Swiss National Supercomputing Centre

Improving the Performance of COSMO-CLM

Matthew Cordery, William Sawyer
Swiss National Supercomputing Centre

Ulrich Schättler
Deutscher Wetterdienst

What is COSMO-CLM?

- COSMO is an operational non-hydrostatic meso-to-micro scale NWP system.
  - Used by MeteoSwiss, DWD,etc.
- The COSMO model in CLimate Mode (COSMO-CLM or CCLM) is a non-hydrostatic regional climate model.

CSCS

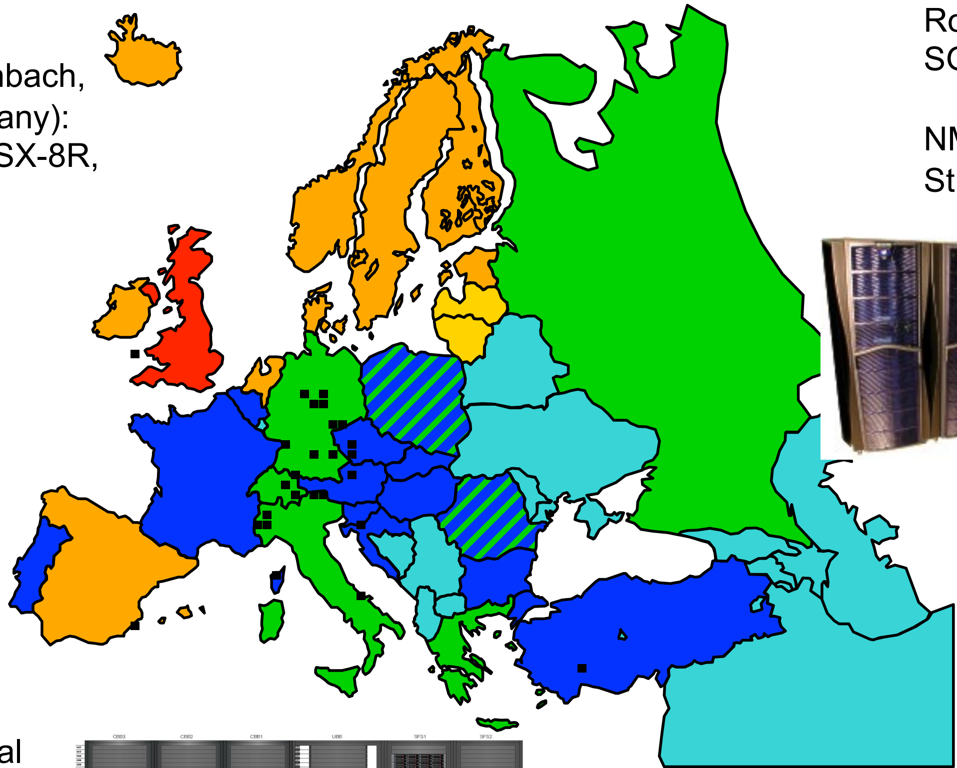
Swiss National Supercomputing Centre

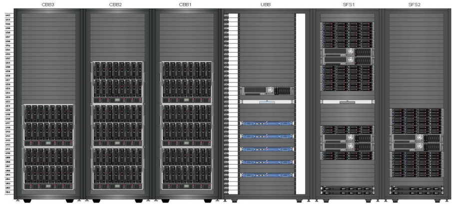
COSMO NWP Applications

DWD (Offenbach, Germany): NEC SX-8R, SX-9

MeteoSwiss:
Cray XT4: COSMO-7 and COSMO-2 use 800+4 MPI-Tasks on 402 out of 448 dual core AMD nodes

ARPA-SIM (Bologna, Italy):
Linux-Intel x86-64 Cluster for testing (uses 56 of 120 cores)





