Recent developments in the US

Exascale Technologies and Innovation in HPC for Climate Models
Hamburg GERMANY

V. Balaji
with contributions from Rich Loft, Mark Govett, Chris Kerr, Kareem Sorathia

NOAA/GFDL and Princeton University

17 March 2014

Outline

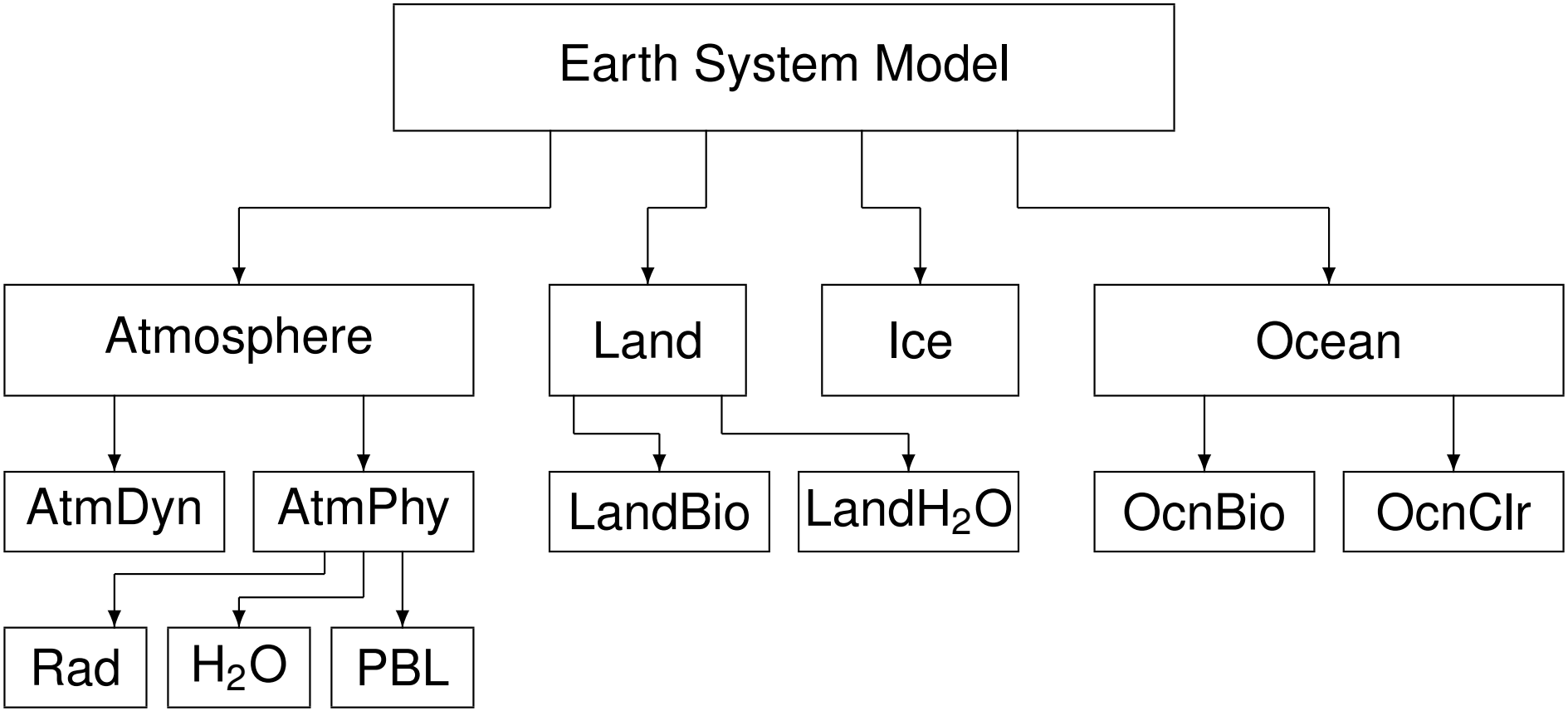
Outline

Climate modeling, a computational profile

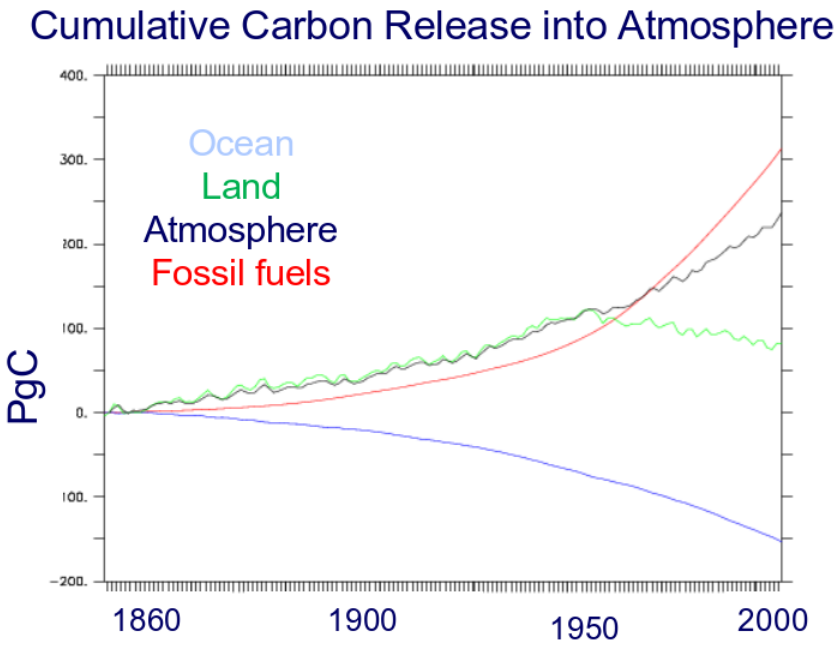
- Intrinsic variability at all timescales from minutes to millennia; distinguishing natural from forced variability is a key challenge.
- coupled multi-scale multi-physics modeling;
- physics components have predictable data dependencies associated with grids;
- Adding processes and components improves scientific understanding;
- New physics and higher process fidelity at higher resolution;
- Ensemble methods to sample uncertainty (ICEs, PPEs, MMEs...)
- algorithms generally possess weak scalability.

In sum, climate modeling requires long-term integrations of weakly-scaling I/O and memory-bound models of enormous complexity.

Earth System Model Architecture



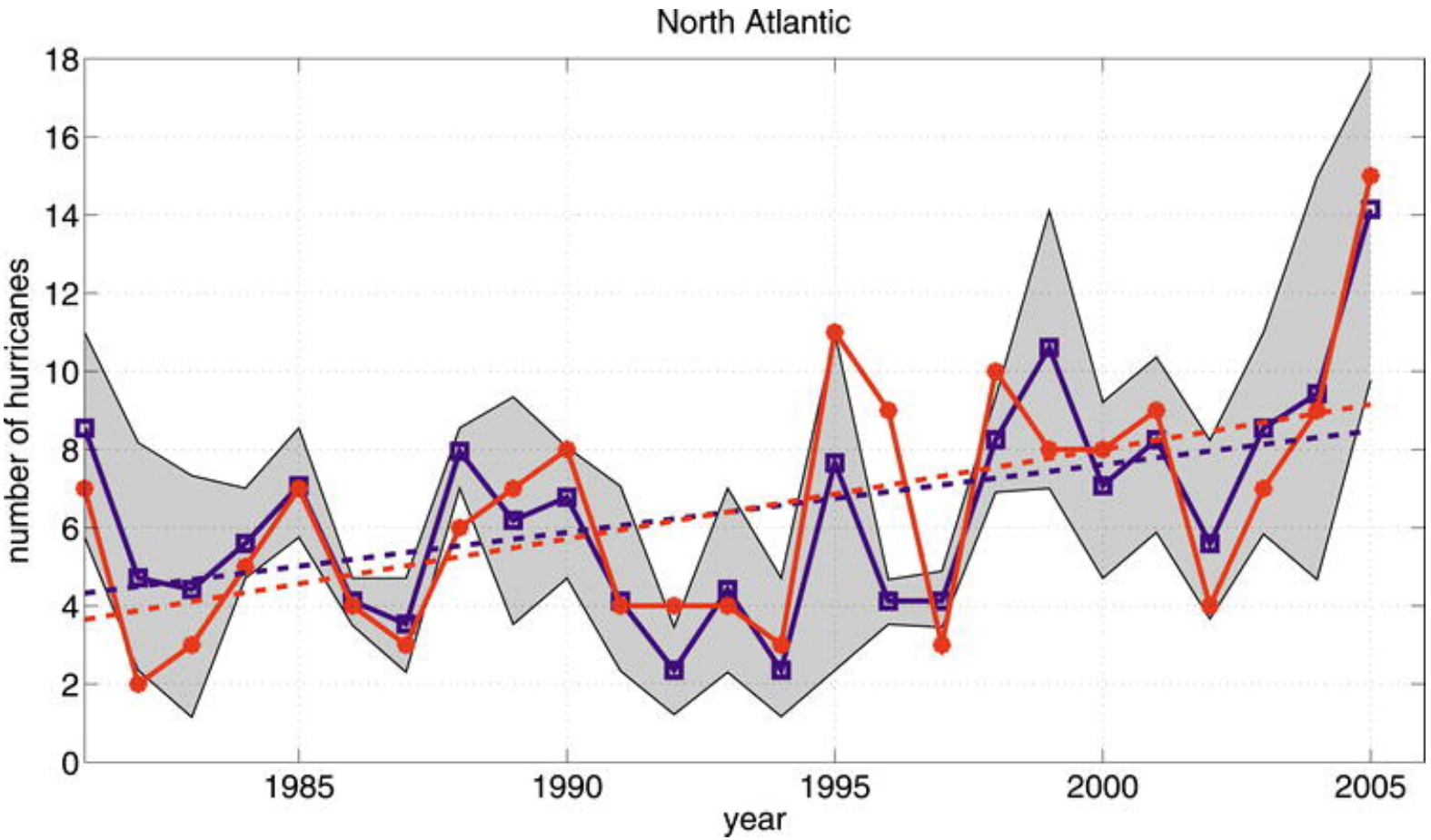
Carbon sources and sinks



- Land carbon fluxes dominant before 1960; then trend changes sign.
- Fossil fuels dominant contemporary source.
- Ocean uptake scales with pCO_2.

