CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Eurohack 2018: Team A-QuICC

Team members: Philippe Marti (ETH), Meredith Plumley (ETH)

Mentors: Christopher Bignamini (ETH Zurich / CSCS),
Theofilos-Ioannis Manitaras (ETH Zurich / CSCS)

Final Presentation – 5.10.2018

Technicalities

Modular design:

- Implementation of the recurrence relations for sparse operators
- Definition of linear operators, field names, etc
- C++ classes to define nonlinear interactions and IO

C++

- Computationally intensive part
- ≈ 100k lines of code

Python

- Embedded in C++ code
- Provides high-level interface to linear part of equations (sparse (block)
- matrices)
- Linear stability calculations through PETSc and SLEPc bindings
- ≈60k lines of code

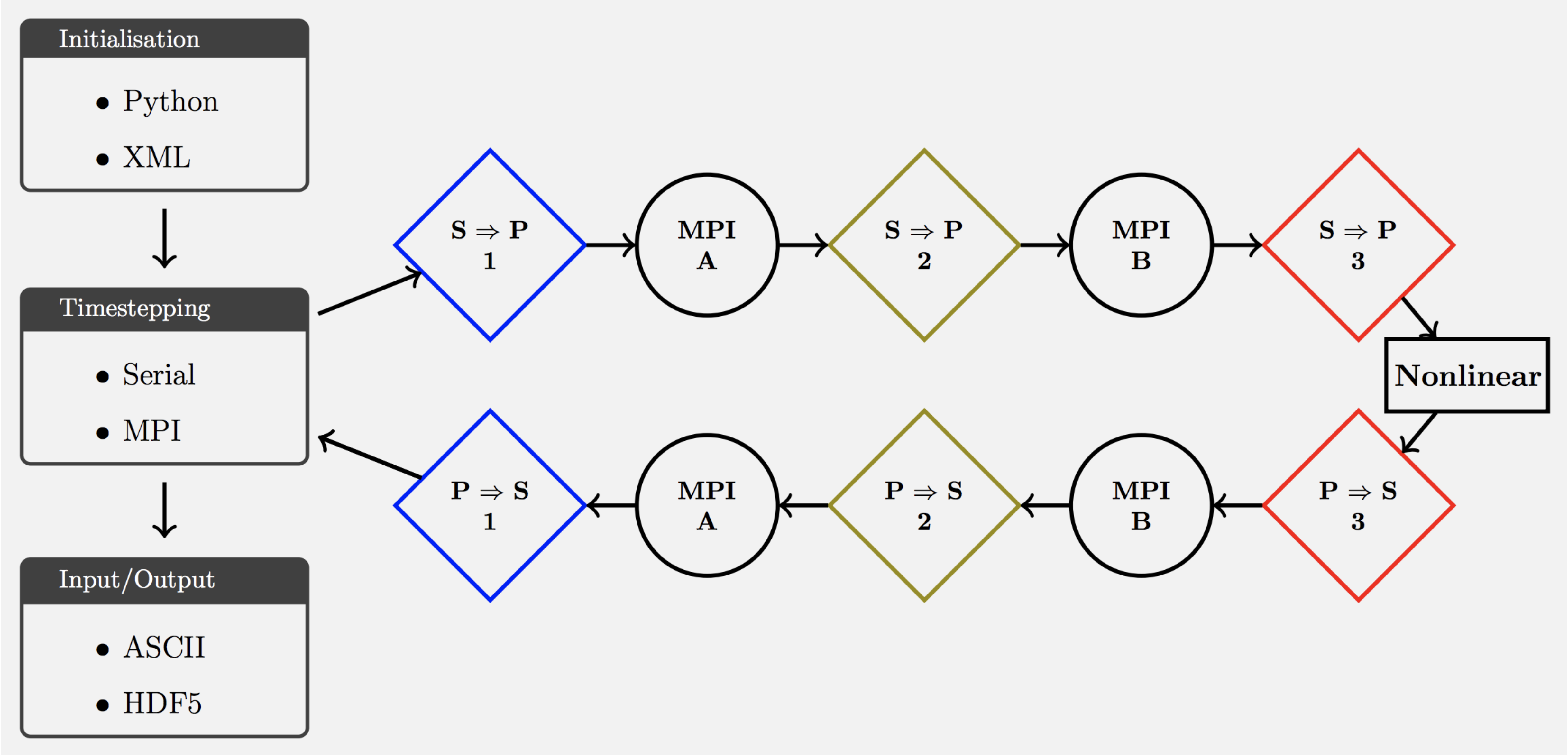
MPI parallelization

- All-to-All required by spectral method: MPI
- Alltoallv or manual Send/Recv
- possibly coupled to parallel sparse solver MUMPS
- Experimental hybrid MPI + threads/OpenMP and on-the-fly calculations

HDF5 file format

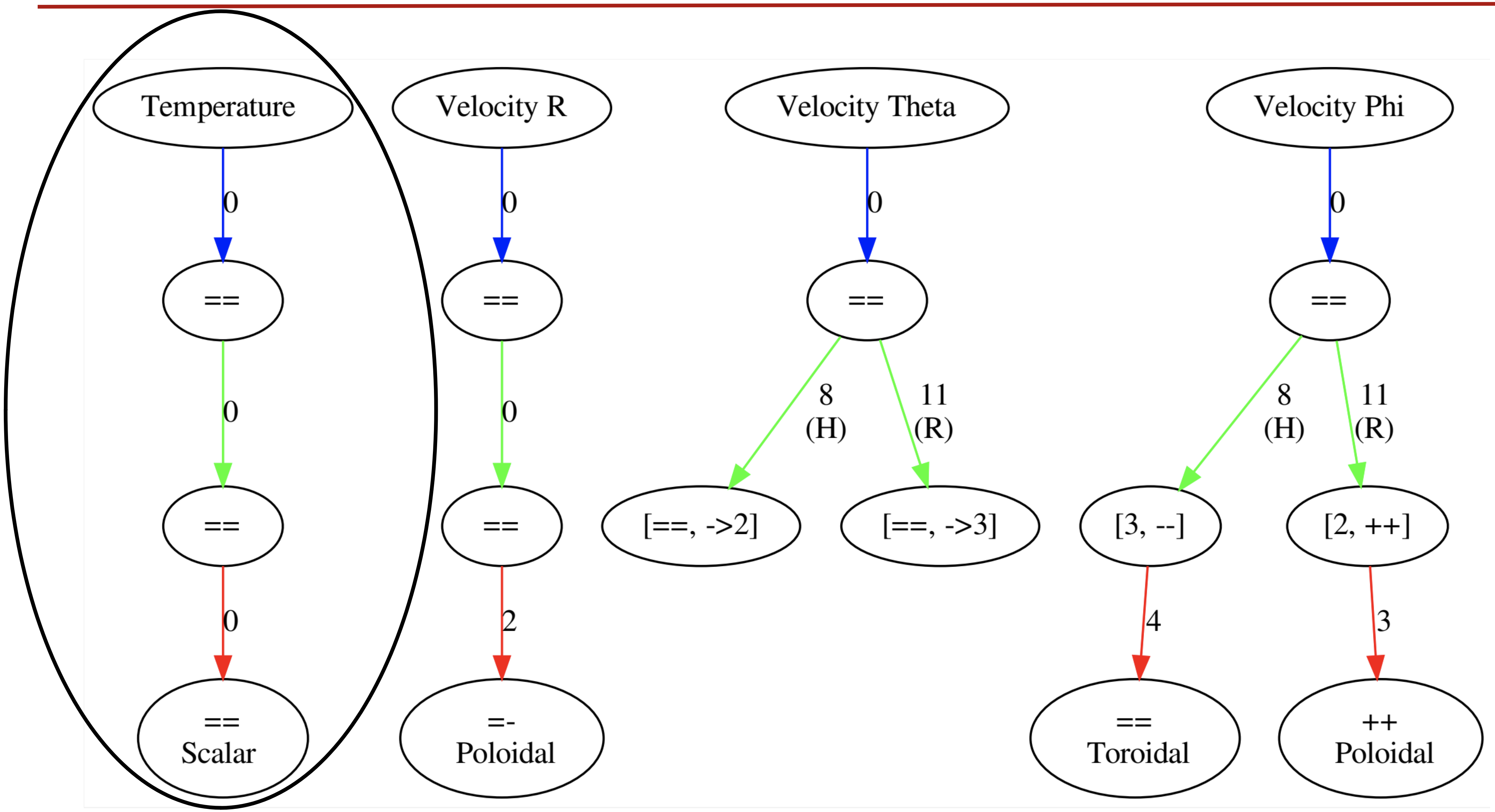
- parallel reading for large datasets (initial states)
- parallel writing for large datasets (snapshots/restart files)