USAM (Rome, Italy):
HP Linux Cluster
XEON biproc quadcore
System in preparation

Roshydromet (Moscow, Russia),
SGI

NMA (Bucharest, Romania):
Still in planning / procurement phase

IMGW (Warsawa, Poland):
SGI Origin 3800:
uses 88 of 100 nodes

ARPA-SIM (Bologna, Italy):
IBM pwr5: up to 160 of 512 nodes at CINECA

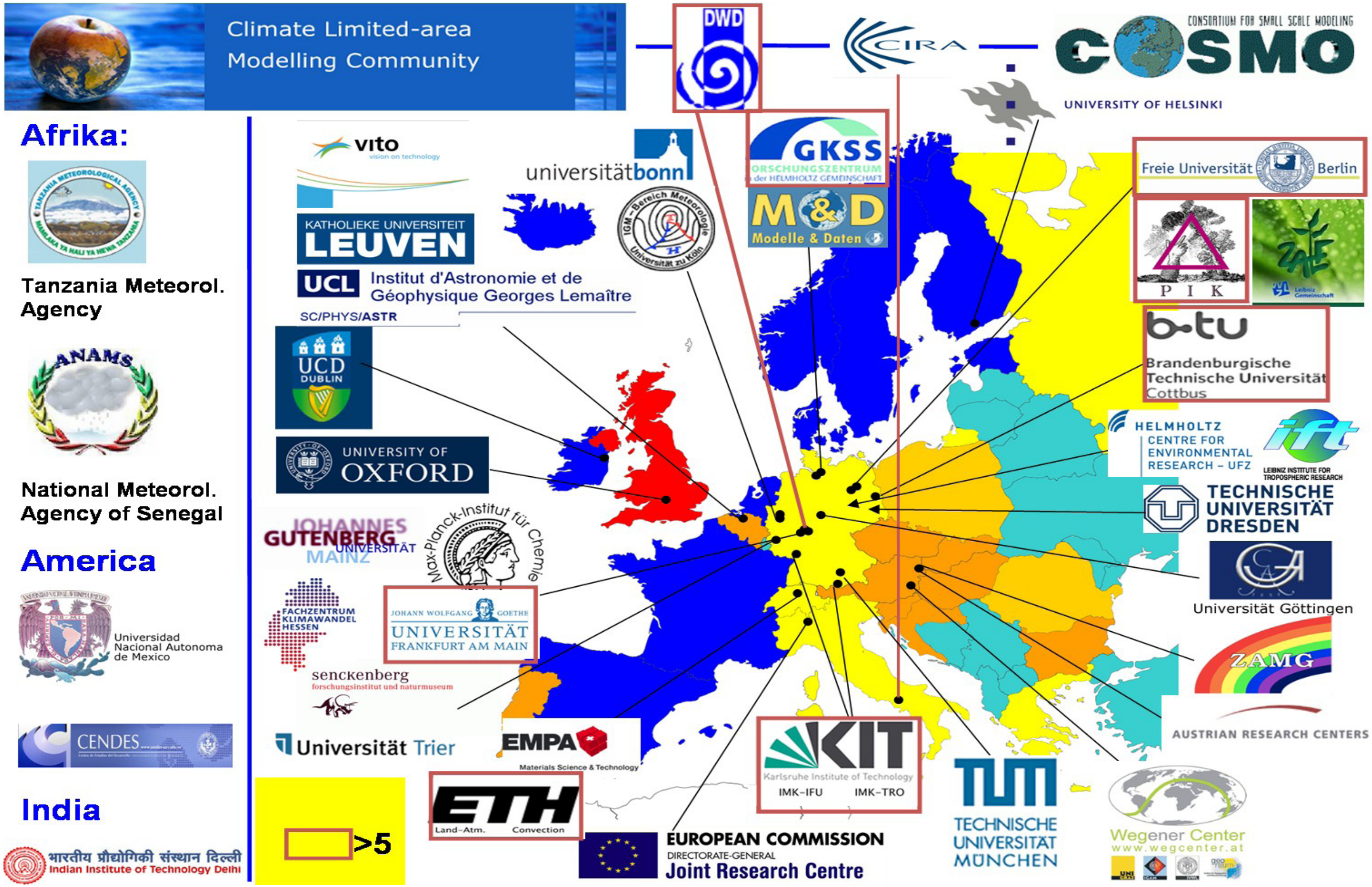
COSMO-LEPS (at ECMWF):
running on ECMWF pwr6 as member-state time-critical application

