PP POMPA (WG6)

Parallel Session

Welcome!

COSMO GM13, Sibiu

Agenda

16:30	Oliver Fuhrer	Overview
17:00	Xavier Lapillonne	OpenACC directives
17:20	Oliver Fuhrer	Dycore rewrite
17:40	André Walser	Single precision
18:00	all	Integration of POMPA developments
18:20	Zbigniew Piotrowski	How to marry POMPA and CELO?
18:40	Massimo Milelli	WG6 Science Plan
19:00	all	Any other discussion points?

Project Management

- **Revised project plan**
- **Final report** on the new dynamical core based on the stencil library
- **SMC** recommended to accepted project plan and to continue developments with goal to deliver a GPU-capable version of COSMO based on the stencil library
- **STC** gave go ahead and accepted revised project plan
- **Goal: POMPA developments in official version in December 2014**

Resources (planned + missing)

4.2 + 0.9

Task 3 Improve current parallelization (**0.6** + **0.1** **FTEs**)

- GPU-capable communication library
- Blocked physics package

Task 4 I/O Strategy (**0.4** **FTEs**)

- Porting of I/O code to GPUs

Task 5 Redesign dycore (**2.2** + **0.2** **FTEs**)

- Consolidation/improvements of stencil library
- Full featured RK dynamical core based on stencil library
- Update to COSMO v5.x
- Knowhow transfer

Task 6 GPU acceleration (**1.0** + **0.6** **FTEs**)

- Full featured physics based on OpenACC
- Update to COSMO v5.x
- Consolidation/improvements of OpenACC sections

PP POMPA Overview

- **Task 1** Performance analysis and documentation
- **Task 2** Redesign memory layout and data structures
- **Task 3** Improve current parallelization (→ discussion)
- **Task 4** Parallel I/O
- **Task 5** Redesign of dynamical core → Oli
- **Task 6** Explore GPU acceleration → Xavier
- **Task 7** Implementation documentation
- **Task 8** Single precision version → André

Overview of GPU Effort

- 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 on every timestep has to be ported

Full GPU Port

GPU-implementation of “full” timestep of COSMO

Aimed for…

- Completeness (full COSMO model)
- Performance (lower time-to-solution, higher efficiency)
- Portability / Maintainability (separation of concerns, no hacks, libraries)
- Durability (knowledge transfer and documentation)

- **Time / resource / technology constraints lead to compromises**

Approach

Dynamical core

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

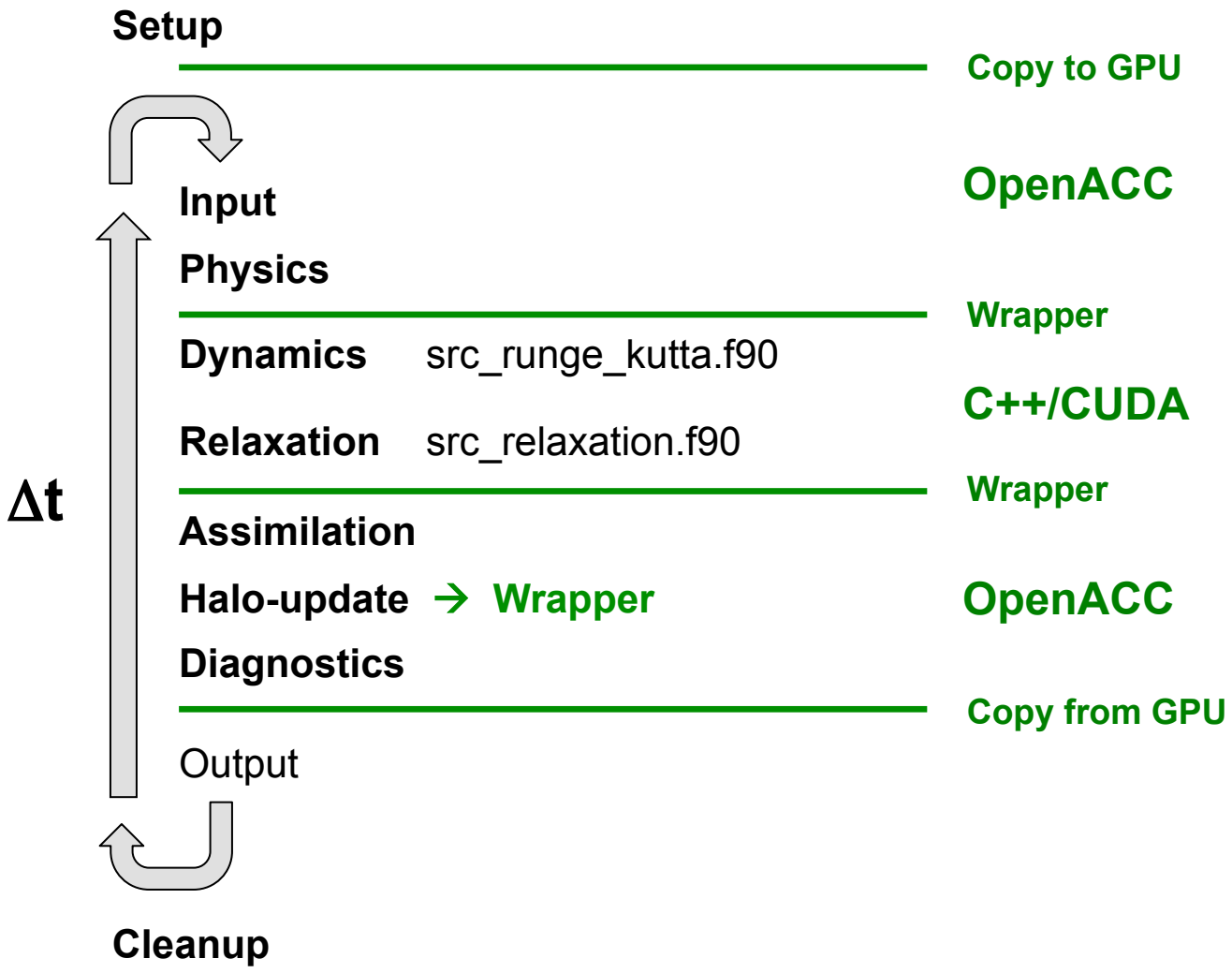
→ **Complete rewrite in C++/CUDA**
→ Development of a stencil library
→ Development of new communication library (GCL)
→ Target architecture x86 CPUs and NVIDIA GPUs.
→ Extendable to other architectures
→ Long term adaptation of the model

