Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

C2SM
Center for Climate Systems Modeling

CSCS
Swiss National Supercomputing Centre

SCS
super computing systems

HP2C

Towards a full GPU version of the COSMO model

Status of the CONSORTIUM FOR SMALL SCALE MODELING COSMO *priority project*

Performance on Massively Parallel Architectures (POMPA)

Philippe Steiner and the whole project team

MeteoSwiss and patners

35th EWGLAM and 20th SRNWP Meeting, 2013, Antalya

Outline

- Motivation
- Why GPUs are attractive for COSMO?
- Approach
- Results
- Conclusion

Fundamental problem

- Clear trend in high performance computing (HPC) architectures to become heterogeneous (GPUs, MIC, …)
- Programming models are not getting simpler (OpenMP, OpenACC, NEC directives, software managed memory, …)
- Accelerators are an attractive alternative for COSMO, but we will always want to run on a plain CPU machine

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?**

A priori not clear how to solve this with the current COSMO code

Potential of GPUs

Chip	CPU (Sandy Bridge)		GPU (Kepler)
Architecture	8 cores		512 cores
Peak Performance	167 GFlops	**compute intensive** **× 8**	1300 GFlops
Memory Bandwidth	60 GB/s	**memory intensive** **× 3**	212 GB/s
Power Consumption	115 Watt		235 Watt
Price per Socket	~ X $		~ Y $

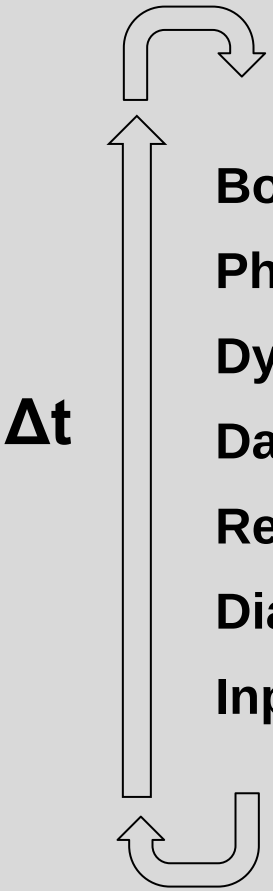
Priority Project POMPA

- **P**erformance **O**n **M**assively **P**arallel **A**rchitectures
- 4 year project (09.2010 – 12.2014)
- Lead: Oliver Fuhrer (MeteoSwiss)