HNMS (Athens, Greece):
IBM pwr4: 120 of 256 nodes

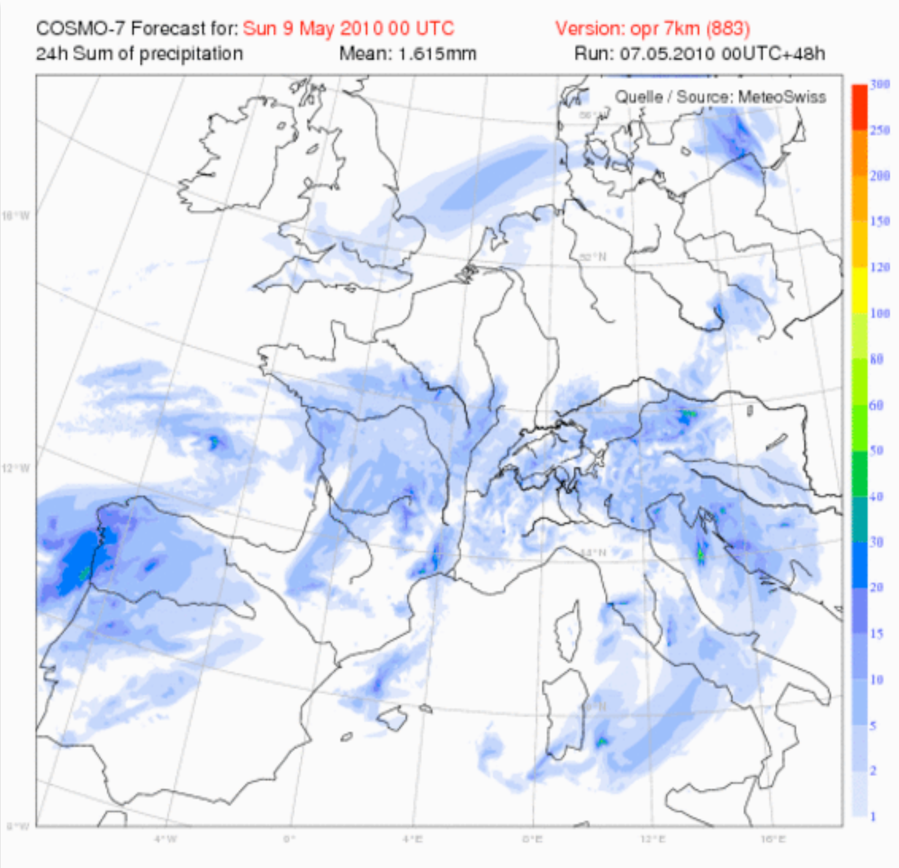
CSCS

Swiss National Supercomputing Centre

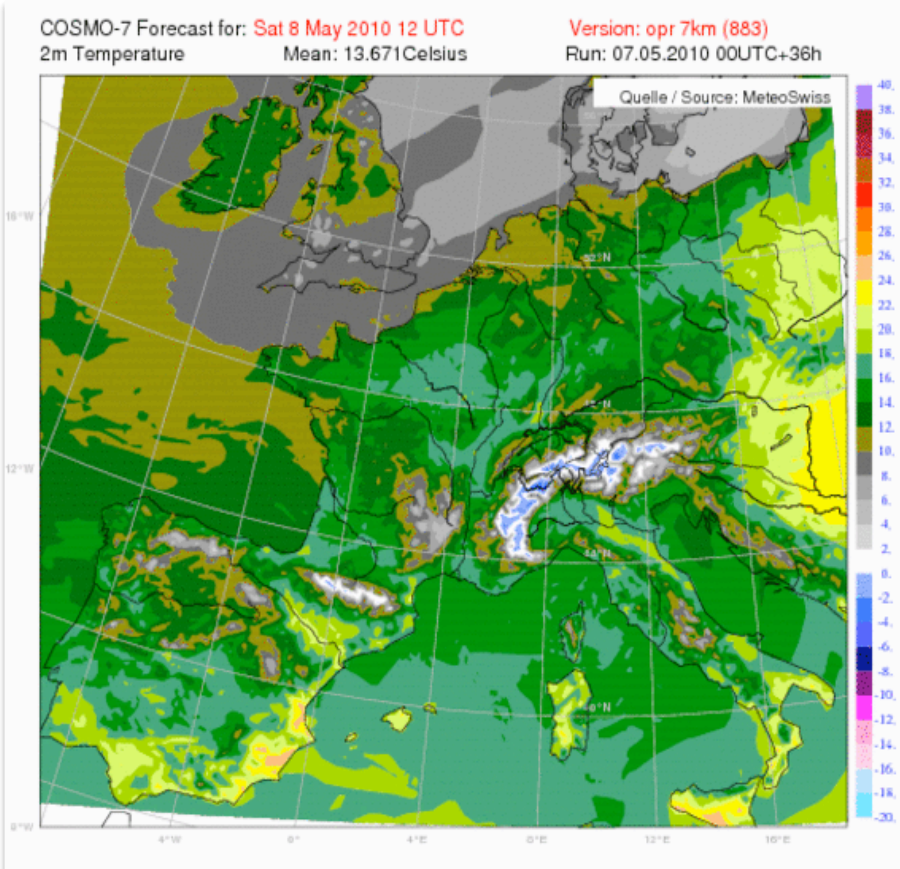
CLM Community 2010



Sample NWP output



Precipitation



2m Temperature

Goal

- An operation NWP prediction forecast for 24 hours make take half an hour of CPU time
- By extension, a 100 year climate simulation on a similar grid would take *several months* of CPU time.
- Our goal is to examine different hybrid programming schemes that will provide a substantial CPU time savings without requiring extensive use of machine resources.
- First step:
  - Examine mixed programming model using MPI and OpenMP.

- The computational grid is a 3D rotated-latitude/longitude structured grid.
- Communications are through 2- or 3-line halo exchanges.
- Many loops are of the form

```
do k = 1, ke
        do j = 1, je
                do i = 1, ie
                        …
                end do
        end do
end do
```