Physics, Data Assimilation, et al.

- 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

Implementation



Demonstration Project

- Leverage the research results of POMPA
- Prototype implementation of the COSMO production suite of MeteoSwiss making aggressive use of GPU technology
- Similar time-to-solution on substantially cheaper hardware:

Cray XE6
144 CPUs

1 cabinet

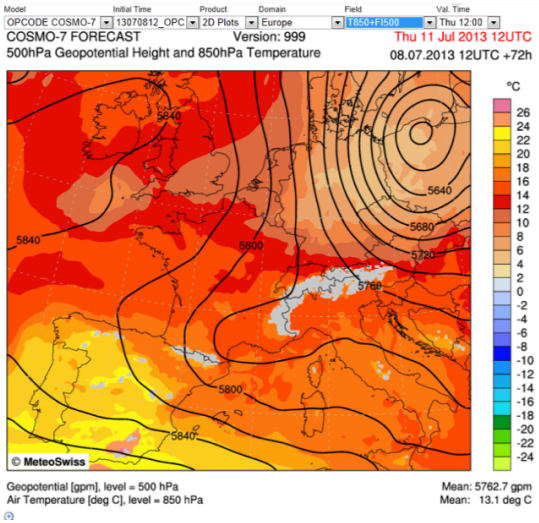
1 CPU
8 GPUs

1 server rack

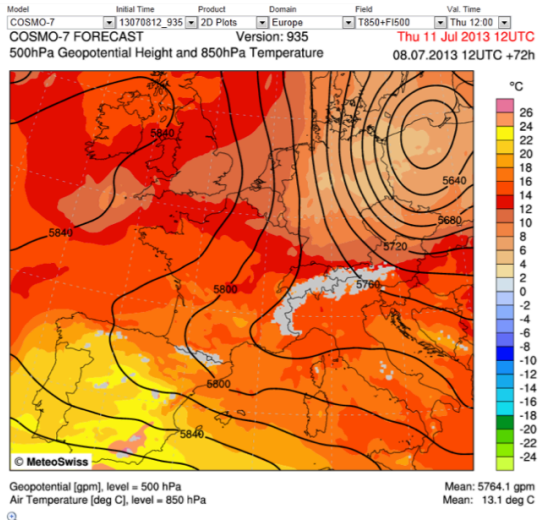