Flow Chart

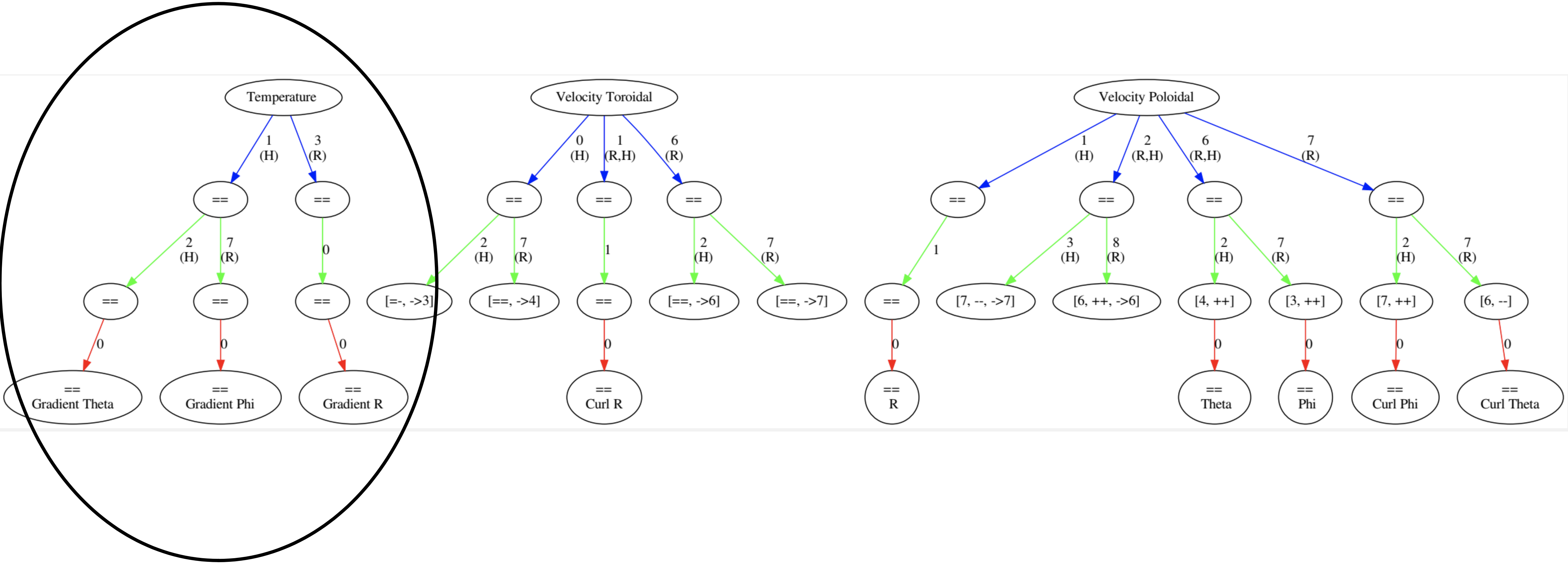


Initialisation
• Python
• XML
Timestepping
• Serial
• MPI
Input/Output
• ASCII
• HDF5
S ⇒ P 1
MPI A
S ⇒ P 2
MPI B
S ⇒ P 3
Nonlinear
P ⇒ S 3
MPI B
P ⇒ S 2
MPI A
P ⇒ S 1