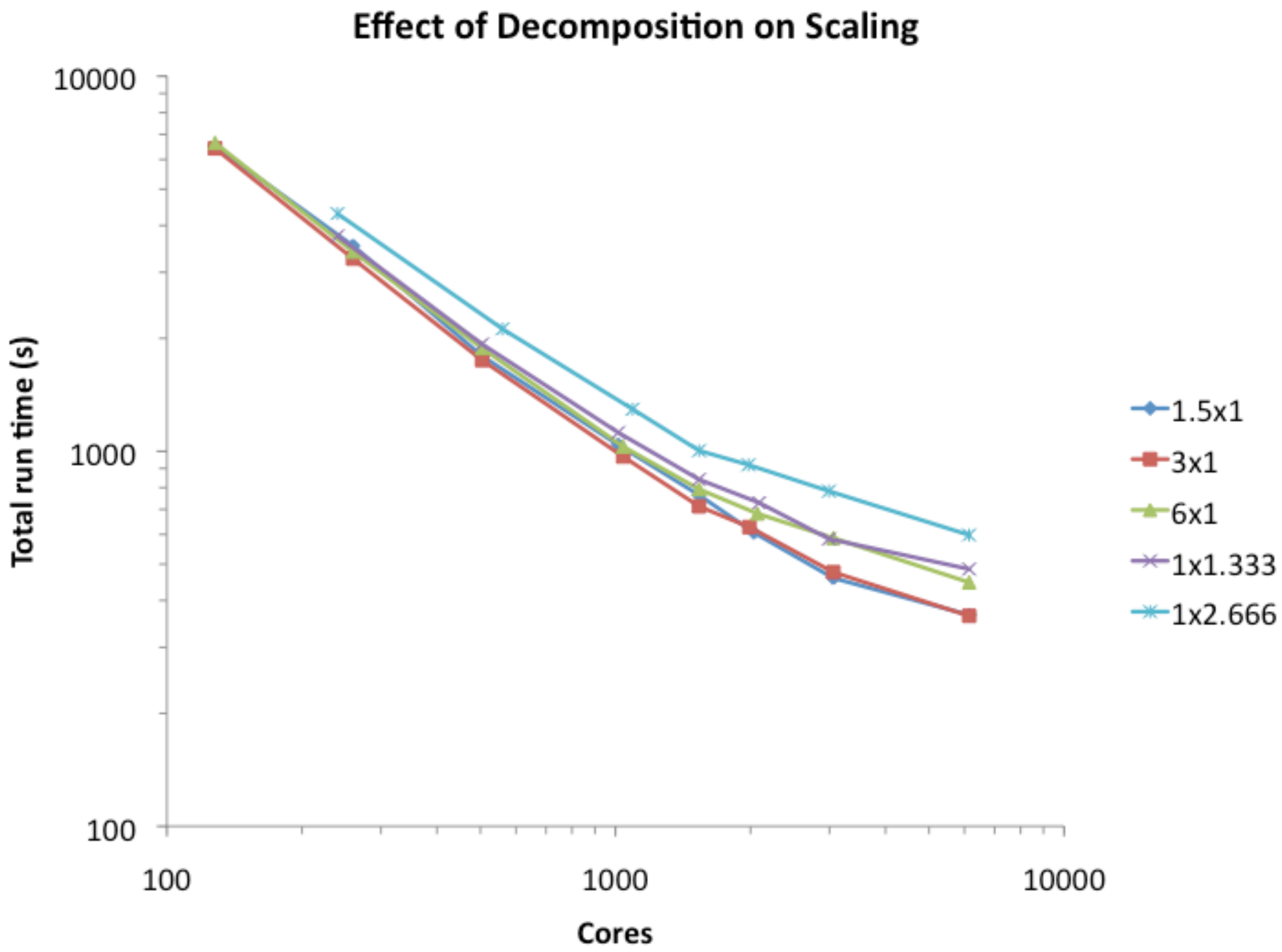
- Some 2D and 4D arrays, but of the same basic structure.

COSMO characteristics

Main computational region is a time stepping loop over

- a <u>dynamical core</u> that solves for fluid motion
- a <u>physical core</u> that computes radiation transfer, precipitation, etc.

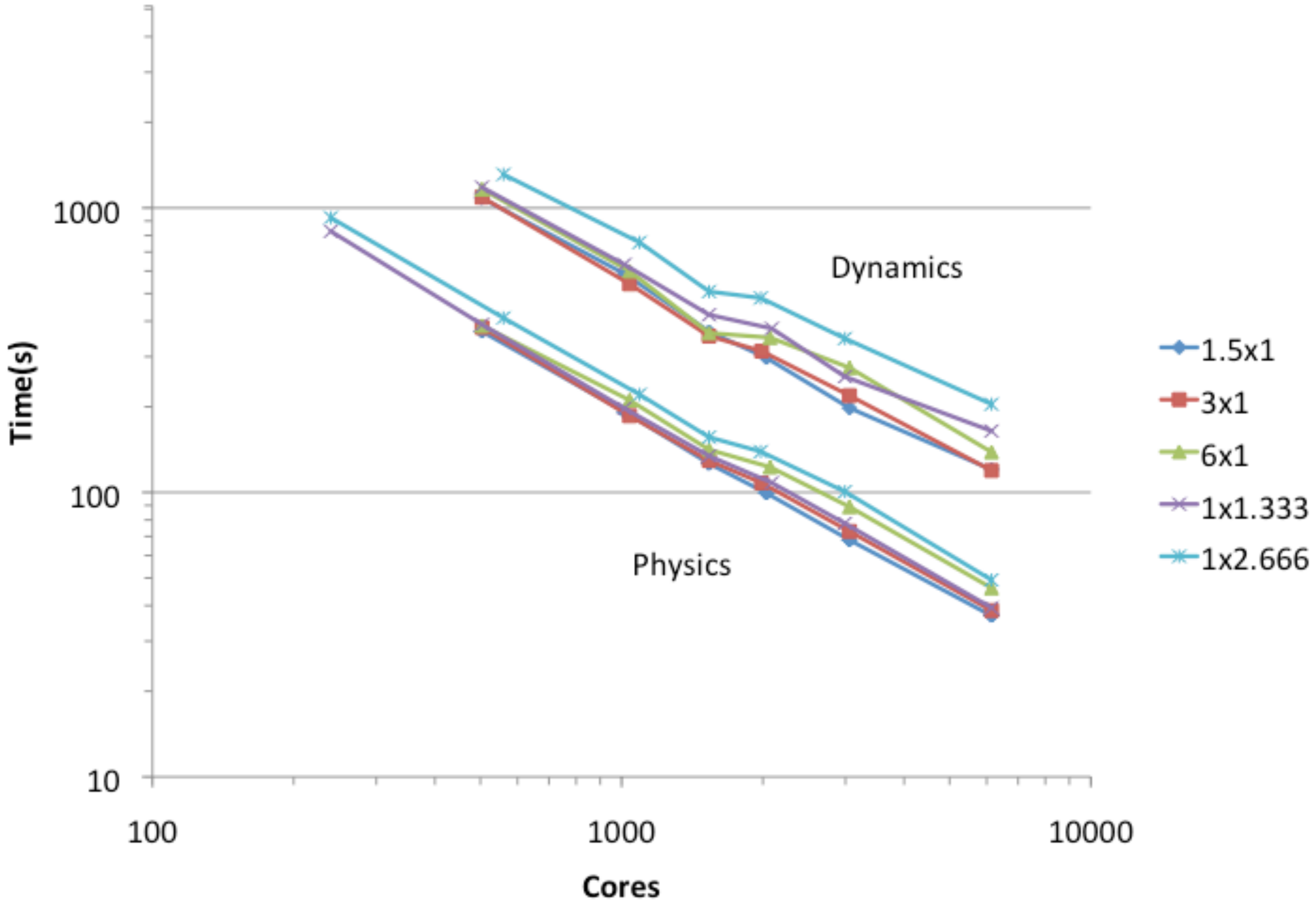
- Examine scaling of 1-km to elucidate 'hotspots'.
  - 1142 longitude points
  - 765 latitude points
  - 90 vertical points
  - 3 halo grid points
  - 4 I/O tasks
  - 1 hour of simulated time