Figure courtesy Ron Stouffer, NOAA/GFDL; pre-publication.

Interannual variability of hurricane frequency



Interannual variability of W. Atlantic hurricane number from 1981-2005 in the C180 runs. (Figure 7 from Zhao and Held 2009).

Outline

The hardware jungle

Upcoming hardware roadmap looks daunting! GPUs, MICs, DSPs, and many other TLAs. . .

- Intel/AMD x86 host coupled to NVIDIA GPU device: host launches many (32, 64, . . .) parallel threads on device.
- AMD Opteron/Firestream + AMD GPU: similar in structure to above, 64-bit FP registers.
- Firestream + AMD APU: integrated on-chip co-processor. Physical memory shared between host and device.
- Intel core + integrated Intel GPU (Ivy Bridge):.Intel version of above.
- Intel core + Intel MIC: still a host/device model but threads can be programmed in OpenMP. (Unhosted in Knights Landing?)

The hardware jungle

More far-fetched stuff . . .

- NVIDIA Denver: ARM processor (low-power) + NVIDIA GPU: shared-memory.
- Texas Instruments! ARM + DSPs
- Convey: x86 + FPGAs. shared virtual memory, permits RMA.
- Tilera, FeiTeng: stream accelerators.
- BG/Q: CPU only, with OpenMP and vector instructions. Reportedly 8x faster than P. Memory/core is relatively small.

Some material above adapted from Wolfe (2012), in HPCWire.

The software zoo

It is unlikely that we will program codes with $10^6 - 10^9$ MPI ranks: it will be MPI+X. Solve for X . . .

- CUDA and CUDA-Fortran: proprietary for NVIDIA GPUs. Invasive and pervasive.
- OpenCL: proposed standard for MICs that can also be implemented for GPUs.
- ACC from Portland Group: accelerator directives that will be treated as comments on non-compliant hardware. Now being proposed as a new standard OpenACC.
- PGAS languages: Co-Array Fortran, UPC, a host of proprietary languages.

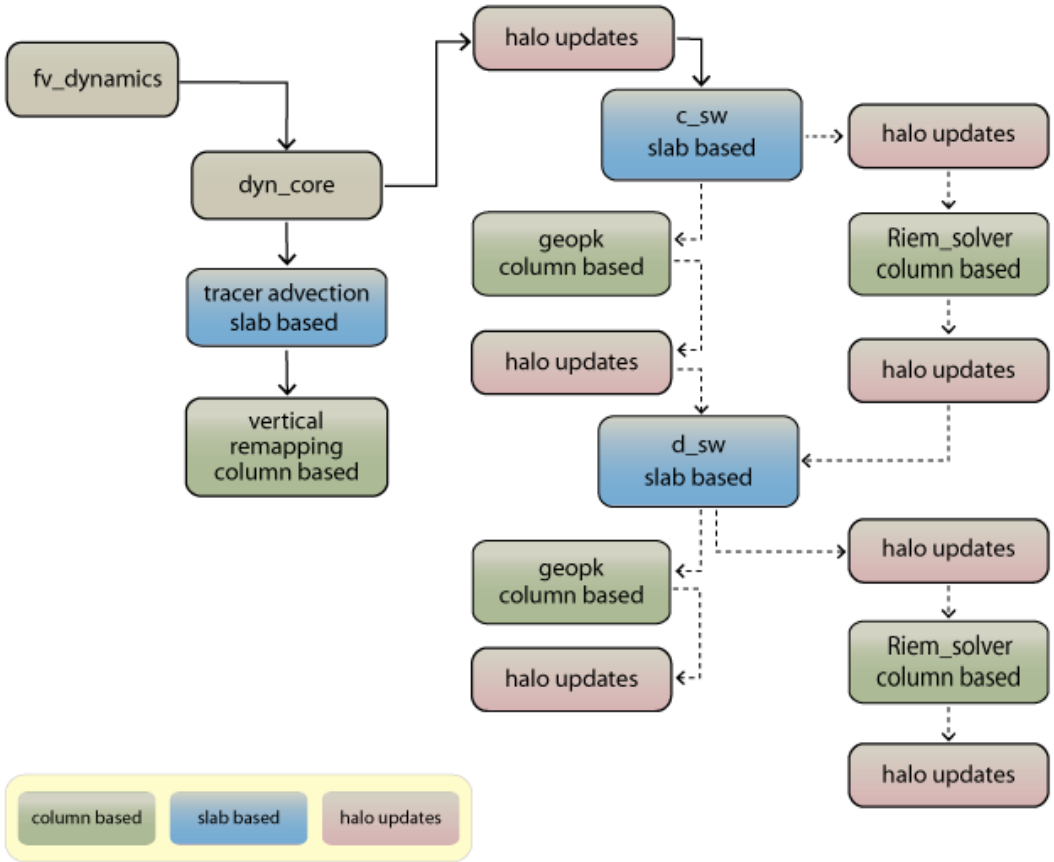
The software zoo

GFDL is taking a conservative approach:

- it looks like it will be a mix of MPI, threads, and vectors.
- Developing a three-level abstraction for parallelism: components, domains, blocks. Kernels work on blocks and must have vectorizing inner loops.
- Recommendation: sit tight, make sure MPI+OpenMP works well, write vector-friendly loops, reduce memory footprint, offload I/O.
- Other concerns:
  - Irreproducible computation
  - Tools for analyzing performance.
  - Debugging at scale.

Recent experience on Titan, Stampede and Mira reaffirm this approach.

Analysis of dycore architecture for GPU/MIC



Study of code for MPI, threads, vectors. (Chris Kerr, Zhi, Kareem Sorathia (NASA), Duane Rosenberg (ORNL), Eric Dolven (Cray)...)

Blocking the dycore for GPU/MIC

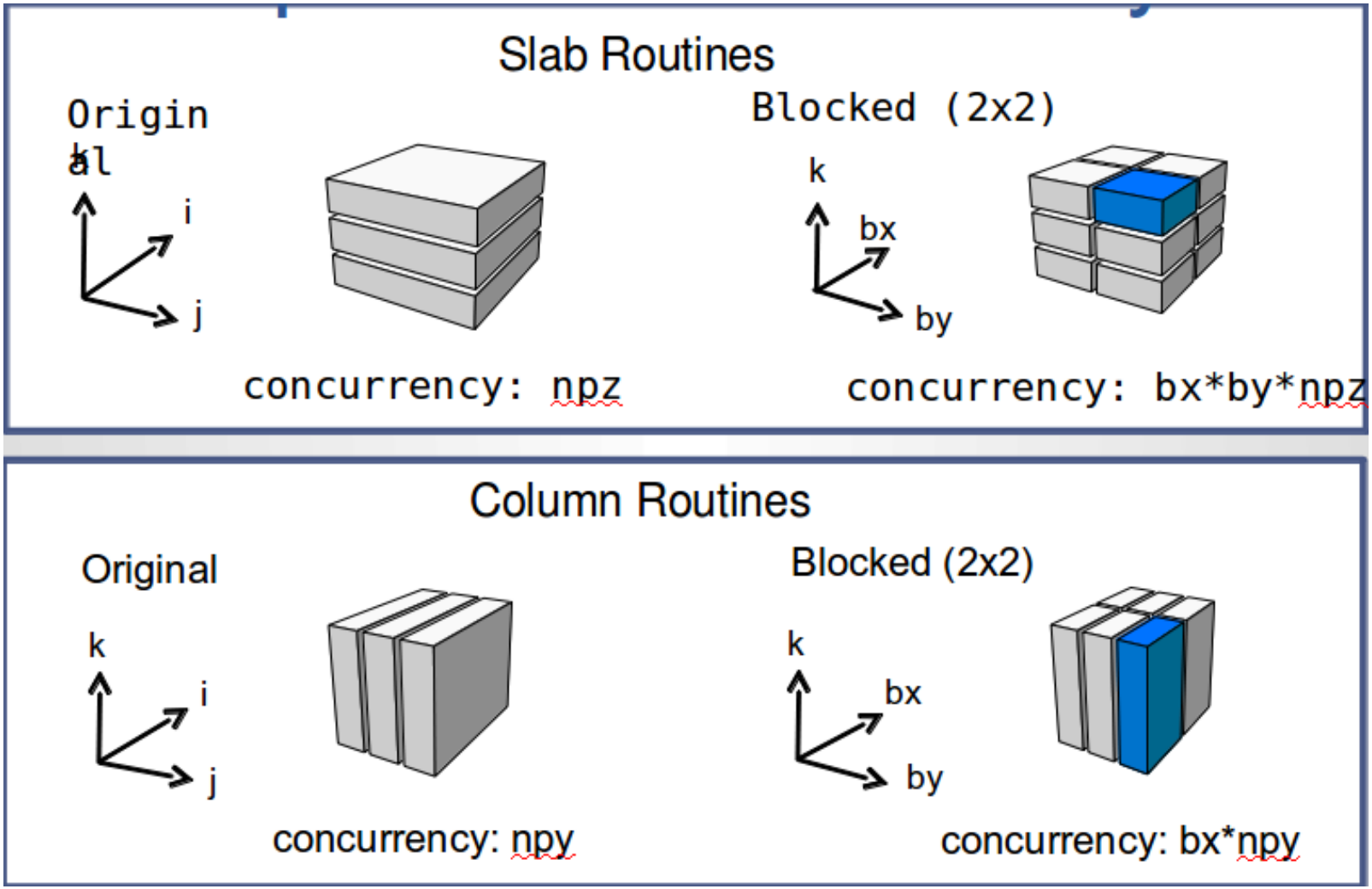


Figure courtesy Kareem Sorathia (NASA). Inner loops on `i` are retained for vectorization.

