Bridging the software and performance gap to exascale for weather and climate simulations

Thomas C. Schulthess
Why extreme-scale computing in Switzerland?
Why extreme-scale computing in Switzerland?

• What computer performance would be necessary to include banner clouds of the Matterhorn in operational weather forecasting?

• A factor 2x in resolution corresponds approximately to a factor 10x in compute performance

• \[ \Delta x = 2200 \text{ m} \] (COSMO-2, until 2015)

• Factor x 1’000’000!
Where we are today, in 2019, and where we go next

Convective clouds ~ 1km (e.g. COSMO-1)

Global runs 100x too slow on Piz Daint!

Fog ~ 100 m – will need 1000x more

Banner clouds ~ 10 m – will need 100'000x more
Resolving convective clouds (convergence?)

**Structural convergence**

Statistics of cloud ensemble:
E.g., spacing and size of convective clouds

**Bulk convergence**

Area-averaged bulk effects upon ambient flow:
E.g., heating and moistening of cloud layer

Source: Christoph Schär, ETH Zurich
Structural and bulk convergence

Statistics of cloud area

- No structural convergence
- Factor 4

Statistics of up- & downdrafts

- Bulk statistics of updrafts converges

Source: Christoph Schär, ETH Zurich

(Panosetti et al. 2018)
Computational power drives spatial resolution

Moore's law with a doubling time of 18 months

Doubling time of 24 months

Can the delivery of a 1km-scale capability be pulled in by a decade?

Source: Christoph Schär, ETH Zurich, & Nils Wedi, ECMWF
Our “exascale” goal for 2022

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal resolution</td>
<td>1 km (globally quasi-uniform)</td>
</tr>
<tr>
<td>Vertical resolution</td>
<td>180 levels (surface to ~100 km)</td>
</tr>
<tr>
<td>Time resolution</td>
<td>Less than 1 minute</td>
</tr>
<tr>
<td>Coupled</td>
<td>Land-surface/ocean/ocean-waves/sea-ice</td>
</tr>
<tr>
<td>Atmosphere</td>
<td>Non-hydrostatic</td>
</tr>
<tr>
<td>Precision</td>
<td>Single (32bit) or mixed precision</td>
</tr>
<tr>
<td>Compute rate</td>
<td>1 SYPD (simulated year wall-clock day)</td>
</tr>
</tbody>
</table>
Running COSMO 5.0 & IFS (“the European Model”) at global scale on Piz Daint

Scaling to full system size: ~5300 GPU accelerate nodes available

Running a near-global (±80° covering 97% of Earths surface) COSMO 5.0 simulation & IFS

- Either on the hosts processors: Intel Xeon E5 2690v3 (Haswell 12c).
- Or on the GPU accelerator: PCIe version of NVIDIA GP100 (Pascal) GPU
# The baseline for COSMO-global and IFS

<table>
<thead>
<tr>
<th></th>
<th>Near-global COSMO\textsuperscript{15}</th>
<th>Global IFS\textsuperscript{16}</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Value</strong></td>
<td>Shortfall</td>
<td>Value</td>
</tr>
<tr>
<td><strong>Horizontal resolution</strong></td>
<td>0.93 km (non-uniform)</td>
<td>0.81×</td>
</tr>
<tr>
<td><strong>Vertical resolution</strong></td>
<td>60 levels (surface to 25 km)</td>
<td>3×</td>
</tr>
<tr>
<td><strong>Time resolution</strong></td>
<td>6 s (split-explicit with sub-stepping)\textsuperscript{*}</td>
<td>–</td>
</tr>
<tr>
<td><strong>Coupled</strong></td>
<td>No</td>
<td>1.2×</td>
</tr>
<tr>
<td><strong>Atmosphere</strong></td>
<td>Non-hydrostatic</td>
<td>–</td>
</tr>
<tr>
<td><strong>Precision</strong></td>
<td>Single</td>
<td>–</td>
</tr>
<tr>
<td><strong>Compute rate</strong></td>
<td>0.043 SYPD</td>
<td>23×</td>
</tr>
<tr>
<td><strong>Other (e.g., physics, …)</strong></td>
<td>microphysics</td>
<td>1.5×</td>
</tr>
<tr>
<td><strong>Total shortfall</strong></td>
<td>101×</td>
<td></td>
</tr>
</tbody>
</table>
MeteoSwiss’ performance ambitions in 2013