Effect of Decomposition on Scaling
Total run time (s)
10000
1000
100
100
1000
10000
Cores
1.5x1
3x1
6x1
1x1.333
1x2.666

1km observations

- Computational intensity is low (<1) for most top routines and the cache hit ratio is > 95%.
- Good scaling out to several thousand cores, though performance is rather low.
- Implies algorithms are memory-bandwidth limited.

Scaling of dynamical and physical cores



Scaling observations

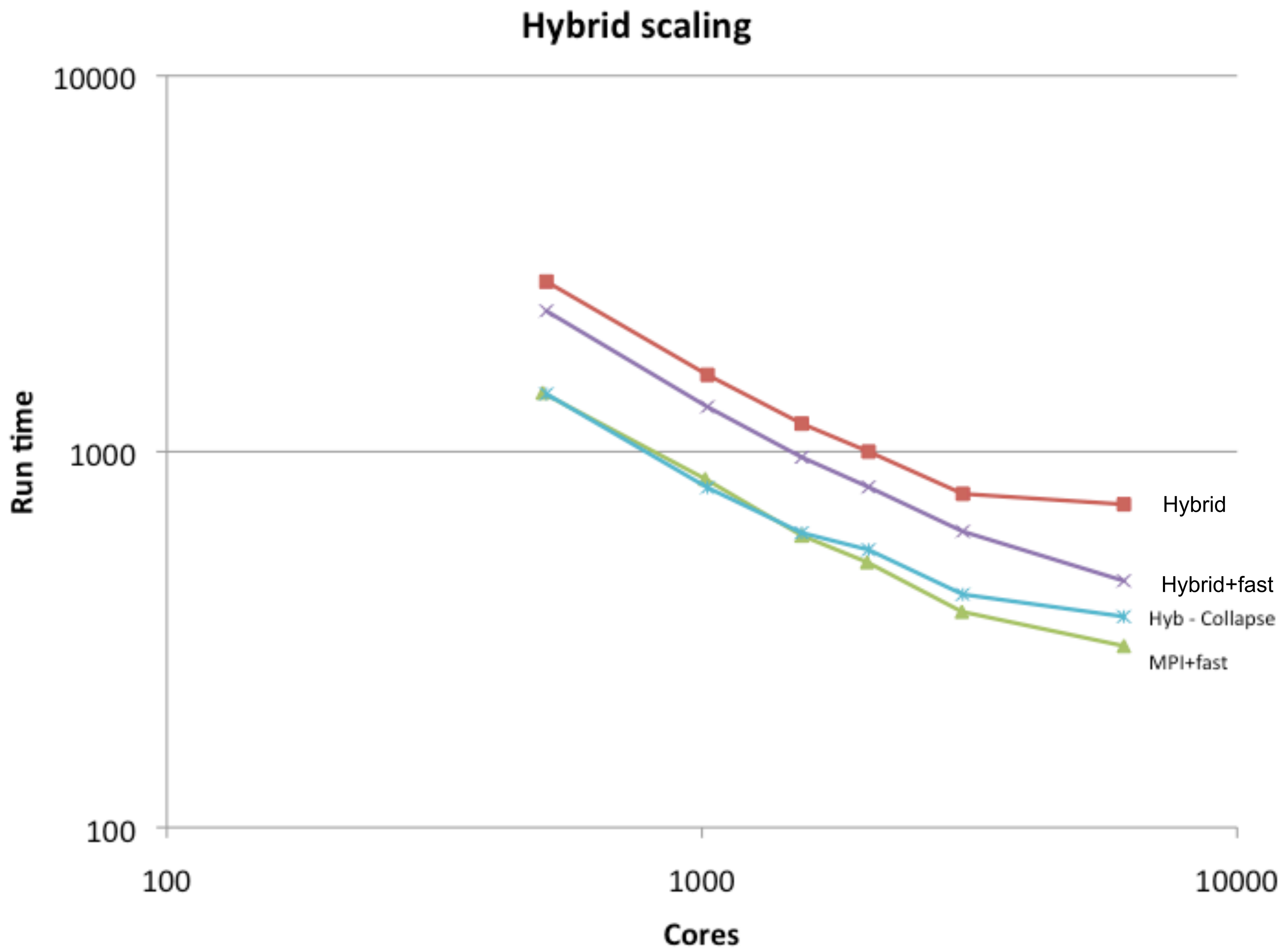
- Physical and dynamical cores scale well out to > 2000 cores
- Dynamical core takes approximately 3x more time than the physical core, regardless of core count.
- I/O and MPI communications are not limiting factors

