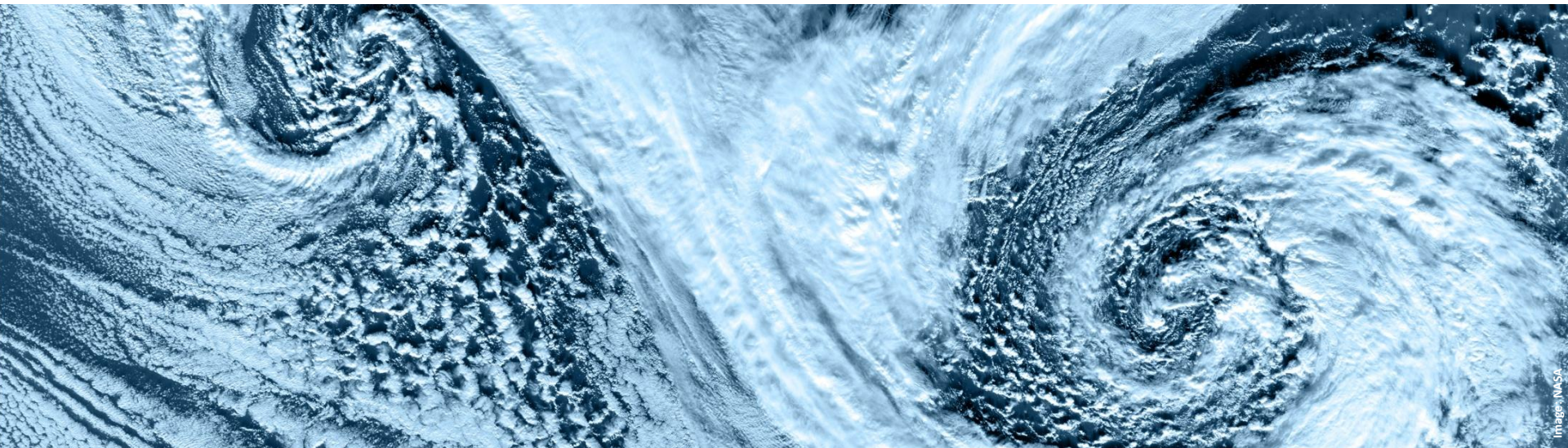


COSMO GPU: The C++ Dynamical Core

Pascal Spörri

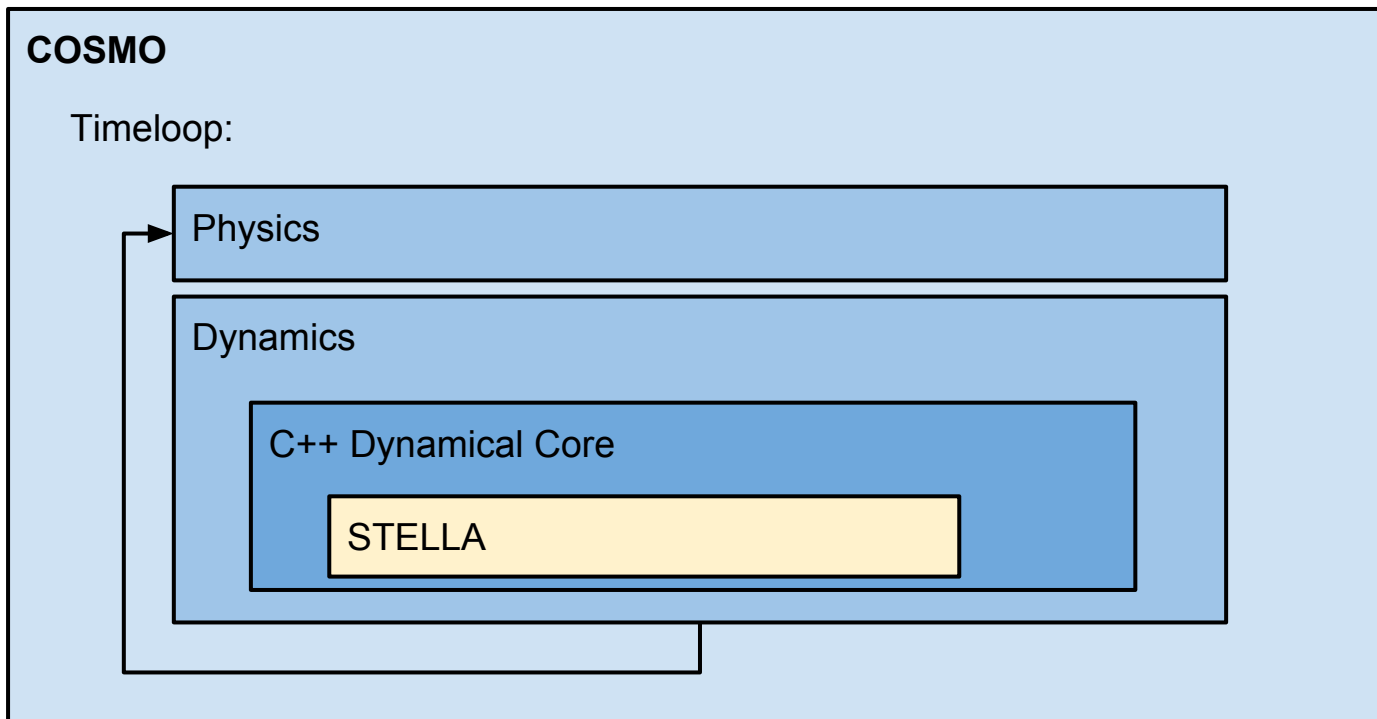
pascal.spoerri@env.ethz.ch



Who

- Maintainer for the C++ Dynamical Core
- MSc ETH Computer Science
- Replaced Andrea Arteaga
- Primarily work at MeteoSwiss
- User [pspoerri](#) on Github

What is the C++ Dynamical Core?



Why

Piz Escha/Kesch: 2x12 node machine with 16 GPUs/node

- COSMO-E 120h forecast: 1 node per member
- COSMO-1 33h forecast: 9 nodes

⇒ **COSMO GPU version required!**

Daily builds of COSMO_GPU are available on Piz Kesch and Piz Daint
`/project/c01/install/${machine}/cosmo5/cosmo_gpu_${precision}`

Projects

COSMO-POMPA

- Maintain C++ Dycore: Performance, Bugs
- Serialization in Fortran
- Build system, testing architecture

STELLA

- Small contributions

Serialbox (Fortran serialization)

- C++/Python interface

COSMO-PRERELEASE (COSMO 5.x dev trunk by DWD)

- Bringing back changes from MeteoSwiss

Status

Dynamical Core version 5.0 is now supported.

- Used pre operationally at MeteoSwiss
- Features (currently) restricted to Namelist used by OWM

Integration COSMO 5.3+ Features

- Work in progress

Dycore Standalone

- Work in progress

Questions?

Contact me if you need \$FEATURE in the C++ Dynamics (lcpp_dycore)

Backup Slides

Testing and Verification

- Unit testing
 - Use Pure Fortran COSMO to compute reference data
 - Unit test the Dycore against against the reference data
- Verification
 - Verify COSMO YUPRTEST with Dycore Enabled/Disabled
 - Use serialization and unit testing to locate the problem

Serialization

Comments get expanded by script

```
!$ser savepoint CoriolisUnittest.Apply-in LargeTimeStep=ntstep  
!$ser data u_nnow=u(:, :, :, nnow) v_nnow=v(:, :, :, nnow)  
!$ser data u_tens=utens v_tens=vtens
```

```
DO k = 1 , ke
```

```
  . . . .
```

```
ENDIF
```

```
!$ser savepoint CoriolisUnittest.Apply-out LargeTimeStep=ntstep  
!$ser data u_tens=utens v_tens=vtens
```

