Experience and challenges with maintaining a GPU-capable version of COSMO in a production environment at MeteoSwiss and ETH

Xavier Lapillonne, MeteoSwiss
V. Clement³, O. Fuhrer¹, C. Osuna¹, K. Osterried³, H. Vogt², A. Walser¹, C. Charpilloz¹, P. Spoerri³, T. Wicky

¹MeteoSwiss, ²CSCS, ³C2SM ETH
Outline

• Can weather forecast benefit from more computational resources?
• COSMO model on heterogeneous architecture
• Learning from GPU port and maintenance aspects
• Conclusions
Can weather forecast benefit from more computational resources?
Increase resolution

- Improve model representation and physical processes

2 x horizontal resolution $\approx 8 \times$ computational cost
Resolution of the computational grid

$\Delta x = 2200 \text{ m}$
Ensemble prediction

- Run N time the same forecast with perturbation
- Assess variability of weather situation
- Probabilistic forecast and warning
COSMO model on heterogeneous architecture
The COSMO model on GPU

- Local area numerical weather prediction and climate model (cosmo-model.org), 350 KLOC F90 + MPI
- Full GPU port strategy: avoid GPU-CPU data transfer
- Approach: OpenACC directives + Domain Specific Language (DSL) re-write

**Initialization**

- **Boundary conditions** → OpenACC port
- **Physics** → OpenACC port
- **Dynamics** → DSL re-write (C++)
- **Data assimilation** → partial OpenACC port
- **Halo-update** → Communication library (GCL)
- **Diagnostics** → OpenACC port
- **I/O** → partial OpenACC port

**Copy to accelerator**

**Δt**

**Cleanup**

Transfer to CPU on I/O step only
Operational setup at Meteoswiss

- Running on hybrid GPU system Piz Kesch
- Current (COSMO-1/E) operational setup represent a \(40x\) in computational cost as compare to previous system (COSMO-2/7)

**ECMWF-Model**
- 9 to 18 km gridspacing
- 2 to 4 x per day
- Provide boundary condition

**COSMO-1**
- 1.1 km gridspacing
- 8 x per day
- 1 to 2 d forecast
- (production since April 2016)

**COSMO-E**
- 2.2 km gridspacing
- 2 x per day
- 5 d forecast
- 21 members
- (production since June 2016)
## Multi-core vs. GPU-accelerated hybrid

<table>
<thead>
<tr>
<th></th>
<th>Piz Dora (old code)</th>
<th>Piz Kesch (new code)</th>
<th>Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sockets</strong></td>
<td>~26 CPUs</td>
<td>~7 GPUs</td>
<td>3.7 x</td>
</tr>
<tr>
<td><strong>Energy</strong></td>
<td>10 kWh</td>
<td>2.1 kWh</td>
<td>4.8 x</td>
</tr>
</tbody>
</table>
Application: Weather alerts
Example: Landslide RhB, 13. August 2014
Weather alerts
Example: Landslide RhB, 13. August 2014
Quasi-global simulation on Piz Daint

**Configuration**
- 80°S to 80°N covering 98.4% of Earth’s surface
- Regular lat/lon grid
- Analytical initial condition
- 10 day simulation
- Minimal I/O
- Metric: SYPD = Simulated years per wallclock day

Visualization by Tarun Chadha (C2SM): clouds > $10^{-3}$ g/kg (white) and precipitation > $4 \times 10^{-2}$ g/kg (blue)
Fuhrer et al., GMD 2018

**Piz Daint**
~5000 hybrid nodes with P100 GPUs (#3 Top500 11.2017)

**MeteoSwiss**
Strong scaling

- Near-global simulations at a fixed horizontal resolution

Fuhrer et al., 2017, GMD, in rev.
Learnings from the GPU port and code maintenance
Operational run on hybrid system

Piz Kesch (Cray CS Storm)

- 2 Cabinets (production & failover) installed at CSCS in July 2015
- 12 “Fat” compute nodes per cabinet
  - 2 Intel CPU Xeon E5 2690 (Haswell)
  - 8 Tesla K80 GPUs (each with 2 GK210 chip)

- Model Operator: “No adverse consequences in terms of maintenance, stability or complexity for operations”
Software development

- GPU support increase complexity of the software
- The new code can also now run in single precision (in addition to double)

⇒ Improve software development process: code review with dedicated tool (github Pull Request), design review ...
⇒ Automatic on demand and nightly build and tests (Jenkins)
Separation of concern

Fuhrer et al. 2014 (doi: 10.14529/jsfi140103)
Gysi et al. 2015 (doi: 10.1145/2807591.2807627)
OpenACC directives

- Incremental insertion in existing code
- Ideal for porting large components – avoid GPU-CPU copy

- May not be always performance portable: different optimal code for CPU and GPU (#ifdef ..)
- Difficult to maintain and debug in large monolithic legacy code
- What about ARM CPU, intel accelerator …?