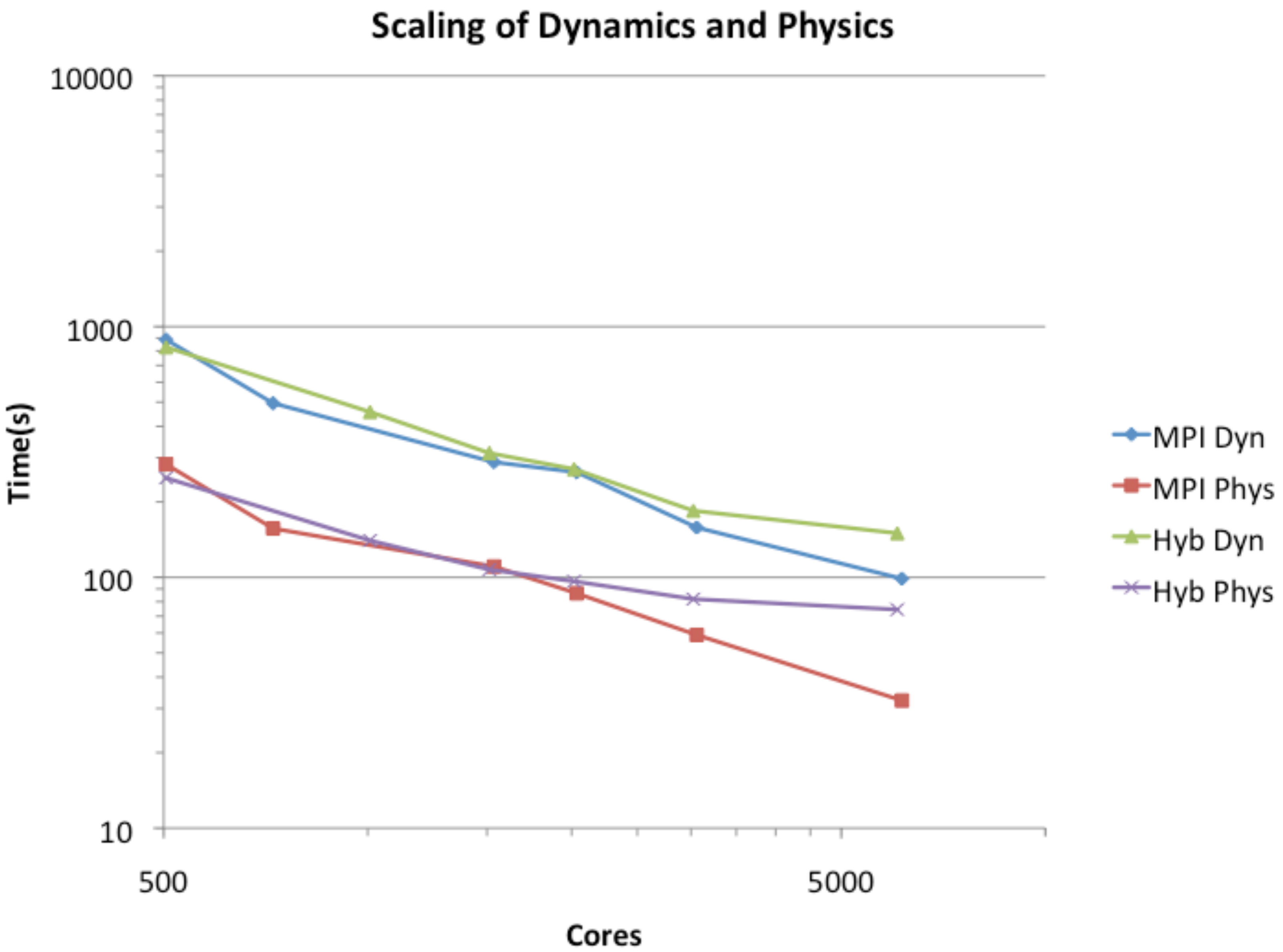
Profiling indicates that during the main time stepping loop

- I/O and MPI communications are not limiting factors
- The dynamical core requires 3x more time than the physical core, regardless of core count

Hybridization path

- Most computational work is encapsulated in multiply-nested loops in multiple subroutines that are called from a main driver loop.
- Most outer loops are over the number of levels, inner loops over latitude/longitude.
- Insert OpenMP PARALLEL DO directives on outermost loop
  - Also attempted use of OpenMP 3.0 COLLAPSE directive.
  - Over 600 directives inserted.
  - Also enabled use of SSE instructions on all routines (previously only used on some routines).