Performance summary: Xeon-SNB vs Xeon-Phi

Phi “speedup” over SNB:

- Overall: 0.73
- Communication: 0.34
- All Computation: 0.86
- Top 4: 0.996

Coding issues:

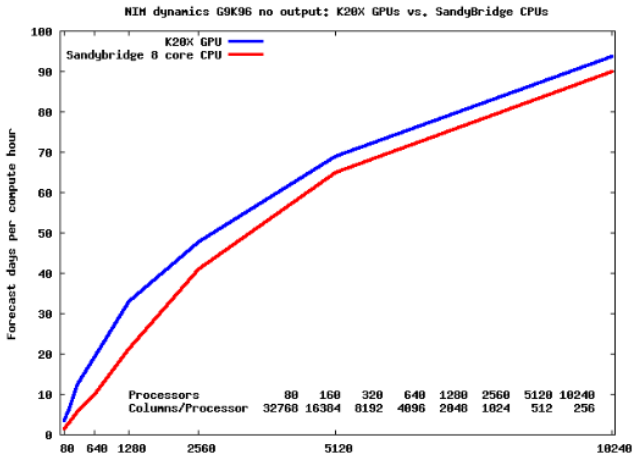
- Vector performance very hard to achieve, even with padding halos for alignment.
- Loop unrolling/stripmining/etc needs to be done by hand.
- Better performance analysis tools needed.

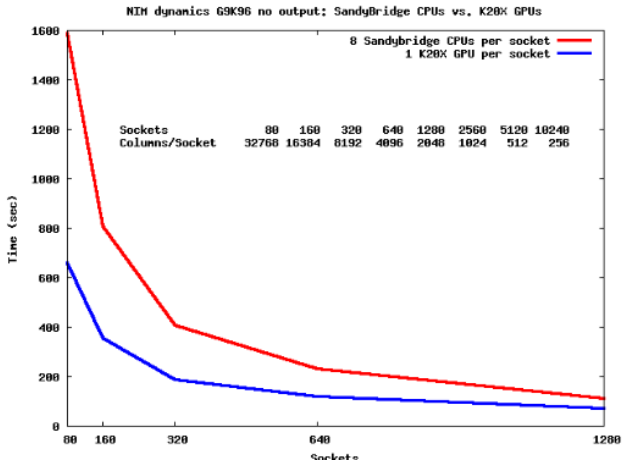
Courtesy Kareem Sorathia. (For details, see Chris Kerr’s talk.)

Results from NIM icosahedral dycore: SNB vs GPU

NIM Dynamics: GPU versus Intel-SB

- Single source code optimized for CPU, MIC & GPU
  - OpenMP directives for CPU & MIC
  - OpenACC, F2C-ACC for NVIDIA GPU
- 15 KM model, 96 levels, single-precision
  - Strong scaling: 80 - 10240 GPUs
  - GPU 2-3x faster than CPU socket for 8192 columns





Courtesy Mark Govett, NOAA/ESRL. (For more, see Will Sawyer's talk).

OpenACC

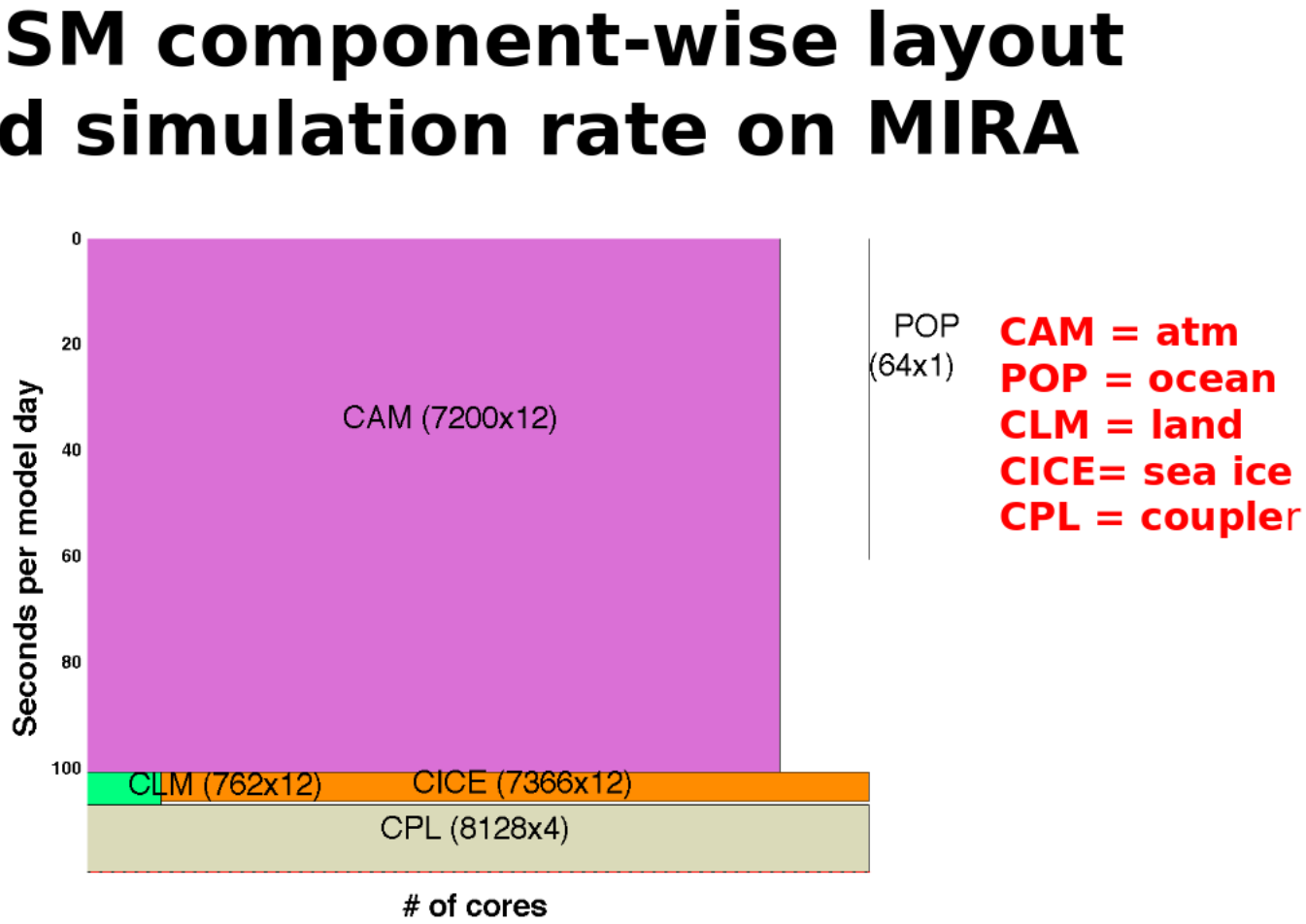
"Very easy to program...", "...is the future", "...will work with vendors to improve performance." – Mark Govett.

```
!ACC$REGION(<64>,<10242>,<flxhi:none,local>) BEGIN
!$acc parallel num_gangs(ihe-ips+1) vector_length(64)
!ACC$DO PARALLEL(1)
!$acc loop gang
    do ipn=ips,ihe
!ACC$DO VECTOR(1)
!$acc loop vector
      do k=1,nvl
        flxhi(k) = vnorm(k,edg,ipn)*dp_edg(k,edg,ipn)
```

Can merge gang and vector on same axis:

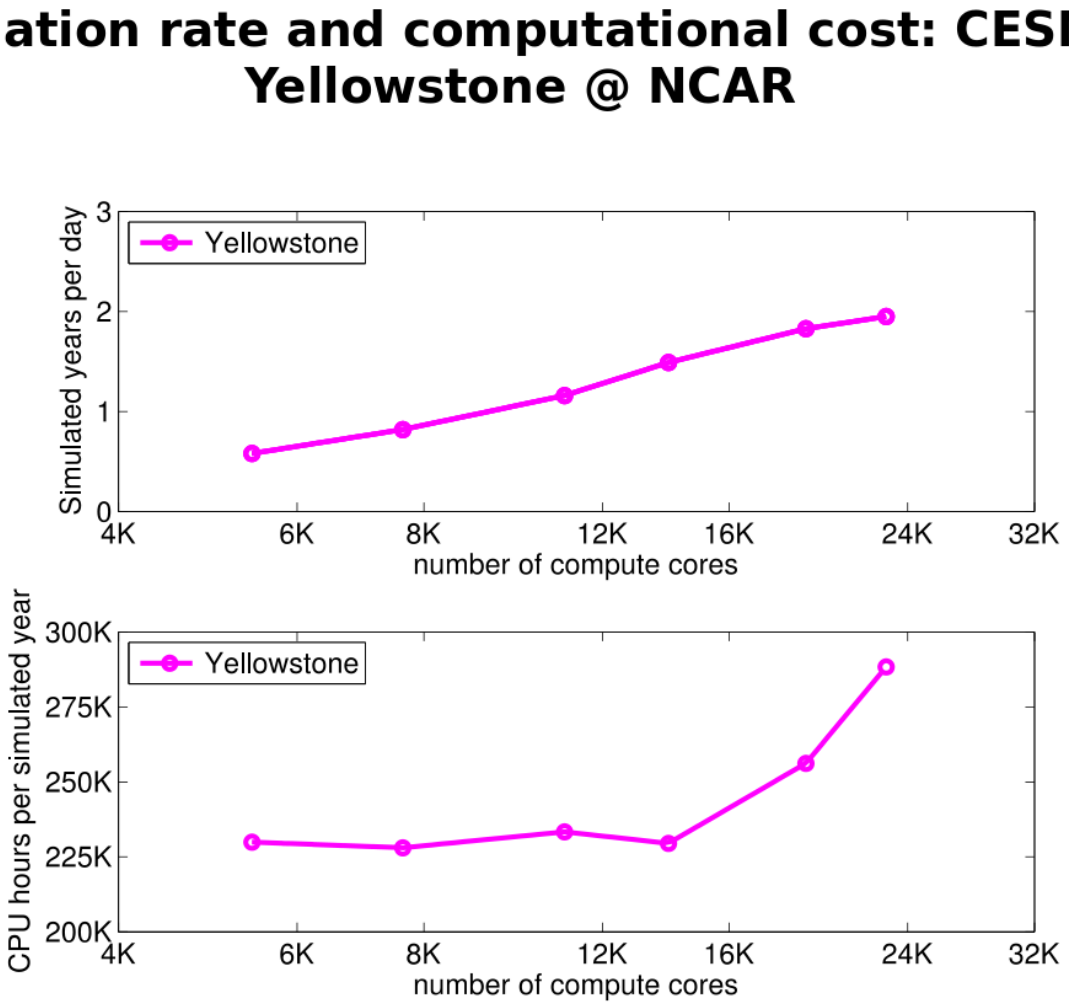
```
do k = kts,kte
!$acc loop gang vector
     do i = its,ite
       za(i,k) = 0.5*(zq(i,k)+zq(i,k+1))
```

CESM on Mira



Courtesy John Dennis and Rich Loft, NCAR. 0.25°atmosphere, 1°ocean on 32k cores of Mira at ∼2 SYPD.

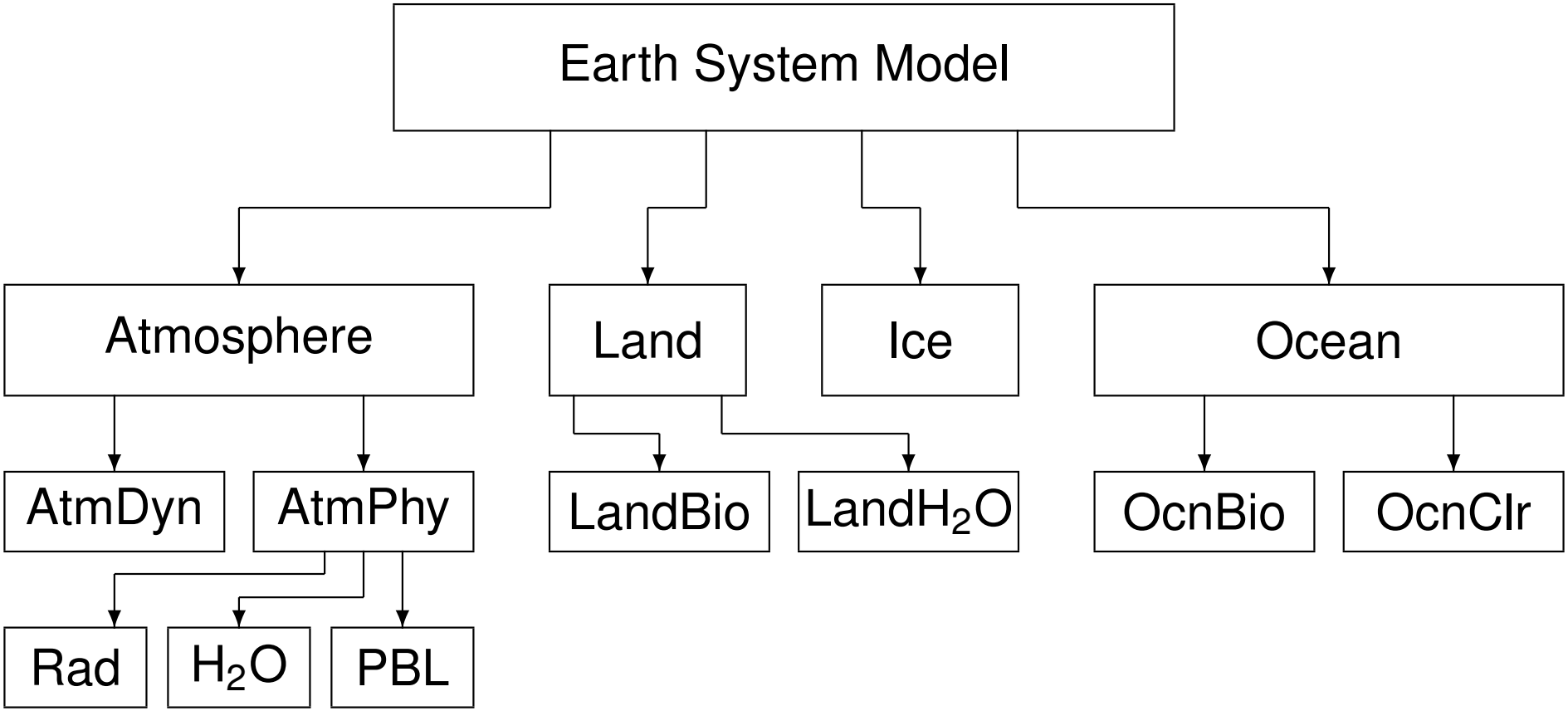
CESM on Yellowstone



Courtesy Rich Loft, NCAR. (See Rich Loft talk today for further details, Session 6 for discussion of real model performance.)

Outline

Earth System Model Architecture



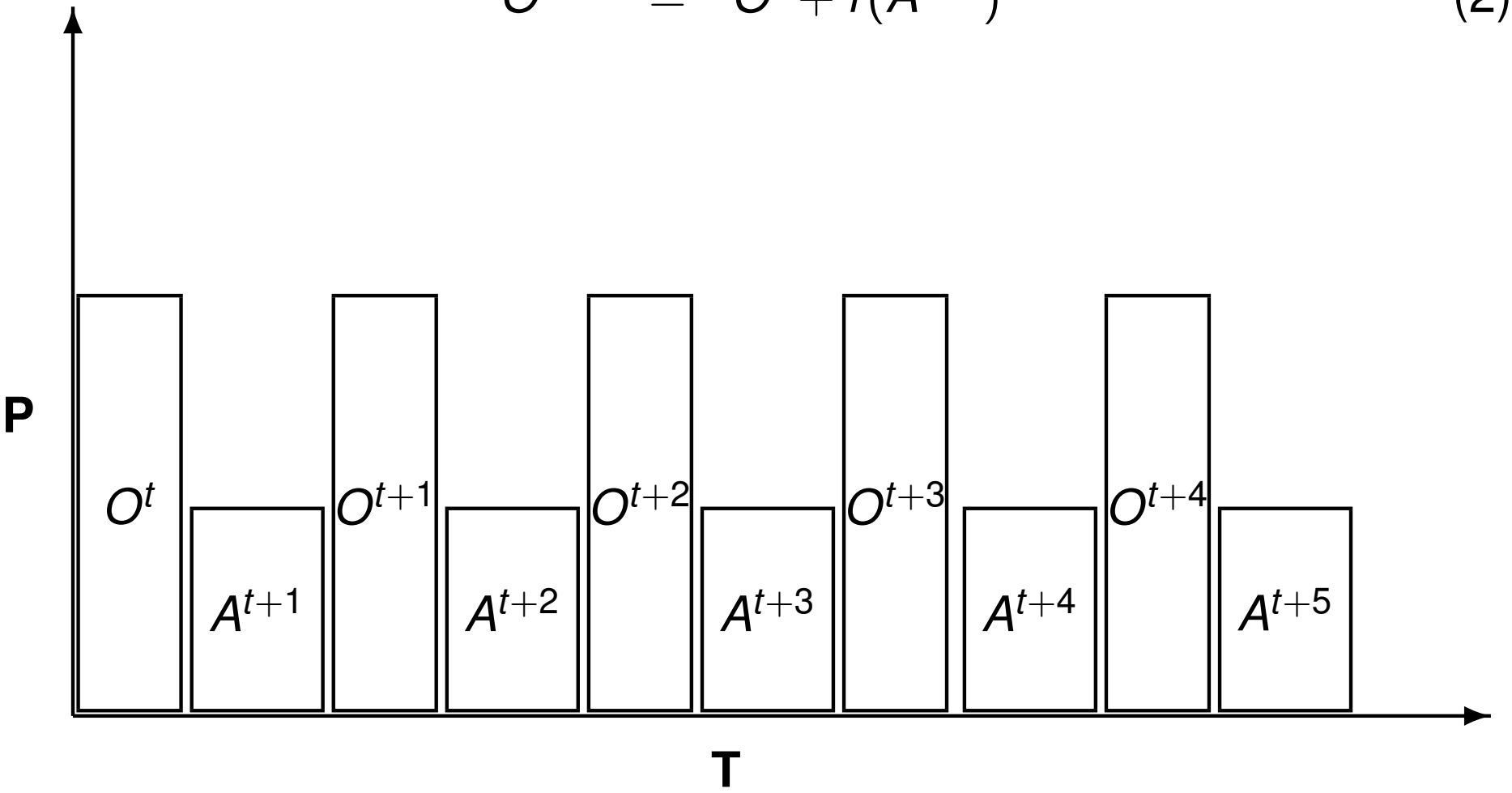
Extending component parallelism to $\mathcal{O}(10)$ requires a different physical architecture!