- **Goal**
  Prepare the COSMO model code for these future HPC architectures

COSMO Workflow

Initialization



Boundary conditions

Physics

Dynamics

Data assimilation

Relaxation

Diagnostics

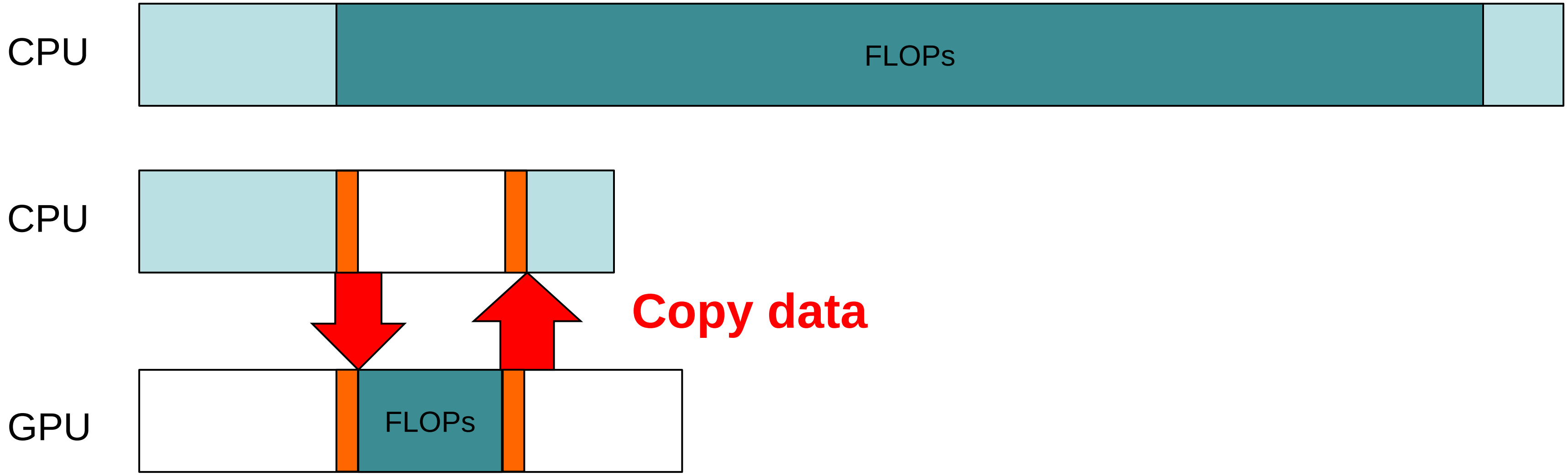
Input / Output

Properties

- PDEs
- Finite differences
- Structured grid
- Sequential workflow

Accelerator approach

- Leverage high peak performance of GPU
- CPU and GPU have different memories



Copy data

This approach does not work for COSMO!

Why does this not work for COSMO?

- Low FLOP count per load/store (stencils!)
- Transfer of data on each timestep too expensive

*

Part	Time/Δt
Dynamics	**172 ms**
Physics	**36 ms**
Total	**253 ms**

vs

§

Transfer of ten prognostic variables

118 ms

All code which touches the prognostic variables within timestep has to be ported

Full GPU Port

POMPA follows the goal of…

GPU-implementation of “full” time step of COSMO

Aim for…

- Completeness (i.e. full COSMO model)
- Performance (i.e. lower time-to-solution)
- Portability / Maintainability (i.e. no hacks)
- Durability (i.e. knowledge transfer and documentation)

Approach

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 (STELLA)
→ 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

Performance of Dynamical Core

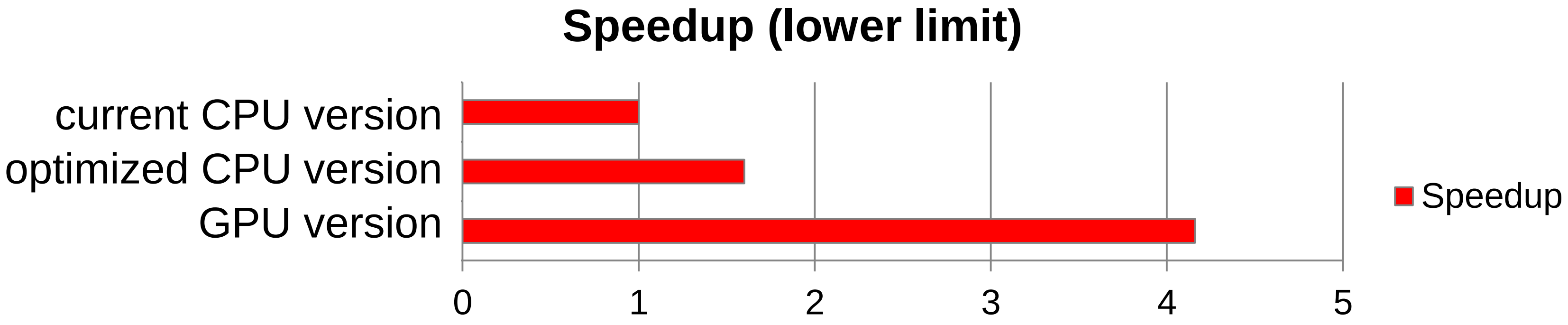
Test domain 128x128x60. CPU: 16 cores Interlagos CPU; GPU: Fermi

CPU Version

- Factor 1.6x – 1.8x faster than the COSMO dycore
- No explicit use of vector instructions (potential for 10-30% improvement)