Hybrid scaling
10000
1000
100
Run time
100
1000
10000
Cores
Hybrid
Hybrid+fast
Hyb - Collapse
MPI+fast

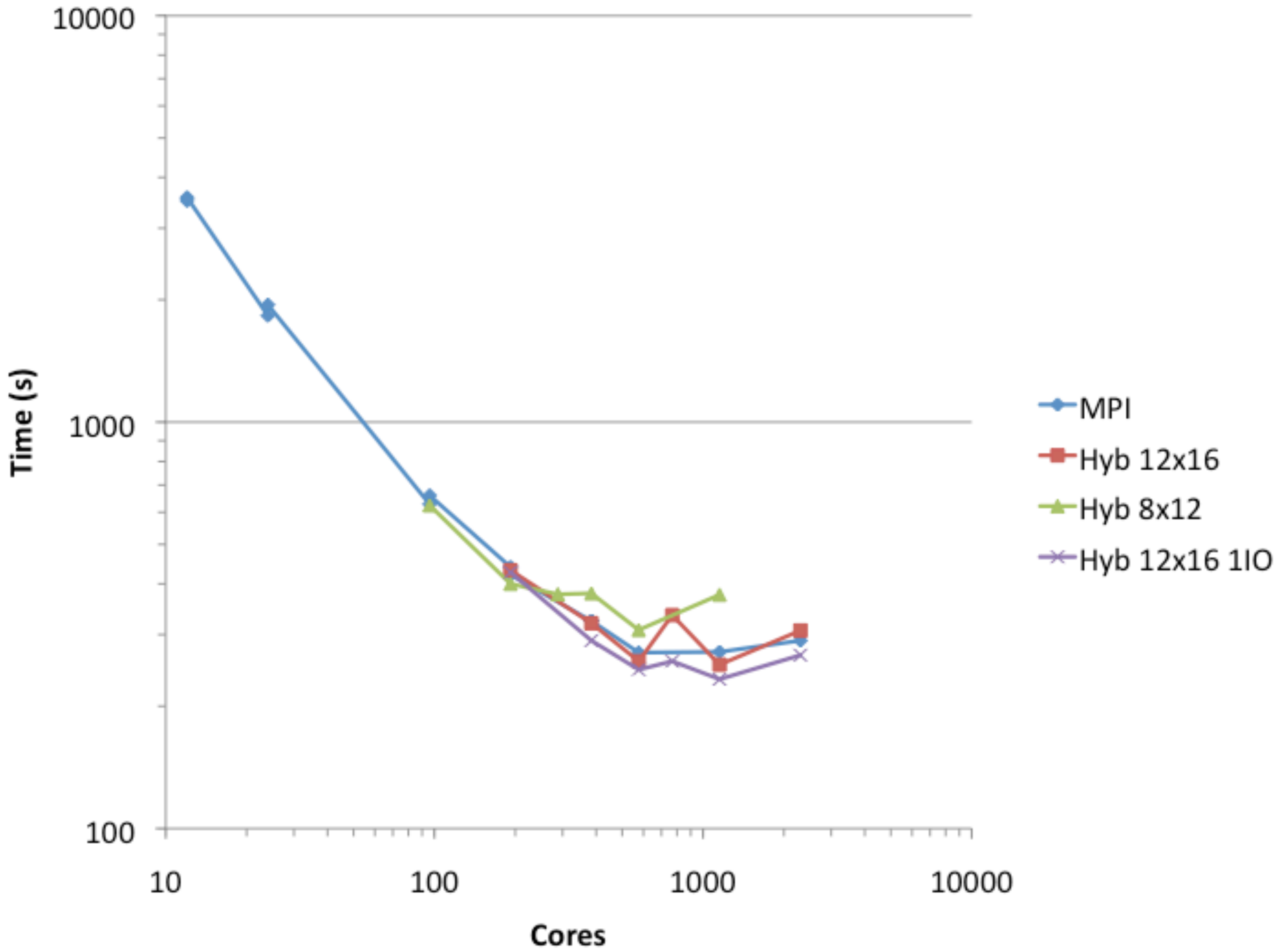

Scaling of Dynamics and Physics
10000
1000
100
10
Time(s)
500
5000
Cores
MPI Dyn
MPI Phys
Hyb Dyn
Hyb Phys

Observations

- Important to have as much of the code as possible compiled to use SSE instructions.
- Important not to overuse COLLAPSE directive which may interfere with compiler optimization
- Code runs approximately as fast as the MPI only version for most core counts.
- Dynamical and physical cores scale well, though the physical core shows a much more pronounced loss of scaling beyond 4 threads.
- Reducing the number of I/O tasks improves performance (why?) and reduces idle cores.