Serial coupling

Uses a forward-backward timestep for coupling.

$$A^{t+1} = A^t + f(O^t) \quad (1)$$

$$O^{t+1} = O^t + f(A^{t+1}) \quad (2)$$

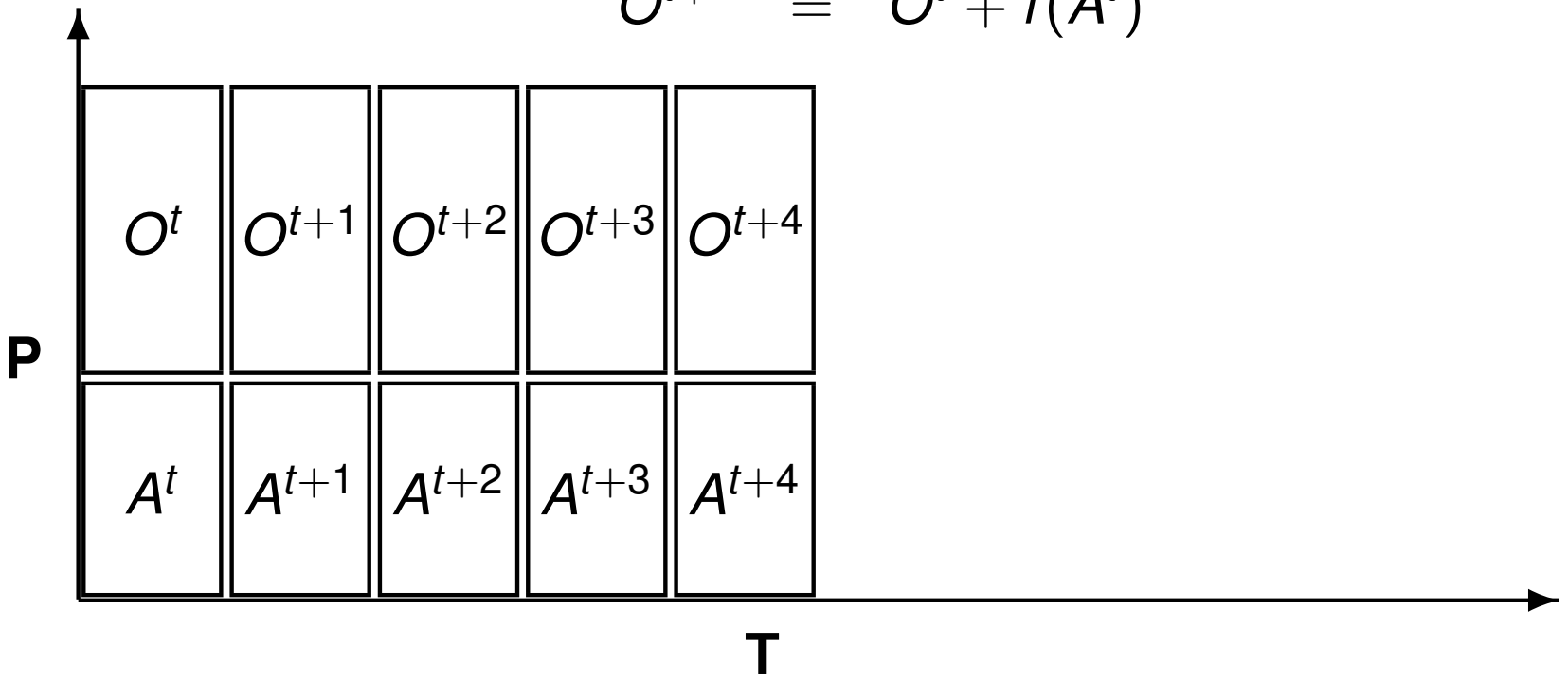


Concurrent coupling

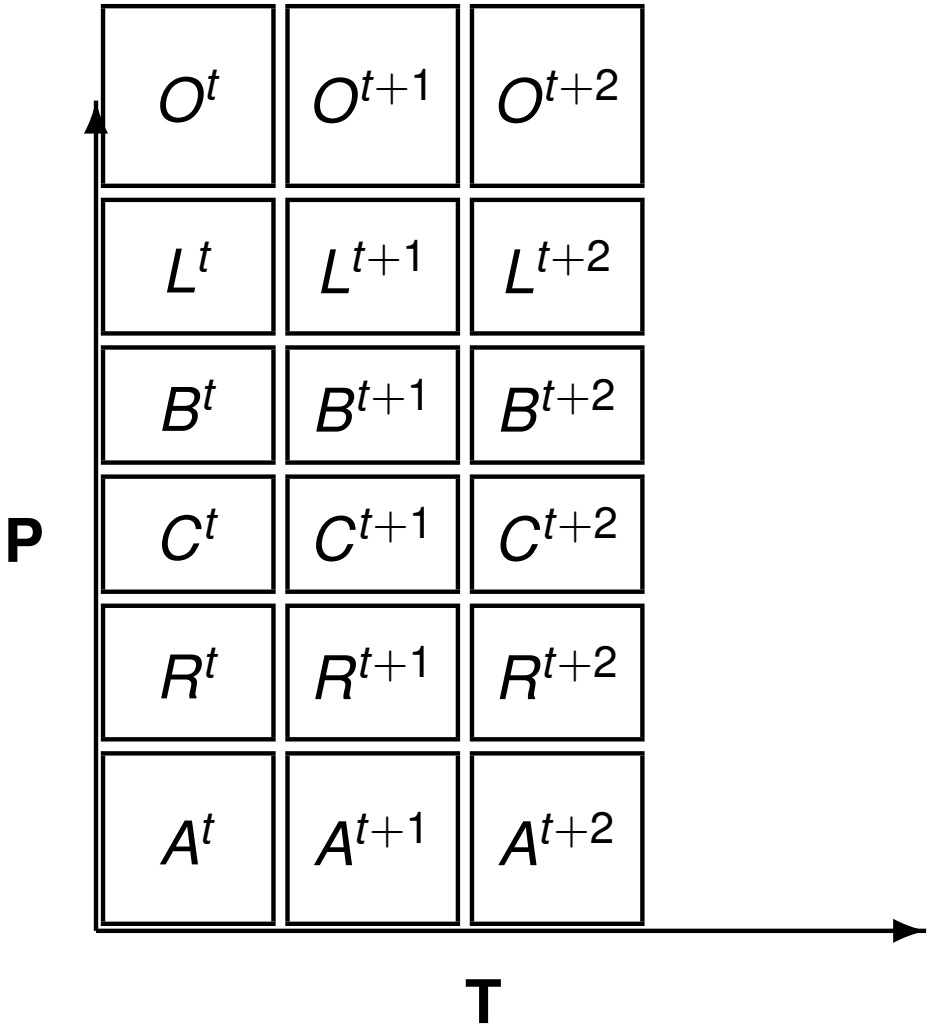
This uses a forward-only timestep for coupling. While formally this is unconditionally unstable, the system is strongly damped*. Answers vary with respect to serial coupling, as the ocean is now forced by atmospheric state from Δt ago.