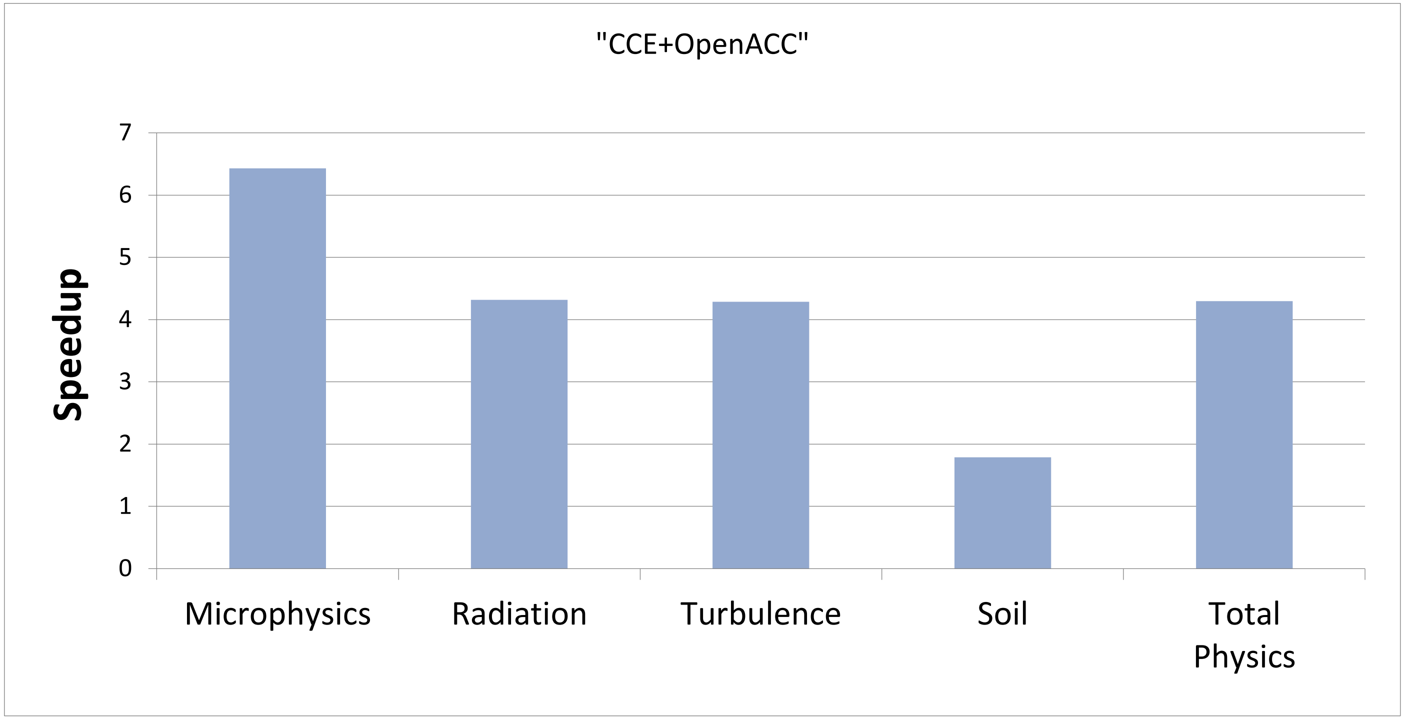
GPU Version

- Same generation GPU is roughly a factor 2.6x faster than CPU
- Ongoing performance optimizations