Status

- Prototype of 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

opcode

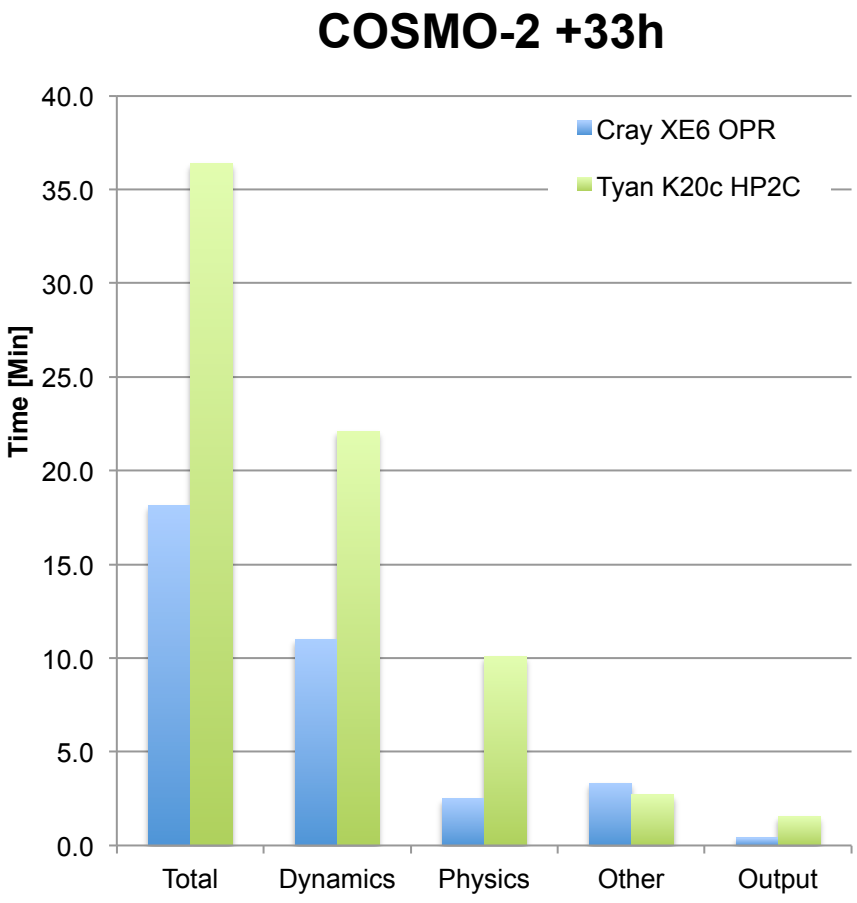


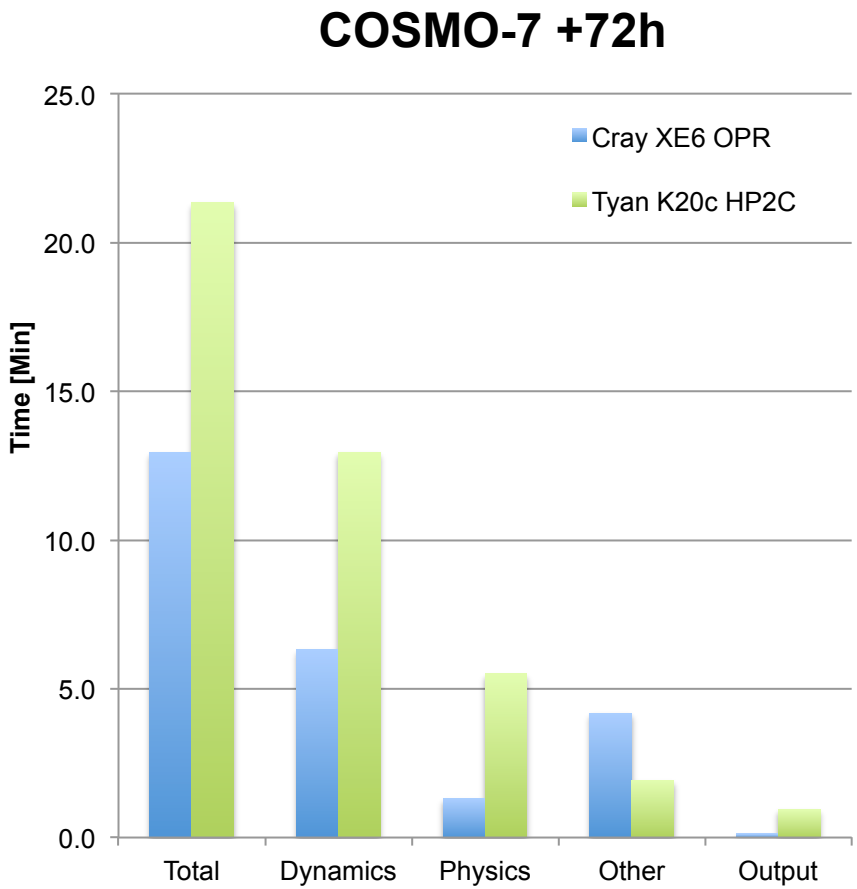
opr



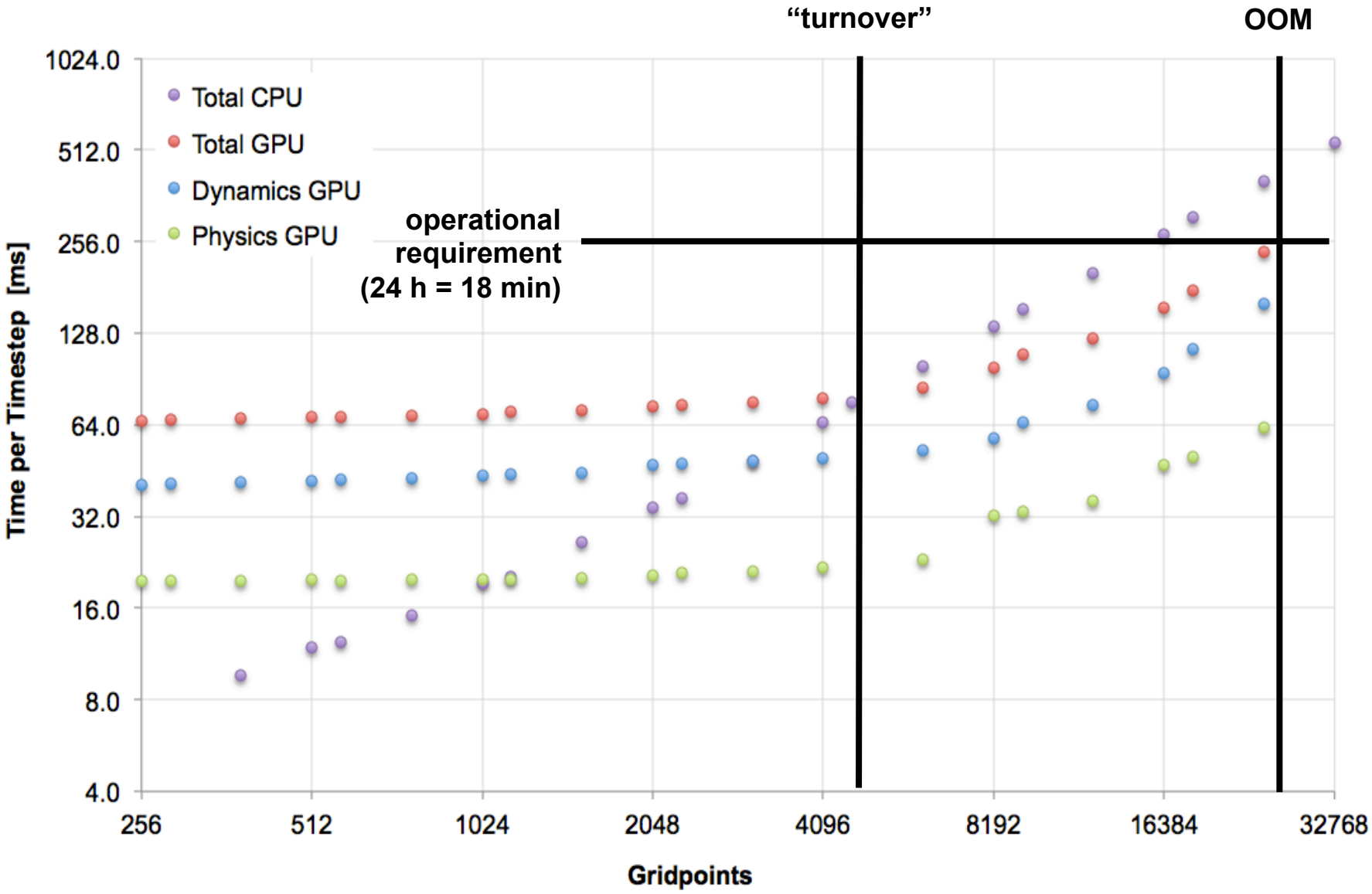
COSMO Performance Comparison

Cray XE6 albis 65 nodes vs. opcode 1 node (8 GPUs):





C-2 Single Node (K20x vs. SB)



Conclusions

- **Everything worked**…
  - Dynamical core re-write
  - Integration of CUDA/OpenACC/Fortran/C++/…
  - GPU-to-GPU communication
  - Collaboration

- Prototype (v4.19) capable of doing real-case simulations is available

- If you are interested, get involved!

Next steps…

- Port remaining parts
  - Physics
  - Dynamical core
  - I/O
- Consolidate code
- Bring developments back to official version
  - Re-ordering of operations
  - New communication interfaces
  - Single precision
  - New handling of BCs
  - Serialization
  - Block physics
  - Static memory allocations
  - Code refactorings
  - …

Thank you!

- Questions?