$$A^{t+1} = A^t + f(O^t) \quad (3)$$

$$O^{t+1} = O^t + f(A^t) \quad (4)$$

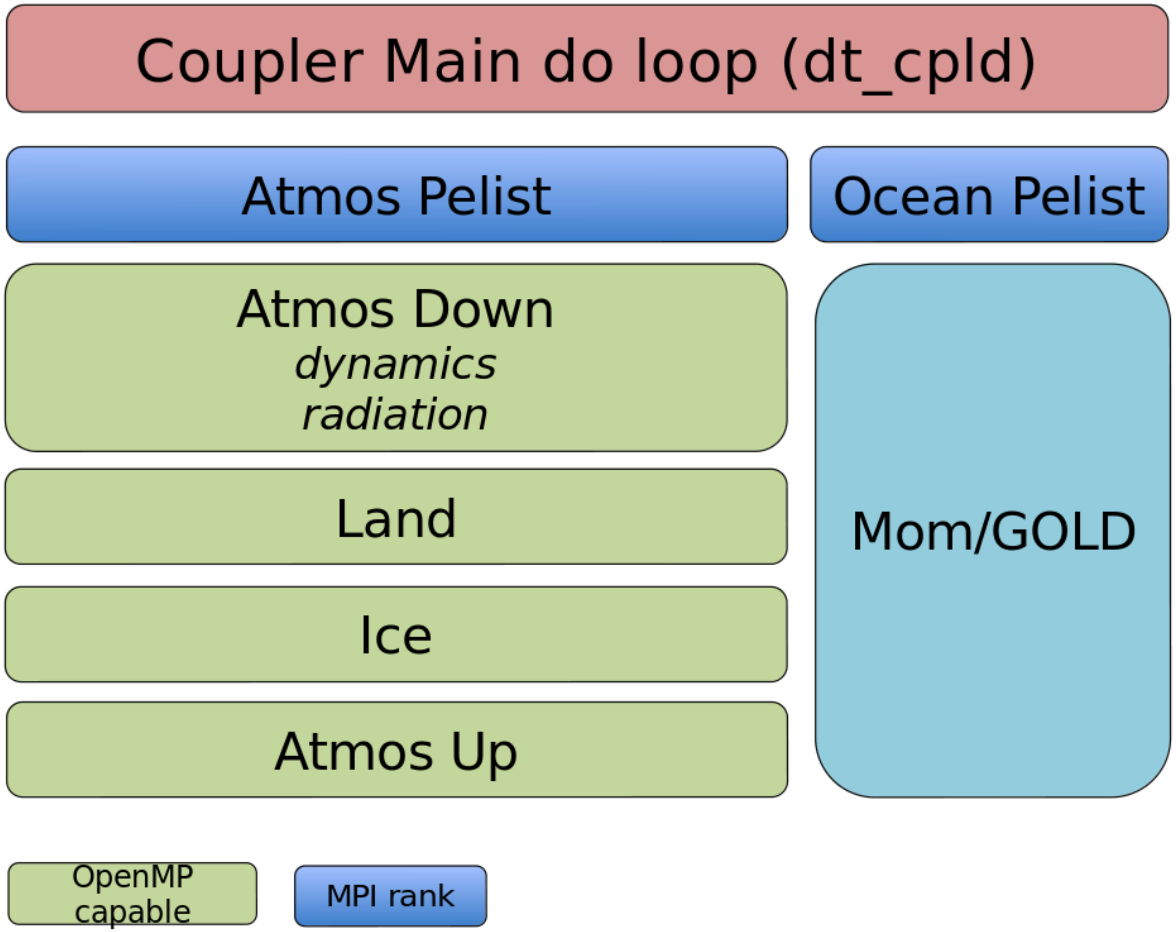


Massively concurrent coupling



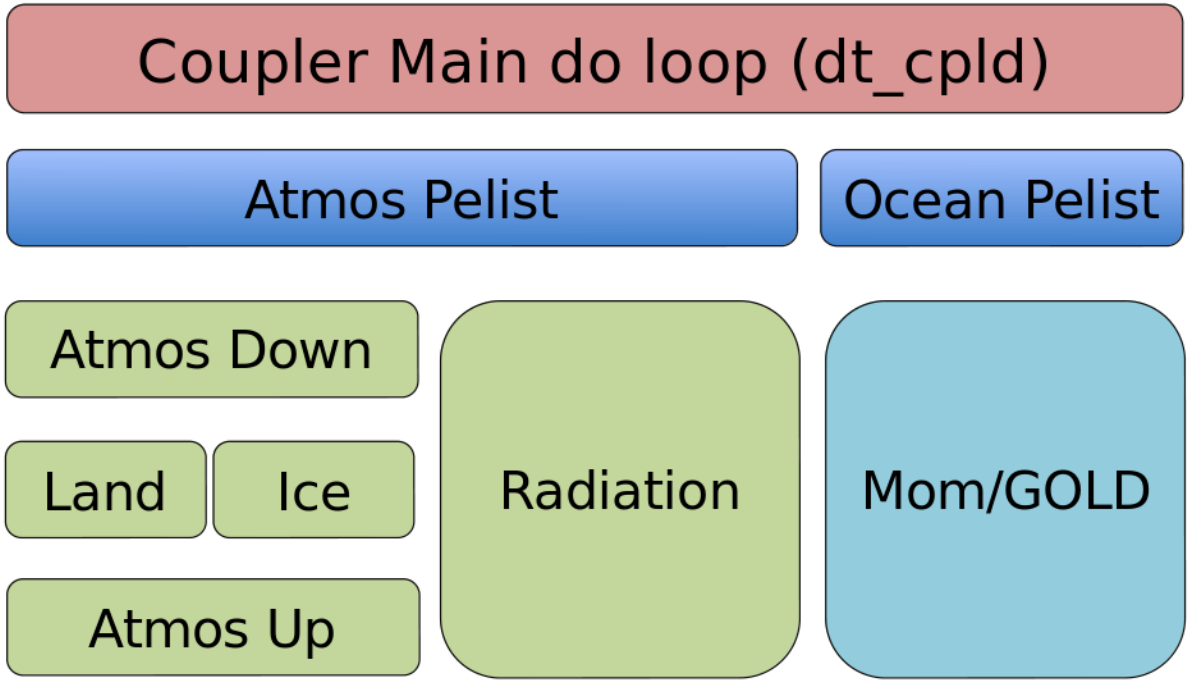
Components such as radiation, PBL, ocean biogeochemistry, each could run with its own grid, timestep, decomposition, even hardware. Coupler mediates state exchange.

Traditional coupling sequence



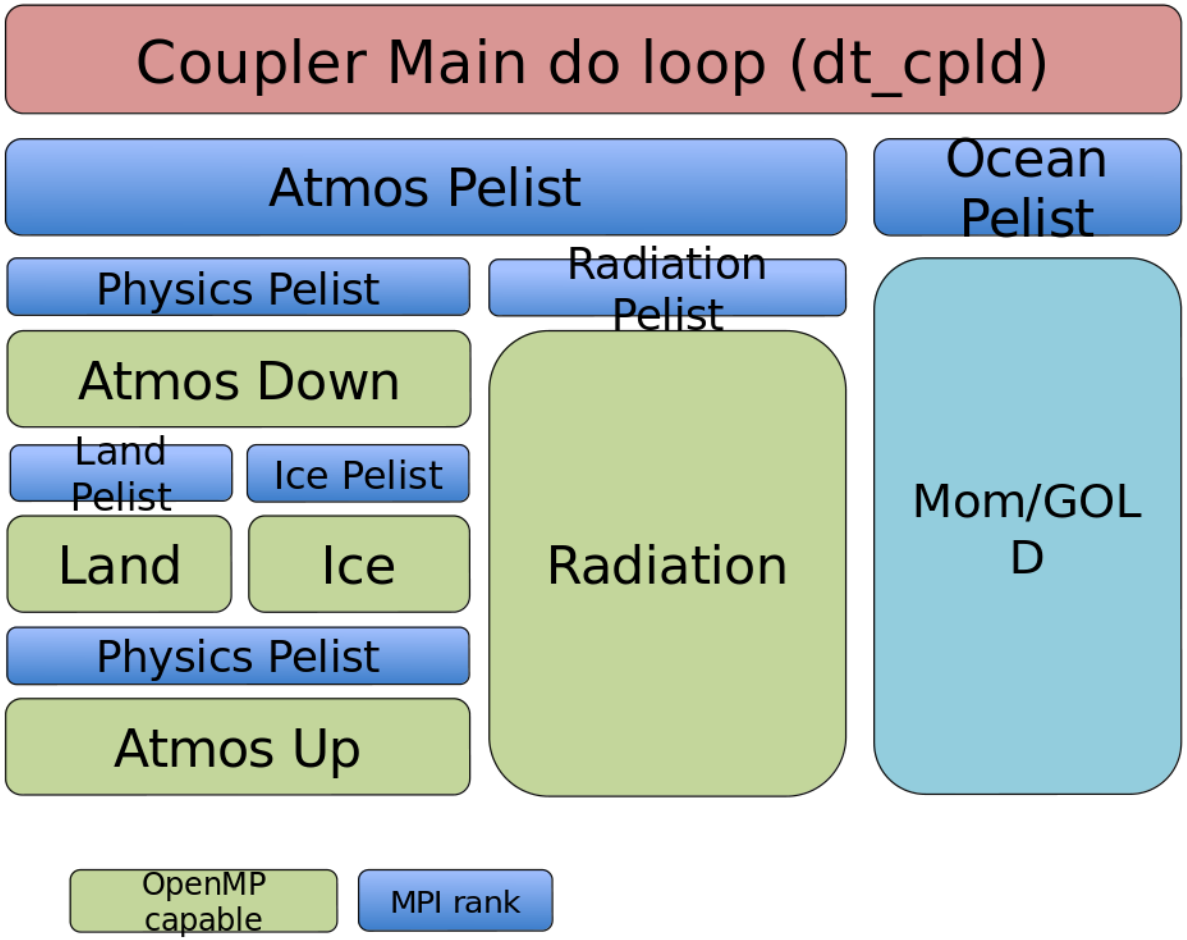
Radiation timestep much longer than physics timestep.
(Figure courtesy Rusty Benson, NOAA/GFDL).