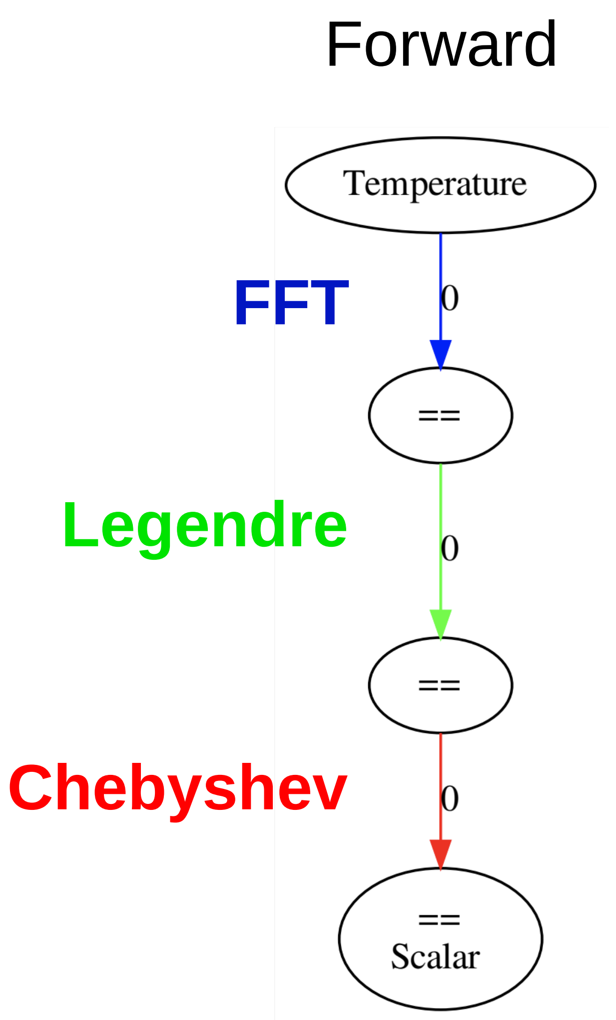
Forward Transforms



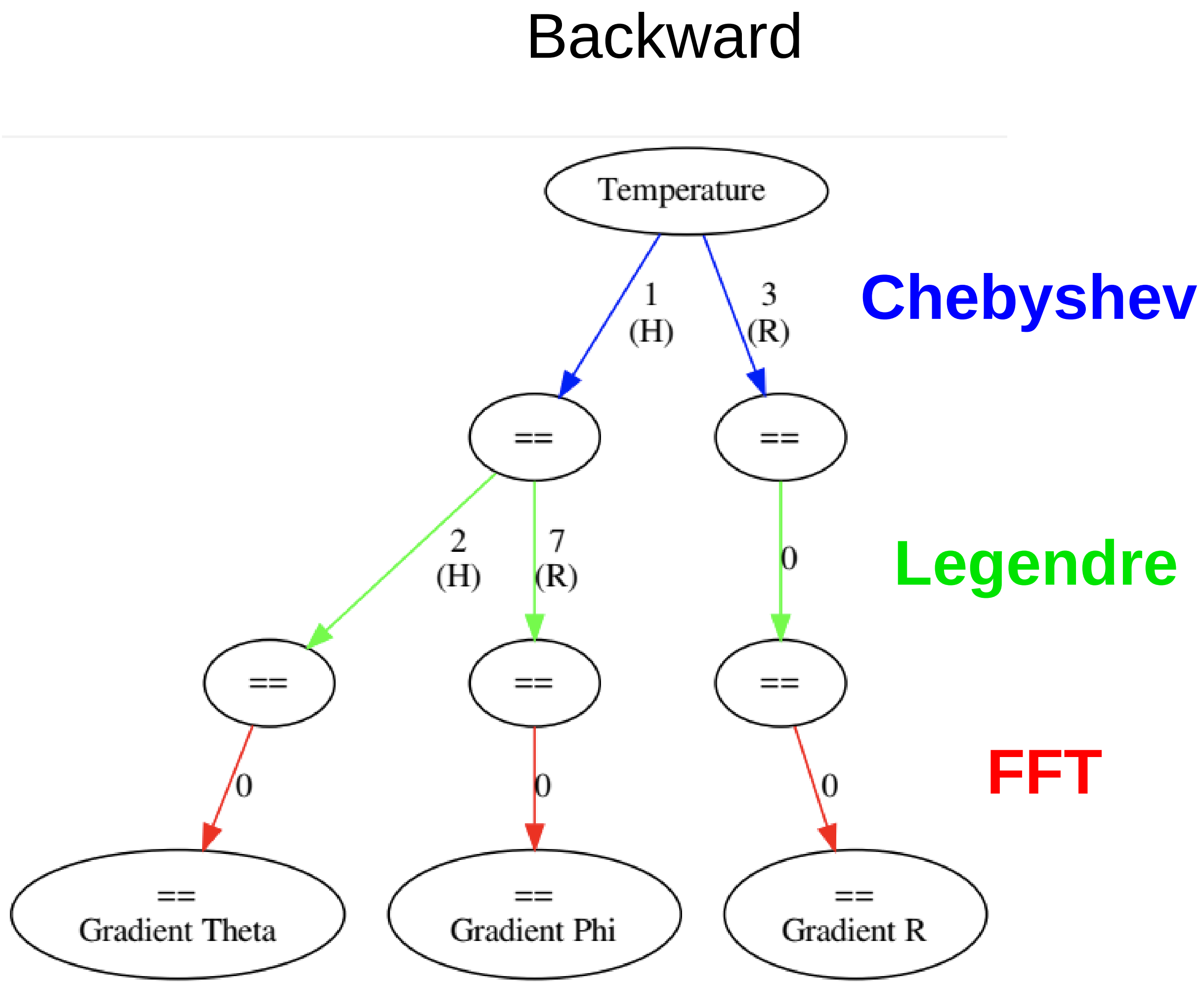
Backward Transforms



Transforms



Forward
Temperature
FFT
0
==
Legendre
0
==
Chebyshev
0
==
Scalar



Backward
Temperature
1
(H)
3
(R)
Chebyshev
==
==
2
(H)
7
(R)
0
Legendre
==
==
==
0
0
0
FFT
==
Gradient Theta
==
Gradient Phi
==
Gradient R

Mixed Fourier (C2R, R2C) Transforms

Steps completed:

1. cuFFT with FFTW interface
2. cuFFT interface
3. CUDA stream and sub batches

Chebyshev (R2R) Transforms

Steps completed:

1. Convert R2R to R2C/C2R
2. cuFFT with FFTW interface