Radiation scheme on CPU (Intel E5-2690v3 “Haswell”) and GPU (NVIDIA K80) using Fortran + OpenMP + OpenACC

Lapillonne and Fuhrer, PPL, 2016
Clement et al. 2018, PASC’18, 2018
#ifdef _OPENACC
 !$acc parallel
 !$acc loop gang vector
 DO j1 = ki1sc, ki1ec
  !$acc loop seq
  DO j3 = ki3sc+1, ki3ec

  ! Determine effect of the layer in *coe_th*
  CALL coe_th_gpu((j1,j3), pduh2oc, pduh2of, j1, j3), &
  pduco2, (j1,j3), pduo3, (j1,j3), &
  palogp, (j1,j3), palogt, (j1,j3), &
  podsc, (j1,j3), podsf, (j1,j3), &
  podac, (j1,j3), podaf, (j1,j3), &
  pbsfc, (j1,j3), pbsff, (j1,j3), &
  kspec, kh2o, kco2, ko3, &
  pa1c(j1), pa1f(j1), pa2c(j1), &
  pa2f(j1), pa3c(j1), pa3f(j1),
)

  pflfu(j1,j3) = (1.0_dp - pclc(j1,j3)) * pa3f(j1) &
  * (pbbbr(j1,j3) - pbbbr(j1,j3+1))
  pflcu(j1,j3) = pclc(j1,j3) * pa3c(j1) &
  * (pbbbr(j1,j3) - pbbbr(j1,j3+1))
  pflfd(j1,j3+1) = -pflfu(j1,j3)
  pflcd(j1,j3+1) = -pflcu(j1,j3)

...

ENDDO  ! End of vertical loop over layers
 !$acc end parallel
#else
 DO j3 = ki3sc+1, ki3ec

  ! Determine effect of the layer in *coe_th*
  CALL coe_th((pduh2oc,pduh2of,pduco2,pduo3,palogp,palogt, &
  podsc,podsfpodac,podaf,pbsfc,pbsff, &
  j3,kspec,kh2o,kco2,ko3, &
  ki1sd,kiled,ki3sd,ki3ed,ki1sc,ki1ec, &
  ldebug_coe_th,jindex, &
  pa1c,pa1f,pa2c,pa2f,pa3c,pa3f)

  ! Set RHS
  DO j1 = ki1sc, ki1ec
   pflfu(j1,j3) = (1.0_dp - pclc(j1,j3)) * pa3f(j1) &
   * (pbbbr(j1,j3) - pbbbr(j1,j3+1))

...
C++ embedded DSL

- Separation of concerns between user code and optimized backend
- Performance portable, future proof: can be extended to new architecture
- Easier to maintain (e.g., unit testing)
- No custom tool (rely on C++/CUDA compiler)
- Steep learning curve, domain scientist did not like the language
- Requires consequent boiler plate codes
- No reduction in LOC
- Still some hardware optimization in user code (e.g., GPU caching), requires expertise

Laplacian with Stella-DSL

```cpp
template<typename TEnv>
struct Divergence {
    STENCIL_STAGE(TEnv)
    STAGE_PARAMETER(FullDomain, phi)
    STAGE_PARAMETER(FullDomain, lap)
    STAGE_PARAMETER(FullDomain, flx)

    static void Do(Context ctx, FullDomain) {
        ctx[div::Center()] = ctx[phi::Center()] -
        ctx[alpha::Center()] * (ctx[flx::Center()]
        ctx[flx::At(jminus1)]) + ctx[fly::Center()]
        ctx[fly::At(jminus1)])
    }
};
```
High level language(s) for weather and climate

- Large legacy model need to be adapted
- Separations of concerns
- Increase abstraction: no explicit data structure, loops, HW-dependent details
- More optimizations, task parallelism
- Higher productivity and code safety

Example gtclang

```c
function avg
    offset off
    storage in
    avg = 0.5 * ( in(off) + in() )
}

function coriolis_force
    storage fc, in
    coriolis_force = fc() * in()
}

operator coriolis
    storage u_tend, u, v_tend, v, fc
    vertical_region ( k_start , k_end )
        u_tend += avg(j-1, coriolis_force(fc, avg(...
        v_tend -= avg(i-1, coriolis_force(fc, avg(...
    }
}
```

gtclang language prototype:
- Generates efficient code for x86 multicore, NVIDIA GPUs, Intel Xeon Phi
- 4x – 6x reduction in LOC
High level Intermediate Representation (HIR)

- Multiple high level DSL (different communities)
- Single HIR and optimization tool chain - currently based on GridTools Framework, Joint development between CSCS / MeteoSwiss / C2SM

CLAW-DSL : Fortran DSL for the physical parameterizations
Conclusions
Conclusions – Take home message

• Weather forecast can profit from HPC developments by increasing resolution and probabilistic capabilities

• Weather and climate model need to be adapted to make efficient use of new and emerging hardware

• The COSMO model was ported to GPU using Domain Specific Language and compiler directives, achieving 4x speed up compare to x86 CPU

• The model is run operationally at MeteoSwiss on a GPU system since 2016

• Maintenance of a legacy code on multiple architecture is challenging: better software engineering practice and more abstraction would improve maintainability
MeteoSwiss
Operation Center 1
CH-8058 Zurich-Airport
T +41 58 460 91 11
www.meteoswiss.ch

MeteoSvizzera
Via ai Monti 146
CH-6605 Locarno-Monti
T +41 58 460 92 22
www.meteosvizzera.ch

MétéoSuisse
7bis, av. de la Paix
CH-1211 Genève 2
T +41 58 460 98 88
www.meteosuisse.ch

MétéoSuisse
Chemin de l'Aérologie
CH-1530 Payerne
T +41 58 460 94 44
www.meteosuisse.ch