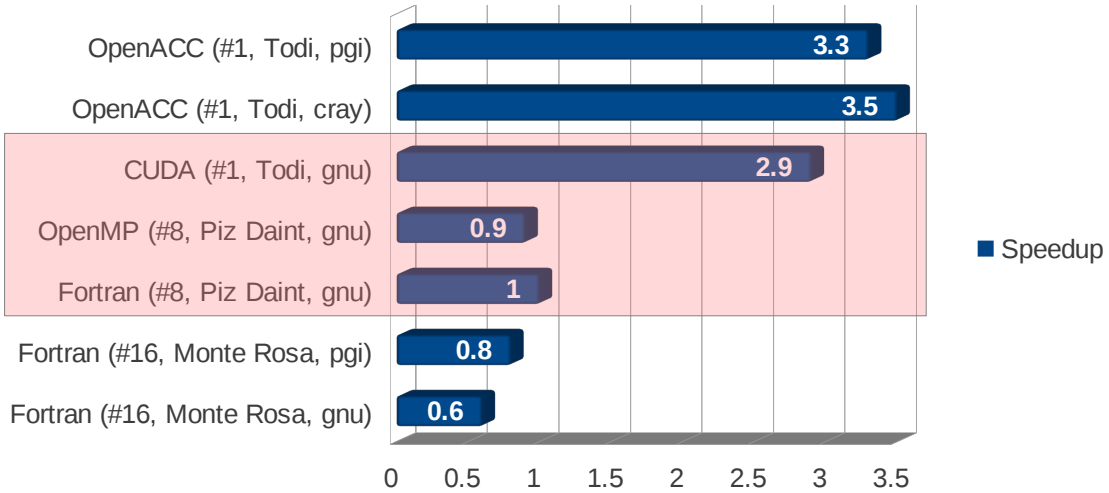
  loop definitions, loop merging, boundary conditions, functions, vertical K-Domains, buffers, software managed caching / data locality, parallelization, ...

STELLA Library

Evaluation of the graupel microphysical parameterisation of COSMO

Single-node evaluation on a 263x92x60 test domain with 1000 iterations

Computing resources	Socket infrastructure description	Compilers
Monte Rosa – Cray XE6	16-core AMD Opteron Interlagos CPU	GNU and PGI compilers
Piz Daint – Cray XC30	8-core Intel Sandy Bridge CPU	GNU compiler
Todi – Cray XK7	K20x GPU	GNU, CRAY and PGI compilers



OpenACC additional optimization

Loops restructured to iterate over blocks:
f(ie,je,ke) → f_b(nproma, ke)

with nproma = ie x je

Exa2Green
energy-aware numerics

SEVENTH FRAMEWORK PROGRAMME

Conclusion

- ***Pros of STELLA:***
  - *Easy of use:* based on standard programming language and compiler, only C++ tool chain needed, better separation of implementation strategy and algorithm, single source code runs on CPU and GPU
  - *User-friendliness:* frees model developer from implementation details: CPU and GPU specific implementation strategies are abstracted
  - *Performance:* achieves good speedups on CPU and GPU for stencils-based codes
  - *Support:* unit testing and serialization framework were crucial for efficient porting and bug hunting

- ***Cons of STELLA:***
  - *Specificity:* provides fixed set of features, some of them are very COSMO specific
  - *Procedure:* generates long compilation times
  - *Coding investments:* requires big change with respect to an original Fortran code, forces sometimes to adopt heavy coding conventions syntax
  - *Effectiveness:* retains not necessarily efficiency with single source code

Thank you for your attention!

Questions?