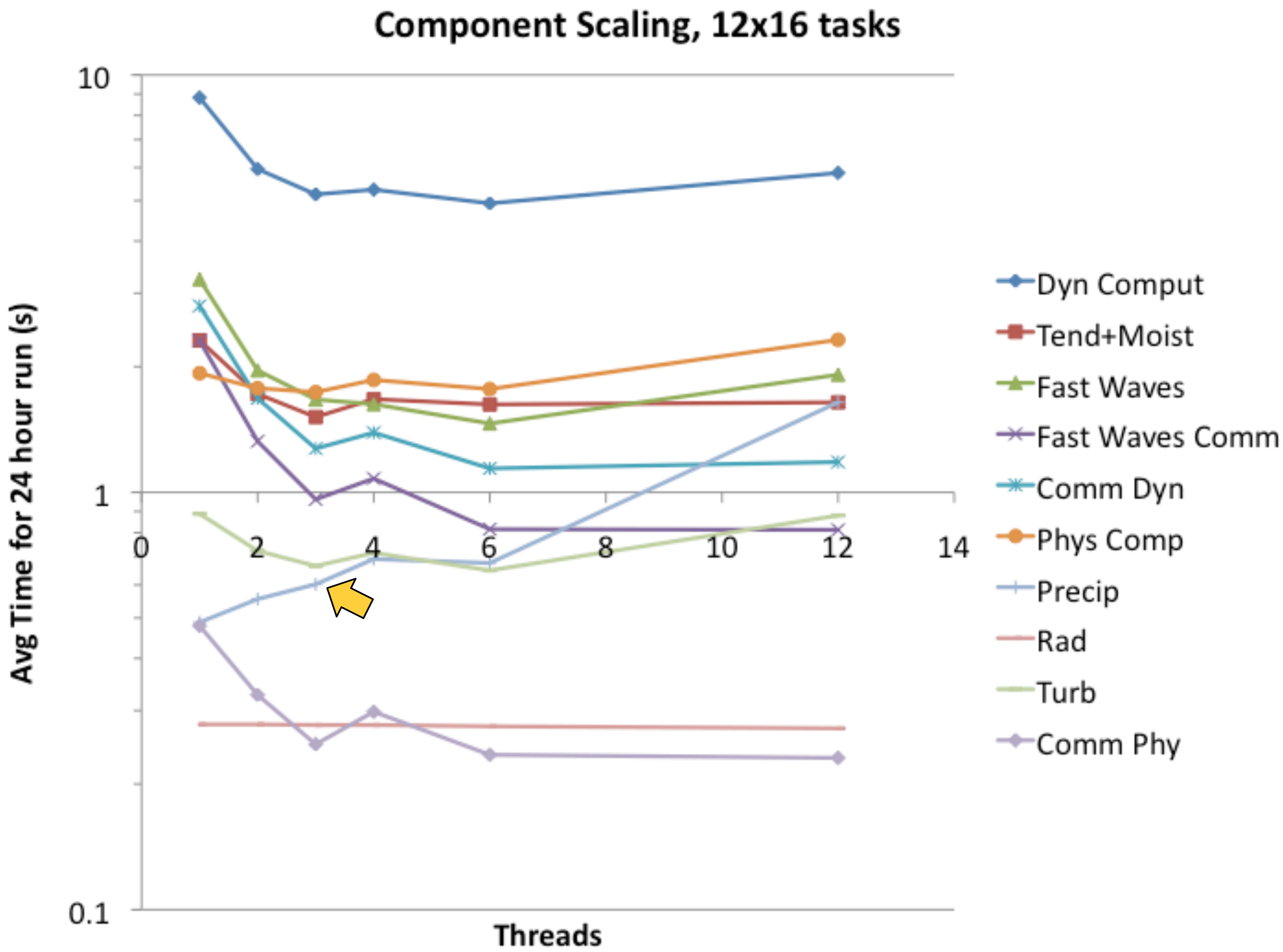
'Climate grid'

- In order to examine the results more similar to what will be used for a 100 year climate science run:
- 102 latitudes
- 102 longitudes
- 60 height levels
- 1 I/O task
- Run for 24 simulated hours.


10000
1000
100
Time (s)
10
100
1000
10000
Cores
MPI
Hyb 12x16
Hyb 8x12
Hyb 12x16 1IO

Observations

- Relatively weak (<10%) improvement over standard MPI code.
- Best results at 3 or 6 threads.
- Performance decrease for 4 threads where threads from one MPI task will span a node.
- Performance decrease going from six to twelve threads indicates still some performance bottlenecks.


Component Scaling, 12x16 tasks
10
1
0.1
0
2
4
6
8
10
12
14
Avg Time for 24 hour run (s)
Threads
Dyn Comput
Tend+Moist
Fast Waves
Fast Waves Comm
Comm Dyn
Phys Comp
Precip
Rad
Turb
Comm Phy

Summary

- Loop level parallelism can achieve some modest performance gains
  - Can require many threaded loops -> OpenMP overhead
  - Can require a lot more software engineering to maintain
    - Introducing new private variables into old loops
    - Introducing new physics that needs to be threaded
  - Can be problematic when dealing with threads that include arrays created using Fortran allocate statement.
- Next task is to examine threading and blocking at a higher level. This will require more extensive work to the code.
  - Improve data re-use
  - Reduce memory footprint
- Investigate new algorithms