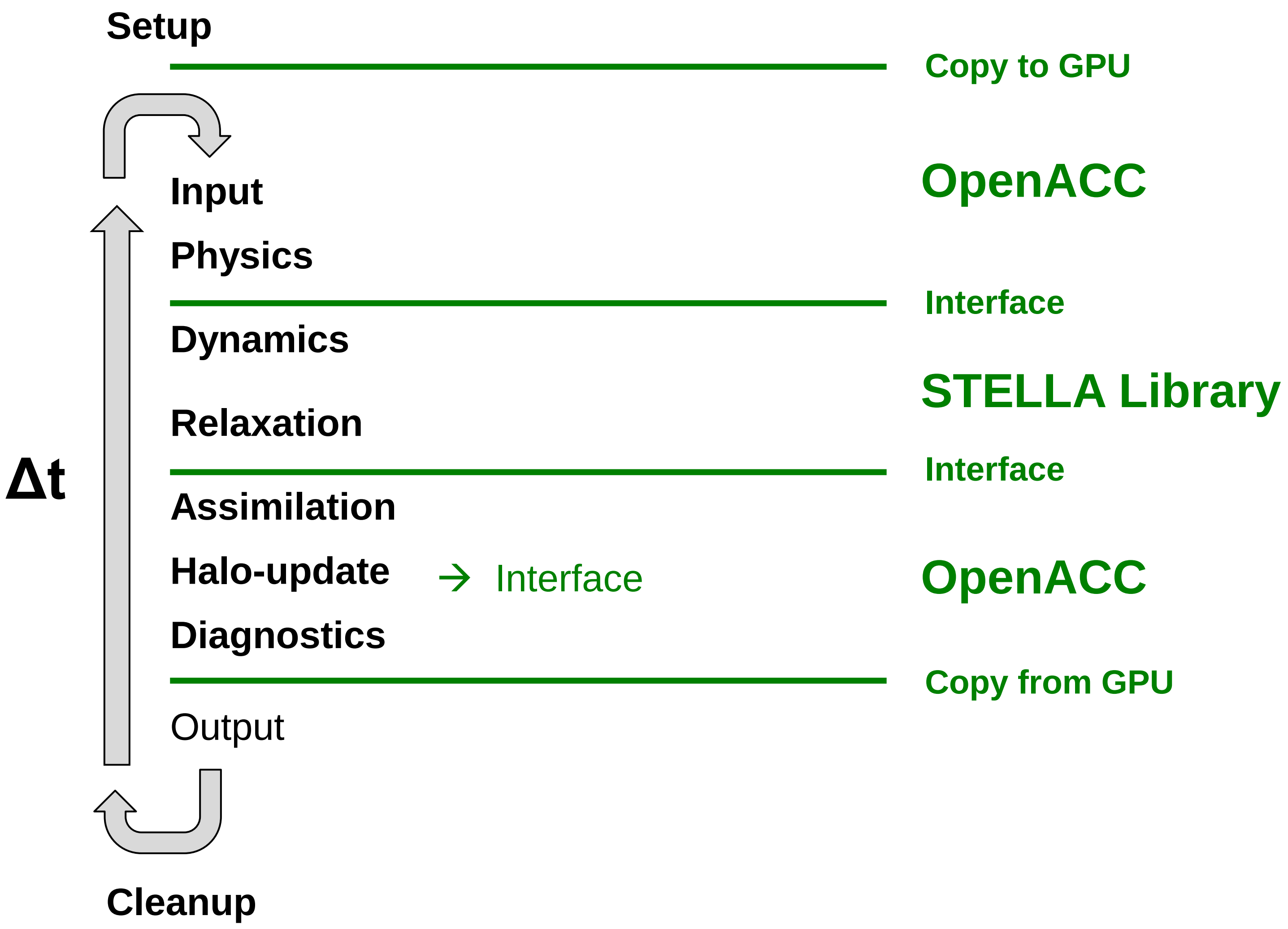
Performance of Physics

Test domain 128x128x60. 16 cores CPU (Sandy Bridge) vs. GPU (Fermi)



- Overall speed up ~4x
- Running the GPU-Optimized code on CPU is about 25% slower
  → separate source code for time critical routines