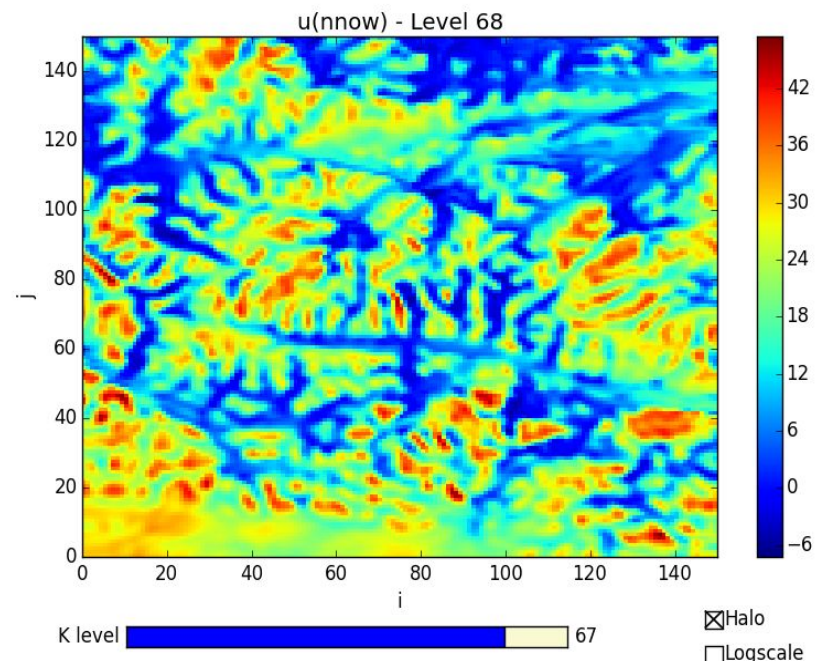
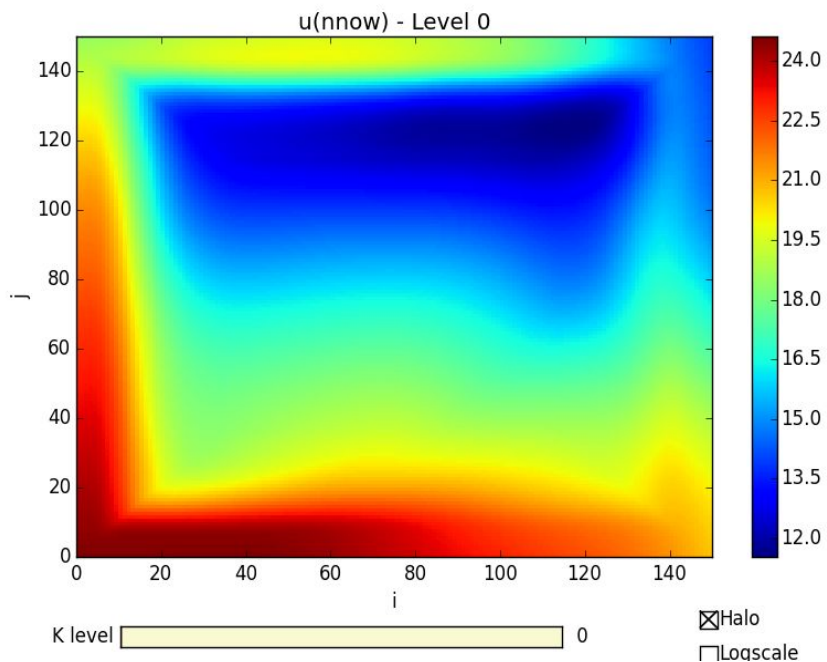
Python Serializer

- Python bindings to serializer
 - Write/read serialized data
 - Export fields to Numpy
 - Visualize 3D Fields
- Sample

```
from serialization_framework import *  
ser = Serializer("/path/to/data", "Fields")  
u_nnow = ser['CoriolisUnittest.Apply-in']['LargeTimeStep'][5]['u_nnow']  
Visualizer(u_nnow, 'u(nnow)')
```