3. cuFFT interface
4. Cuda stream and sub batches

Associated Legendre Transforms

Steps completed:

1. Replace matrix multiplication (Eigen) with cuBLAS (no speed up because of memcopy)
2. Allocate and populate operators on device before computation
3. Use streams to overlap data transfer and computation
4. Multiple cuBLAS handles and streams so transforms are computed in parallel with C++ native threads

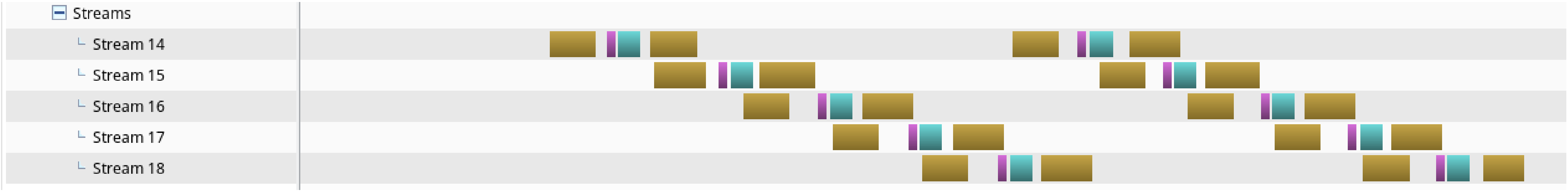
Timing: Mixed Fourier

Resolution: spectral 128 (physical 256), batch size: 1000

1.5 x faster compared to a single core

Comments:

- Higher resolution with spectral 256 (physical 512) speedup the same
- Smaller is slower (significantly, up to 10 x)



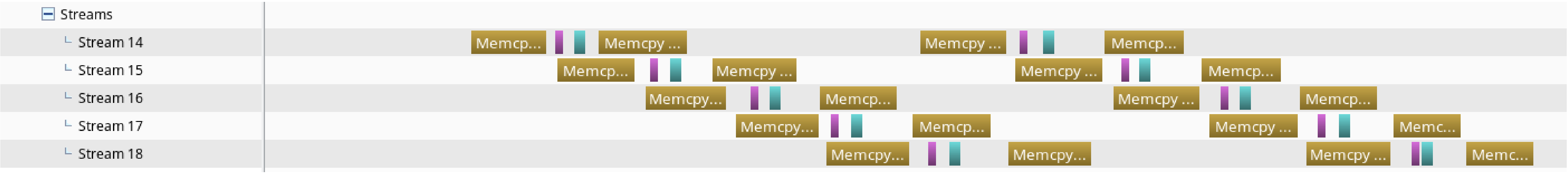
Timing: Chebyshev

Resolution: spectral 128 (physical 256), batch size: 1000

1.5 x slower compared to a single core

Comments:

- Higher resolution with spectral 256 (physical 512) speed down the same



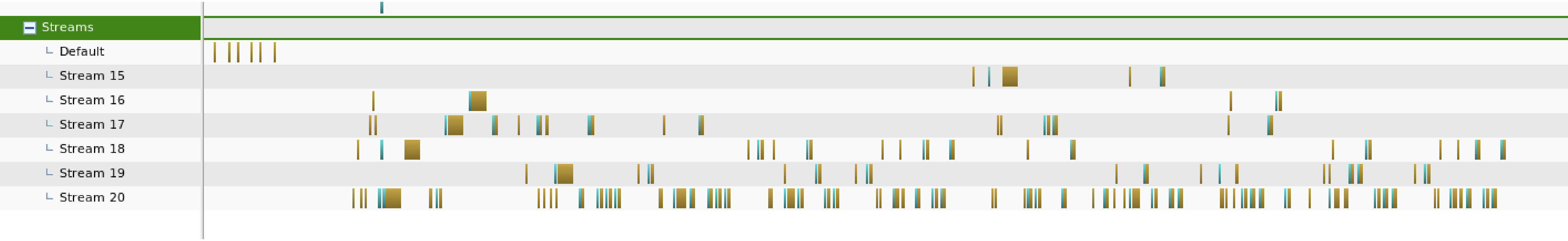
Timing: Legendre

Resolution: spectral 400 (physical 800), batch size: 20

9 x faster compared to a single core

Comments:

- GPU occupancy is low
- Trying to understand stream 20 calls



Performance discussion

- Strongly depends on resolution
- Up to factors of 3 depending on the number of streams and the sub-batching
- Memory management needs to be done higher in the call tree

Next Steps:

- More detailed performance analysis (streams timeline)
- Data management for full 3D transforms
- Clean code design

- Associated Legendre
  - same operation for missing operators (copy/paste)
- Mixed FFT
  - same operation for missing operators
  - Pre/post processing needs to be ported
- Chebyshev
  - Pre/post processing needs to be ported

Feedback

- cuFFT should have a DCT! ;)
- GPU port made easier high level abstractions
- Method for choosing the number of streams
- Mixed type multiplication for cuBLAS