- Constant budget for investments and operations
- Ensemble with multiple forecasts
- Grid 2.2 km → 1.1 km
- Data assimilation

We need a 40x improvement between 2012 and 2015 at constant cost
COSMO: old and new (refactored) code

**Old Code (current / Fortran):**
- Physics (Fortran)
- Dynamics (Fortran)
- MPI
- System

**New Code (new / Fortran):**
- Physics (Fortran) with OpenMP / OpenACC
- Dynamics (C++)
- Stencil library
- Boundary conditions & halo exchg.
- Shared Infrastructure
- Generic Comm. Library
- MPI or whatever
- System

* two different OpenMP backends
Where the factor 40 improvement came from

Investment in software allowed mathematical improvements and change in architecture

- 2.8x Moore's Law & arch. improvements on x86
- 2.3x Change in architecture (CPU → GPU)
- 2.8x Mathematical improvements (resource utilisation, precision)
- 1.7x from software refactoring (old vs. new implementation on x86)
- 6x Requirements from MeteoSwiss
- 24x Ensemble with multiple forecasts
- 10x Grid 2.2 km → 1.1 km
- 10x Constant budget for investments and operations
- 1.3x additional processors
- Bonus: reduction in power!

There is no silver bullet!
Since April 2016, the Swiss version* of the COSMO model is running operationally on GPUs

(*) Swiss version of the COSMO model is running at 1km horizontal resolution over Alpine region and was (in 2016) ~10x more efficient than the state of the art
The baseline for COSMO-global and IFS

<table>
<thead>
<tr>
<th></th>
<th>Near-global COSMO&lt;sup&gt;15&lt;/sup&gt;</th>
<th>Global IFS&lt;sup&gt;16&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Value</td>
<td>Shortfall</td>
</tr>
<tr>
<td><strong>Horizontal resolution</strong></td>
<td>0.93 km (non-uniform)</td>
<td>0.81×</td>
</tr>
<tr>
<td><strong>Vertical resolution</strong></td>
<td>60 levels (surface to 25 km)</td>
<td>3×</td>
</tr>
<tr>
<td><strong>Time resolution</strong></td>
<td>6 s (split-explicit with sub-stepping)&lt;sup&gt;*&lt;/sup&gt;</td>
<td>–</td>
</tr>
<tr>
<td><strong>Coupled</strong></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td><strong>Atmosphere</strong></td>
<td>Non-hydrostatic</td>
<td>–</td>
</tr>
<tr>
<td><strong>Precision</strong></td>
<td>Single</td>
<td>–</td>
</tr>
<tr>
<td><strong>Compute rate</strong></td>
<td>0.043 SYD</td>
<td>3×</td>
</tr>
<tr>
<td><strong>Other (e.g., physics,...)</strong></td>
<td>microphysics</td>
<td>1.5×</td>
</tr>
<tr>
<td><strong>Total shortfall</strong></td>
<td>101×</td>
<td></td>
</tr>
</tbody>
</table>

Goal is to stay within ~ 5MW

100x (single trajectory) times 50x (ensemble)
Memory use efficiency

Fuhrer et al., Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2017-230, published 2018

\[ MUE = \text{I/O efficiency} \cdot \text{BW efficiency} = \frac{Q}{D} \cdot \frac{B}{\hat{B}} = 0.67 \]

Necessary data transfers
Achieved BW
Actual data transfers
Max achievable BW (STREAM)

0.55 w. regard to peak BW
2x lower than peak BW

\[ \text{2x lower than peak BW} \]

\[ \text{Max achievable BW (STREAM)} \]

\[ \text{0.55 w. regard to peak BW} \]

\[ \text{2x lower than peak BW} \]

\[ \text{Max achievable BW (STREAM)} \]
Can the 100x shortfall of a grid-based implementation like COSMO-global be overcome?