Proposed coupling sequence



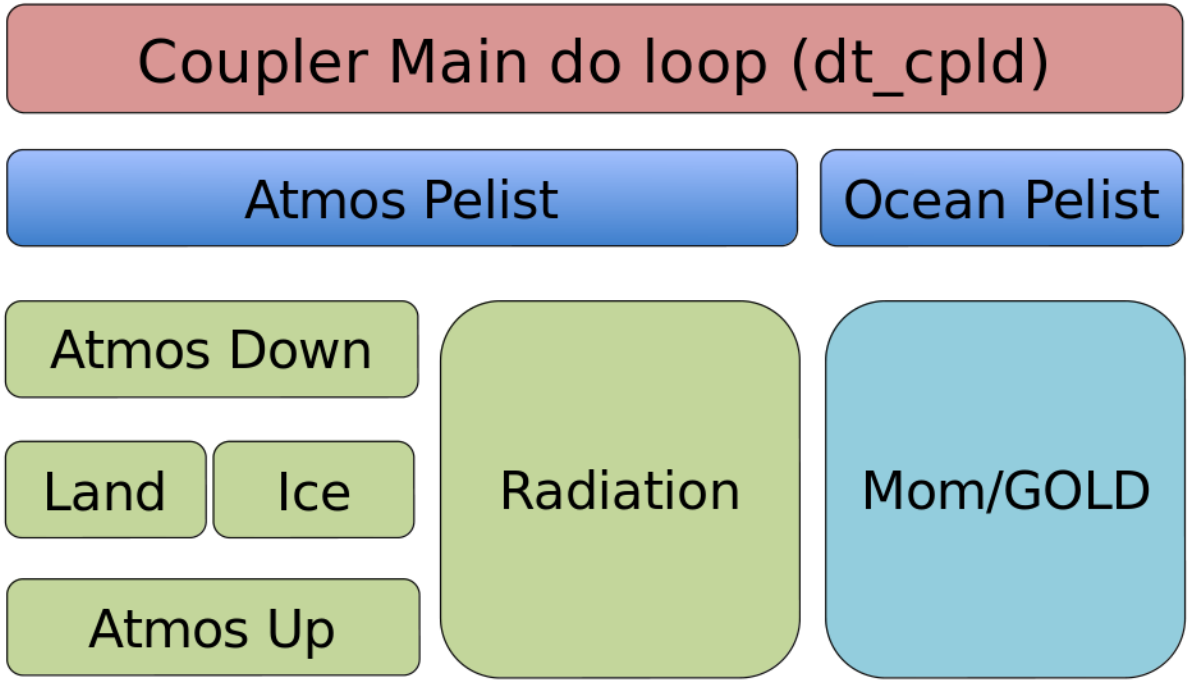
Radiation executes on physics timestep from lagged state.
(Figure courtesy Rusty Benson, NOAA/GFDL).

Proposed coupling sequence using pelists



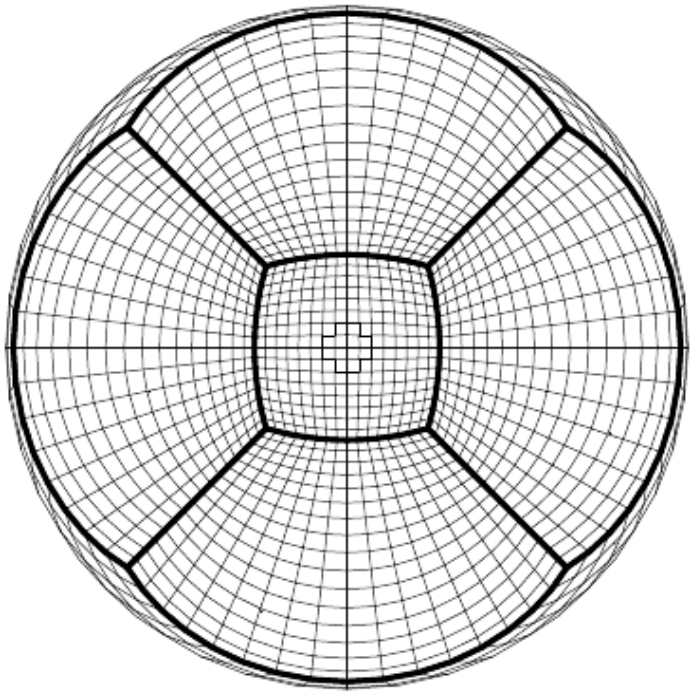
Requires MPI communication between physics and radiation.
(Figure courtesy Rusty Benson, NOAA/GFDL).

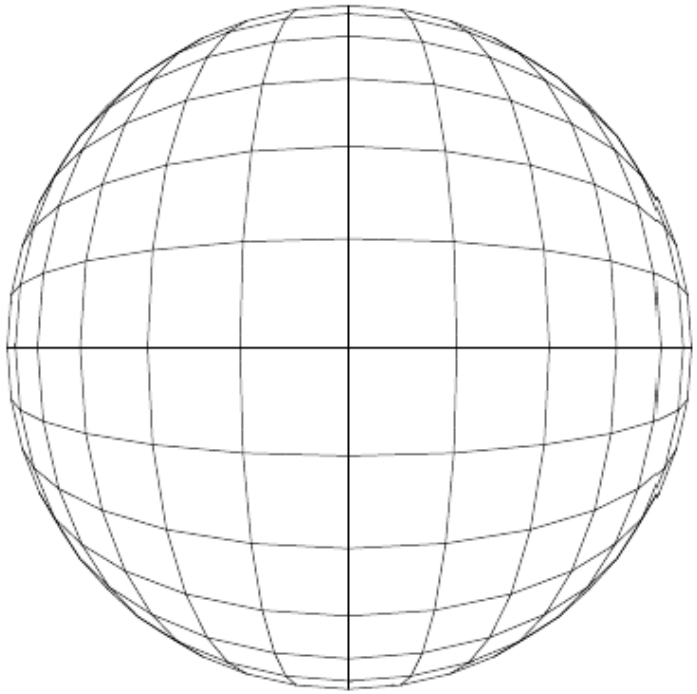
Proposed coupling sequence: hybrid approach



Physics and radiation share memory.
(Figure courtesy Rusty Benson, NOAA/GFDL).

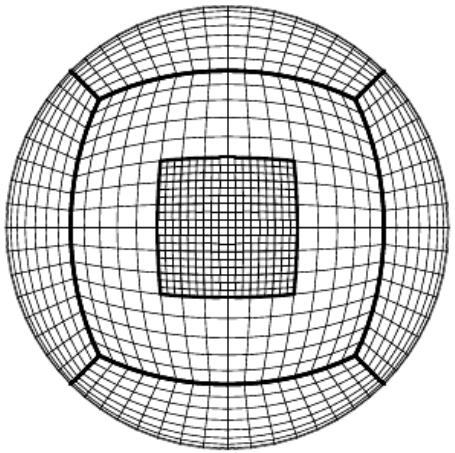
Stretched grids



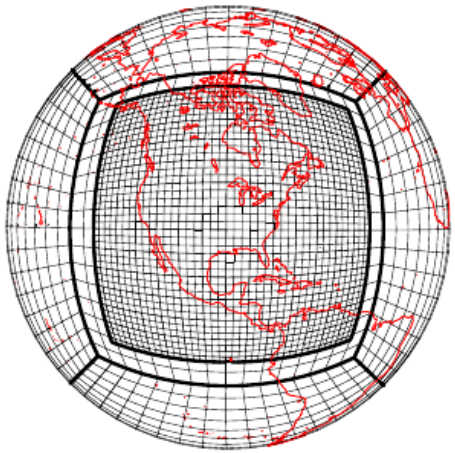


- Opposing face gets very coarse
- Discontinuities in slope
- Scale-aware parameterizations required