Implementation



Setup

Copy to GPU

Input

Physics

OpenACC

Interface

Dynamics

Relaxation

STELLA Library

Interface

Δt

Assimilation

Halo-update → Interface

Diagnostics

OpenACC

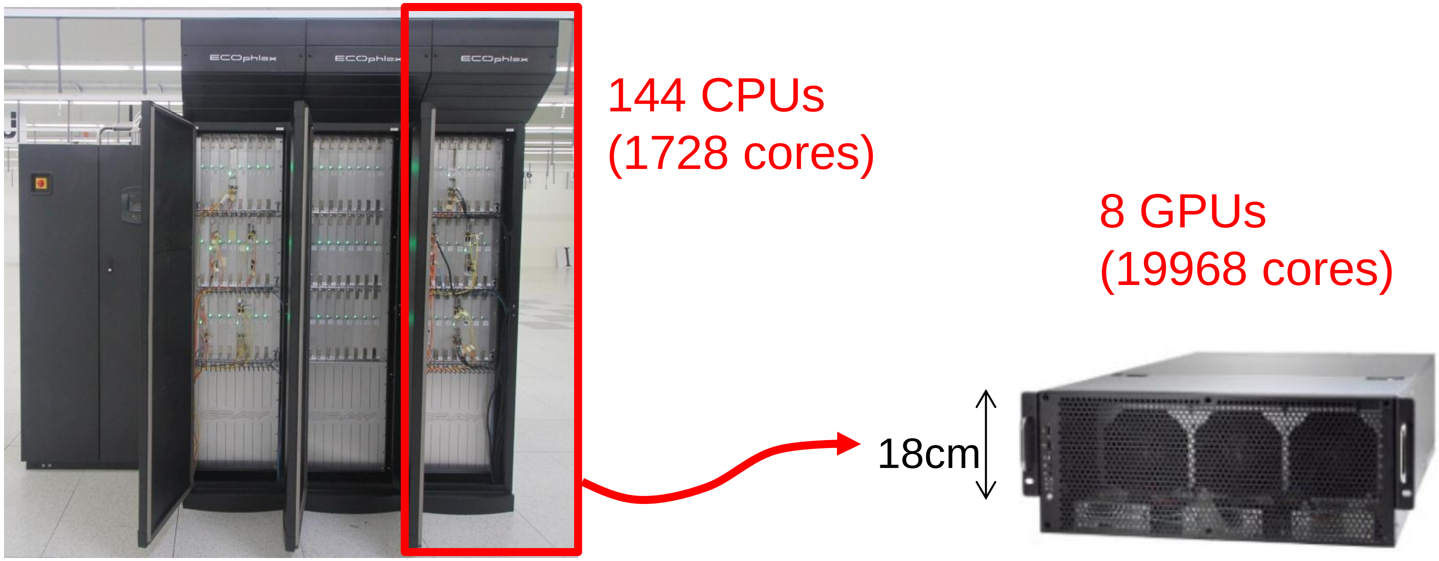
Copy from GPU

Output

Cleanup

Demonstrator

- Prototype implementation of the COSMO production suite of MeteoSwiss making aggressive use of GPU technology



144 CPUs
(1728 cores)

8 GPUs
(19968 cores)

18cm

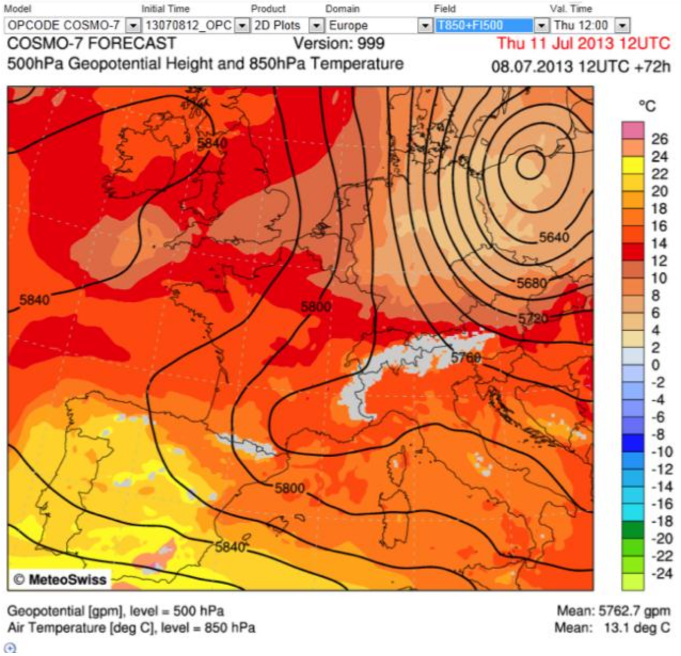
1 cabinet Cray XE6

Tyan Server FT77A

- Same time-to-solution on substantially cheaper hardware:
  Factor ~3x in price, factor ~9x in power consumption