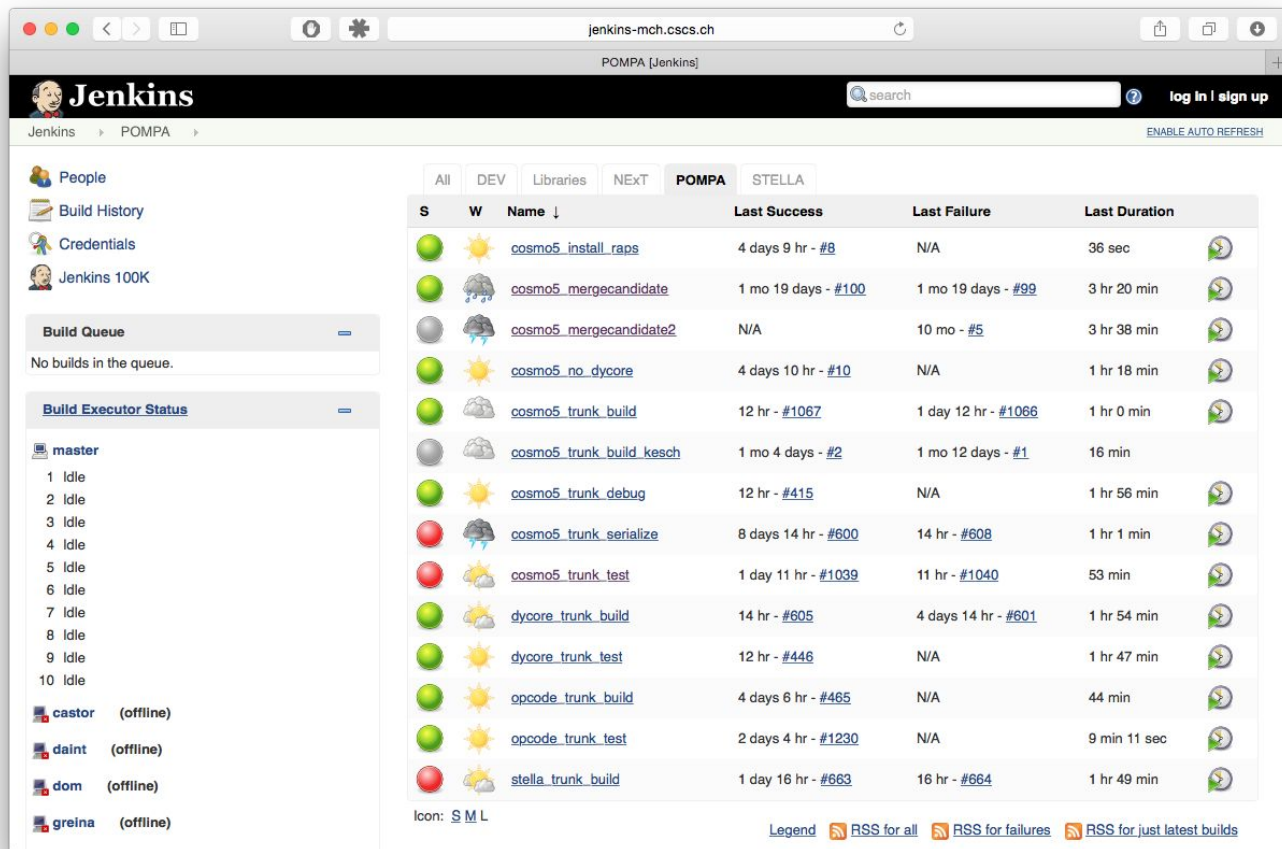
Python Visualizer

Explore data interactively



Automation

- Nightly builds
- Nightly tests



The screenshot shows the Jenkins web interface for the POMPA project. The main content area displays a table of build jobs with columns for status, workspace, name, last success, last failure, and last duration. The jobs are categorized by workspace (DEV, Libraries, NExT, POMPA, STELLA). The POMPA workspace contains several jobs, including 'cosmo5_install_raps', 'cosmo5_mergcandidate', 'cosmo5_mergcandidate2', 'cosmo5_no_dycore', 'cosmo5_trunk_build', 'cosmo5_trunk_build_kesch', 'cosmo5_trunk_debug', 'cosmo5_trunk_serialize', 'cosmo5_trunk_test', 'dycore_trunk_build', 'dycore_trunk_test', 'opcode_trunk_build', 'opcode_trunk_test', and 'stella_trunk_build'. The left sidebar shows the 'Build Queue' (empty) and 'Build Executor Status' (10 idle executors on the master node).

S	W	Name ↓	Last Success	Last Failure	Last Duration
🟢	☀️	cosmo5_install_raps	4 days 9 hr - #8	N/A	36 sec
🟢	☁️	cosmo5_mergcandidate	1 mo 19 days - #100	1 mo 19 days - #99	3 hr 20 min
🟡	☁️	cosmo5_mergcandidate2	N/A	10 mo - #5	3 hr 38 min
🟢	☀️	cosmo5_no_dycore	4 days 10 hr - #10	N/A	1 hr 18 min
🟢	☁️	cosmo5_trunk_build	12 hr - #1067	1 day 12 hr - #1066	1 hr 0 min
🟡	☁️	cosmo5_trunk_build_kesch	1 mo 4 days - #2	1 mo 12 days - #1	16 min
🟢	☀️	cosmo5_trunk_debug	12 hr - #415	N/A	1 hr 56 min
🔴	☁️	cosmo5_trunk_serialize	8 days 14 hr - #600	14 hr - #608	1 hr 1 min
🔴	☁️	cosmo5_trunk_test	1 day 11 hr - #1039	11 hr - #1040	53 min
🟢	☀️	dycore_trunk_build	14 hr - #605	4 days 14 hr - #601	1 hr 54 min
🟢	☀️	dycore_trunk_test	12 hr - #446	N/A	1 hr 47 min
🟢	☀️	opcode_trunk_build	4 days 6 hr - #465	N/A	44 min
🟢	☀️	opcode_trunk_test	2 days 4 hr - #1230	N/A	9 min 11 sec
🔴	☀️	stella_trunk_build	1 day 16 hr - #663	16 hr - #664	1 hr 49 min