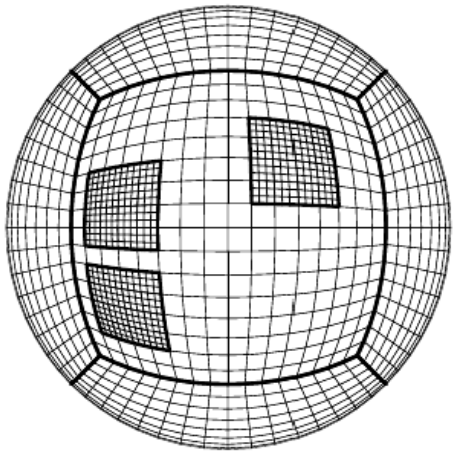
Nested grids



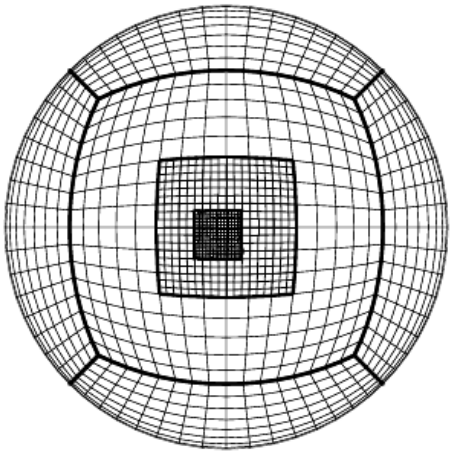
3:1 nested grid



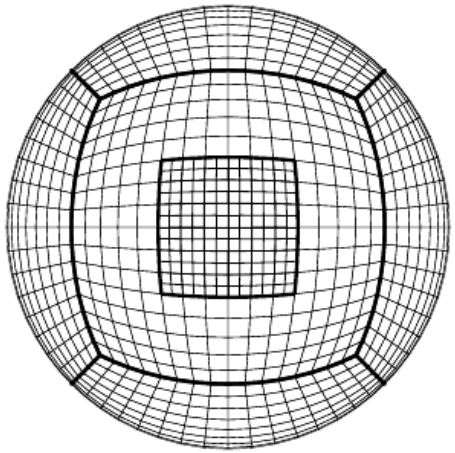
Large nest for RCMs



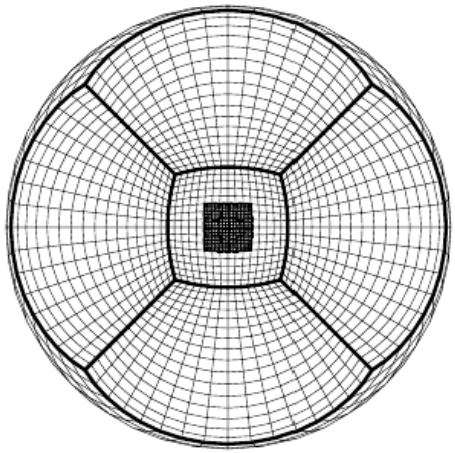
Multiple nests



Telescoping nests



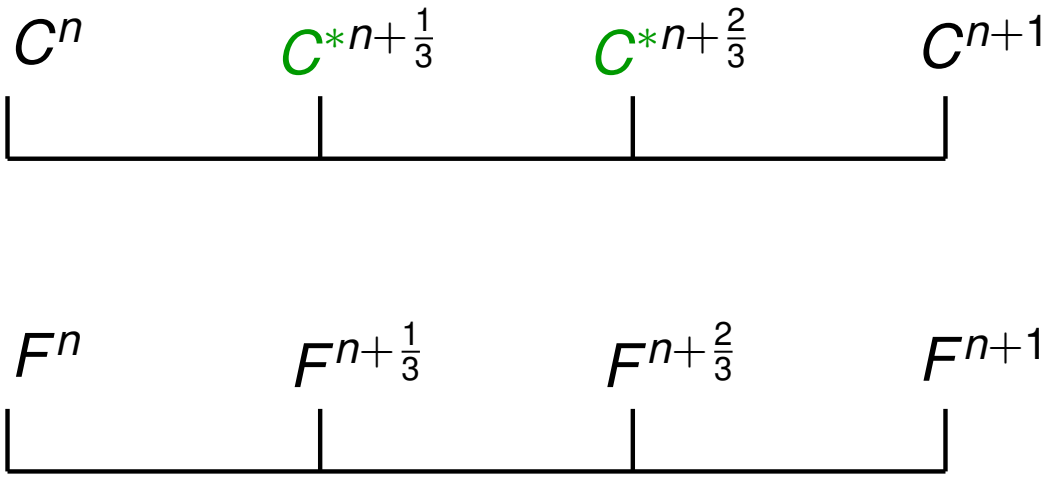
2:1 nested grid



Nest in stretched grid

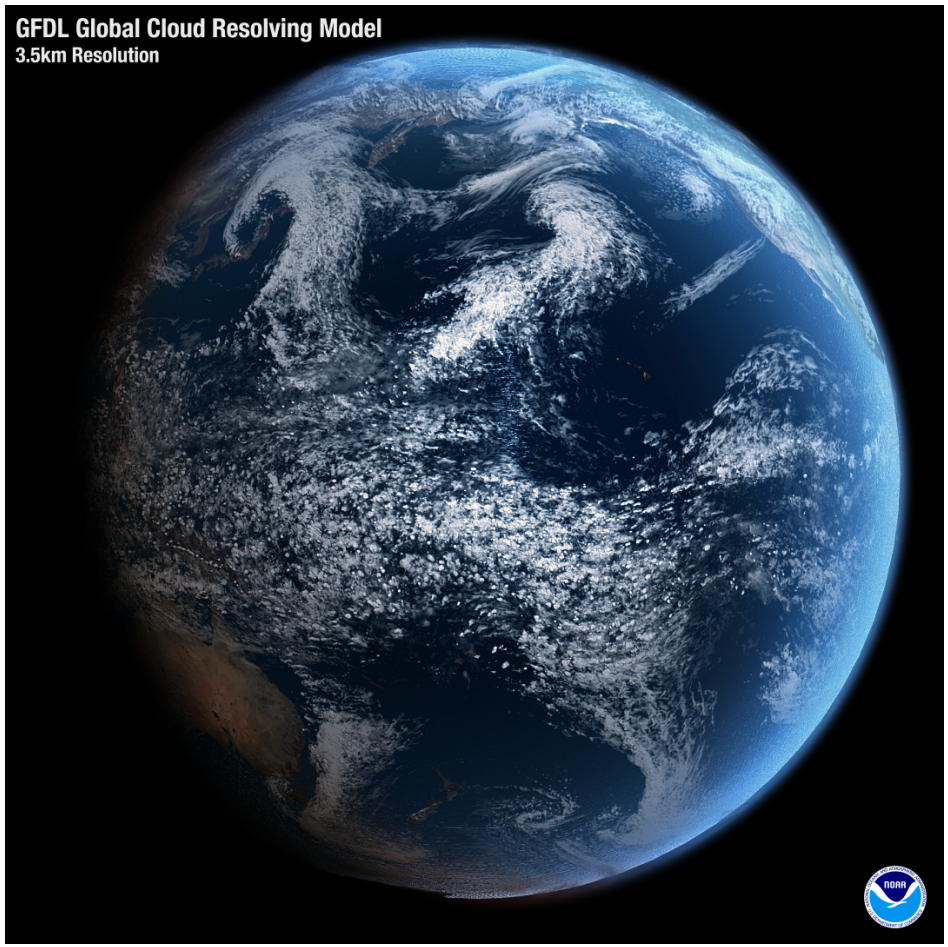
Concurrent two-way nesting

Typical nesting protocols force serialization between fine and coarse grid timestepping, since the C^* are estimated by interpolating between C^n and C^{n+1}.



We enable concurrency by instead estimating the C^* by extrapolation from C^{n-1} and C^n, with an overhead of less than 10%. (See Harris and Lin 2012 for details.)

C2560: 3.5 km resolution global cloud-resolving model



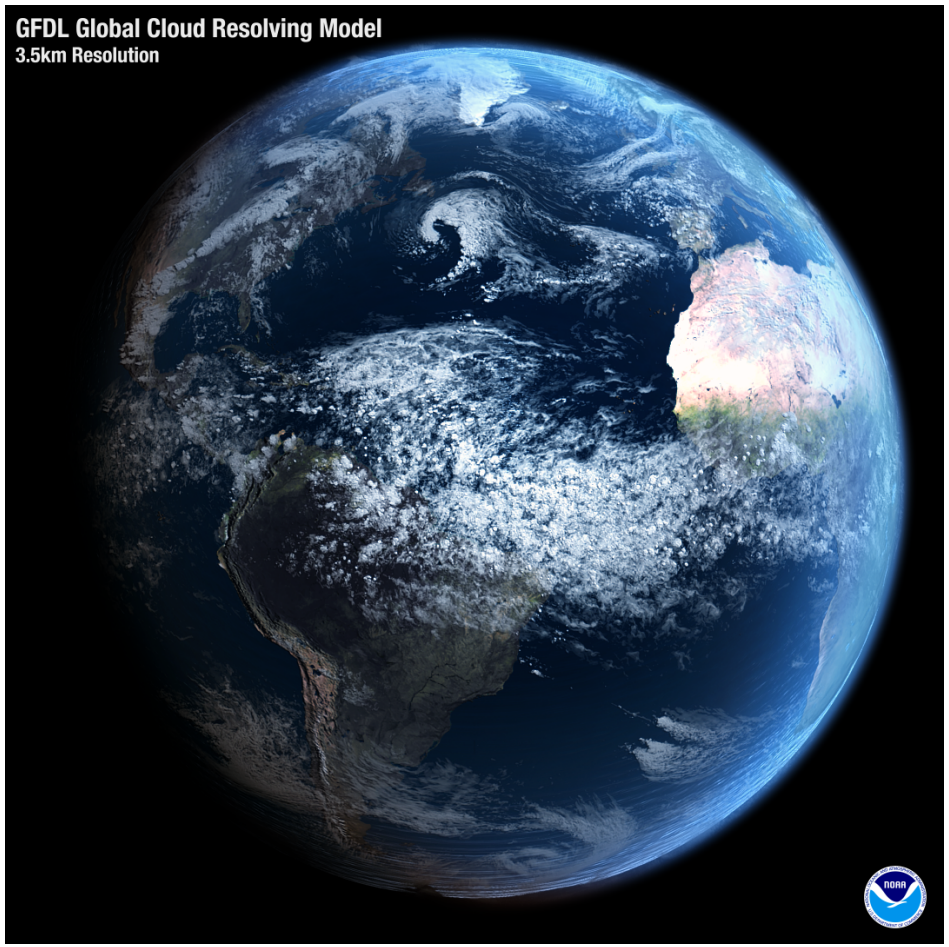


Figure courtesy S-J Lin and Chris Kerr, NOAA/GFDL.

Acknowledgements ANL Early Science Program run on BG/Q.

Outline

Challenges for the next generation of models

(... and for the ENES Exascale Workshop).

- Models are more than “simulators”: it must be possible to infer the physical basis of emergent behavior.
- What is “co-design”? Probably little or no influence on hardware, but can we have co-design between applications and compilers?
- Can we have high-level programming models or frameworks to take advantages of new approaches to parallelism? What are the right abstractions?
- Can component-level parallelism via framework superstructure be pushed to $\mathcal{O}(10)$?
- Do we need to approach models as experimental biological systems? (single organism or “cell line” not exactly reproducible; only the ensemble is.)
- How do we analyze and understand performance on a “sea of functional units” (Kathy Yelick’s phrase)?

Acknowledgments: Rich Loft, Mark Govett, Kareem Sorathia