Current status

- Branch of COSMO running on GPU-hardware
- Regular runs (00 UTC and 12 UTC)
- Full operational chain
  (plots delivered into visualization software)
- Almost full featured, missing features in progress



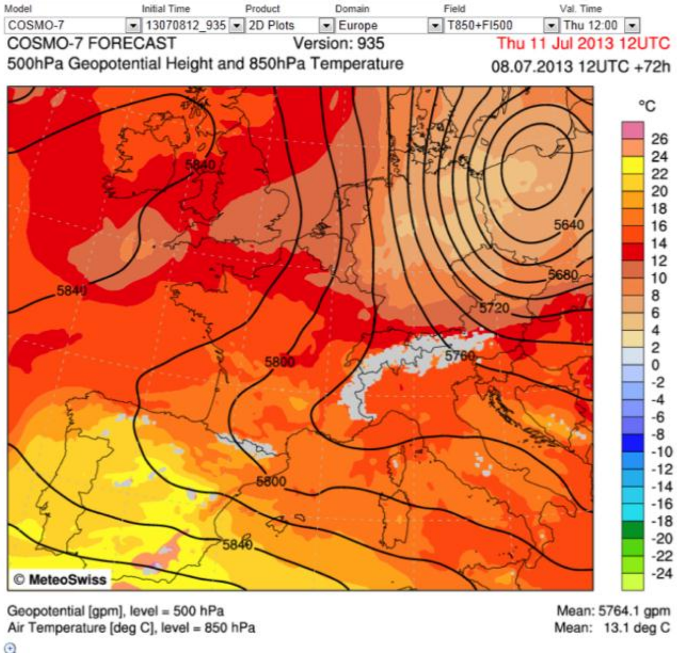
Model: OPCODE COSMO-7 | Initial Time: 13070812_OPC | Product: 2D Plots | Domain: Europe | Field: T850+FI500 | Val. Time: Thu 12:00

COSMO-7 FORECAST Version: 999 Thu 11 Jul 2013 12UTC

500hPa Geopotential Height and 850hPa Temperature 08.07.2013 12UTC +72h

Geopotential [gpm], level = 500 hPa Mean: 5762.7 gpm

Air Temperature [deg C], level = 850 hPa Mean: 13.1 deg C



Model: COSMO-7 | Initial Time: 13070812_935 | Product: 2D Plots | Domain: Europe | Field: T850+FI500 | Val. Time: Thu 12:00