1. Icosahedral/octahedral grid (ICON/IFS) vs. Lat-long/Cartesian grid (COSMO)
   - 2x fewer grid-columns
   - Time step of 10 ms instead of 5 ms
   - 4x

2. Improving BW efficiency
   - Improve BW efficiency and peak BW
   - (results on Volta show this is realistic)
   - 2x

3. Strong scaling
   - 4x possible in COSMO, but we reduced available parallelism by factor 1.33
   - 3x

4. Remaining reduction in shortfall
   - Numerical algorithms (larger time steps)
   - Further improved processors / memory
   - 4x

But we don’t want to increase the footprint of the 2022 system succeeding “Piz Daint”
Much of the data present here was from this article

Abstract—We present a roadmap towards exascale computing based on true application performance goals. It is based on two state-of-the-art European numerical weather prediction models (IFS from ECMWF and COSMO from MeteoSwiss) and their current performance when run at very high spatial resolution on present-day supercomputers. We conclude that these models execute about 100–250 times too slow for operational throughput rates at a horizontal resolution of 1 km, even when executed on a full petascale system with nearly 5000 state-of-the-art hybrid GPU-CPU nodes. Our analysis of the performance in terms of a metric that assesses the efficiency of memory use shows a path to improve the performance of hardware and software in order to meet operational requirements early next decade.
The good news: memory performance is improving!
What about ensembles and throughput for climate? (Remaining goals beyond 2022)

1. Improve the throughput to 5 SYPD

   Change the architecture from control flow to data flow centric (reduce necessary data transfers)

\[
MUE = \text{I/O efficiency} \cdot \text{BW efficiency} = \frac{Q}{D} \frac{B}{B}
\]

2. Reduce the footprint of a single simulation by up to factor 10-50

   We may have to change the footprint of machines to hyper scale!
COSMO: old and new (refactored) code

- **Main (current / Fortran)**
  - Physics (Fortran)
  - Dynamics (Fortran)
  - MPI

- **Main (new / Fortran)**
  - Physics (Fortran) with OpenMP / OpenACC
  - Dynamics (C++)
  - Boundary conditions & halo exchg.
  - Stencil library
  - Shared Infrastructure
  - Generic Comm. Library
  - MPI or whatever

* two different OpenMP backends
COSMO: old and new (refactored) code

What about software, programming environment and toolchain?

* two different OpenMP backends
Vision for the toolchain

- Extension (e.g. DAPPY)
- GT4Py (Python)
- GT4F (Fortran)
- GTClang (C++)
- GridTools

High-level intermediate representation → Domain specific checkers → Optimisers → Code generator

- read before write
- missing boundary update
- data dependency race conditions
- out of bounds stencil access
- Software managed caches
- Full vertical parallelisation
- Stage fusion
- Data locality exploit
- Strong/weak scaling optimiser
- Native C/C++/Fortran
- Optimised GridTools Generator (C++)
GridTools – released 04/2019, finally!

(Current plan is to use GridTools as performance IR in the 2022 timeframe)
Grid Tools for Python (GT4Py) – arch. of 10/2019 release

- Transformation from Python AST to executable code
- Intermediate representation
  - Definition IR (high-level)
  - Implementation IR (low-level)
- Analysis phase to lower the IR
- Code generation with different targets
  - Python (Debug, NumPy)
  - C++ (GT with X86, Cuda, Phi)
This will work for any other domain and much better than writing “performance portable” Fortran/C/C++ code with OpenMP directives!!!
Collaborators on Exascale (climate)

- Tim Palmer (U. of Oxford)
- Bjorn Stevens (MPI-M)
- Peter Bauer (ECMWF)
- Oliver Fuhrer (Vucan)
- Nils Wedi (ECMWF)
- Carlos Osuna (MeteoSwiss)
- Torsten Hoefer (ETH Zurich)
- Mauro Bianco (CSCS)
- Christoph Schar (ETH Zurich)

As well as the team from CSCS’ Scientific Software and Libraries (SLL) section
Thank you!