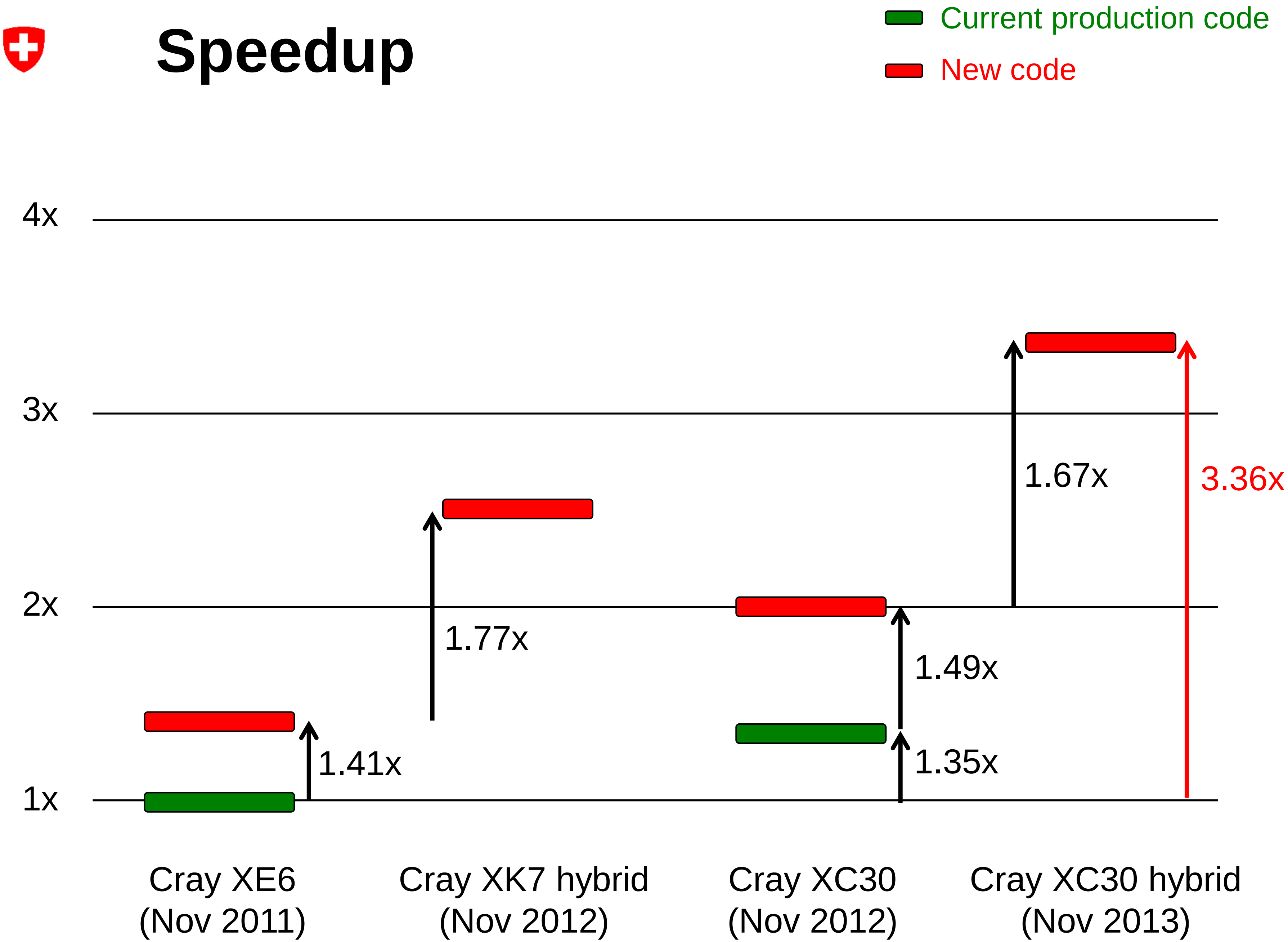
COSMO-7 FORECAST Version: 935 Thu 11 Jul 2013 12UTC

500hPa Geopotential Height and 850hPa Temperature 08.07.2013 12UTC +72h

Geopotential [gpm], level = 500 hPa Mean: 5764.1 gpm

Air Temperature [deg C], level = 850 hPa Mean: 13.1 deg C

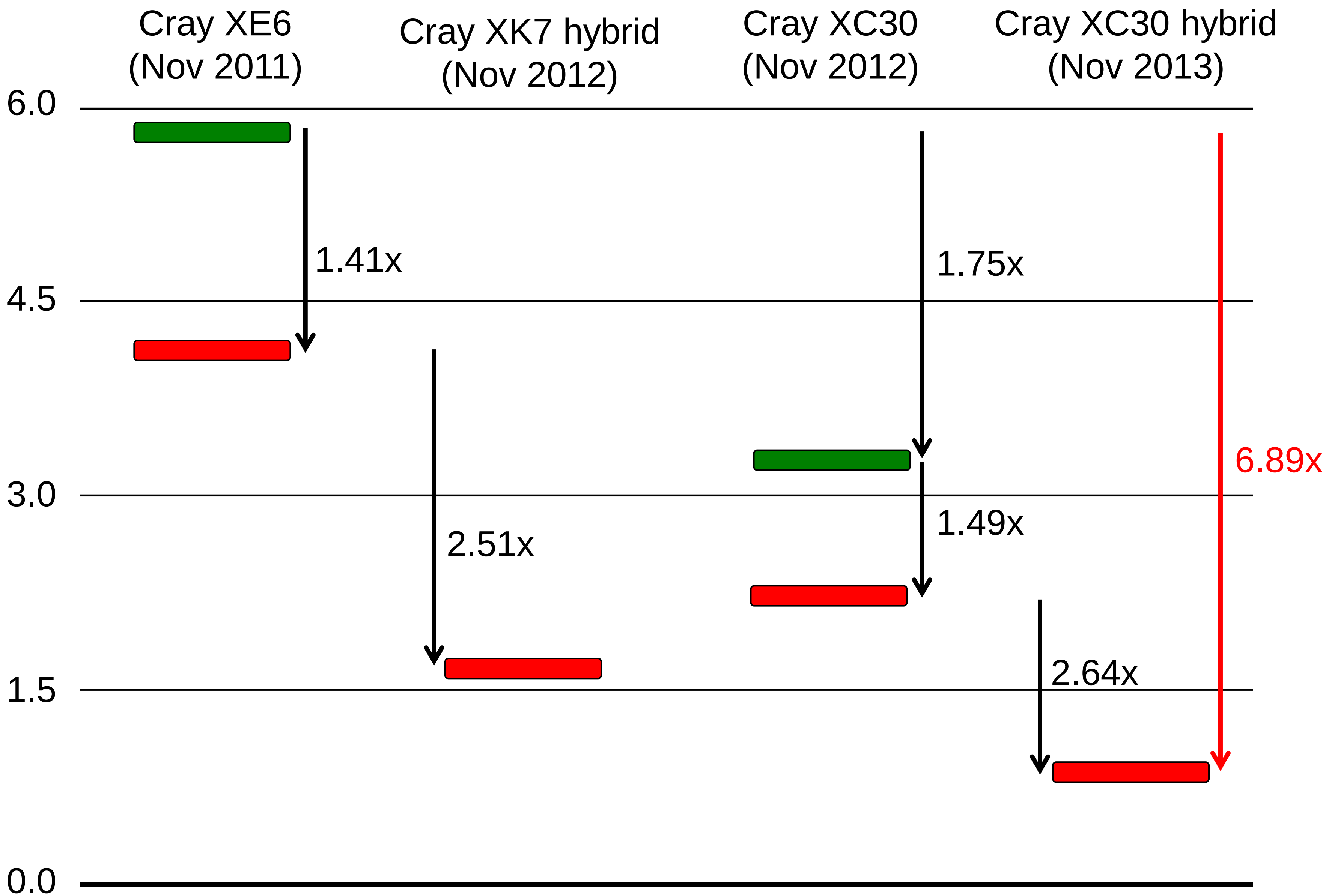
Speedup


🟩 Current production code
🟥 New code
4x
3x
2x
1x
1.41x
1.77x
1.35x
1.49x
1.67x
3.36x
Cray XE6
(Nov 2011)
Cray XK7 hybrid
(Nov 2012)
Cray XC30
(Nov 2012)
Cray XC30 hybrid
(Nov 2013)

Energy (kWh/member)

🟩 Current production code

🟥 New code



Next steps

- Upgrade to latest model version
- Unify CPU/GPU versions (physics & assimilation)
- Bring developments back to trunk
- Improve feature completeness
- Next version of STELLA

Conclusions

- **Complete rewrite of dynamical core using stencil library**
  - Single source code for GPU and CPU
  - Modern software engineering
  - Speedup of ~2x for CPU and ~5x for GPU

- **Porting of rest of code using compiler directives**
  - Physics (Speedup ~4x for GPU)
  - Assimilation (no speedup)

- **Integration of these developments into main trunk of COSMO code until end of 2014**

Thank you for your attention