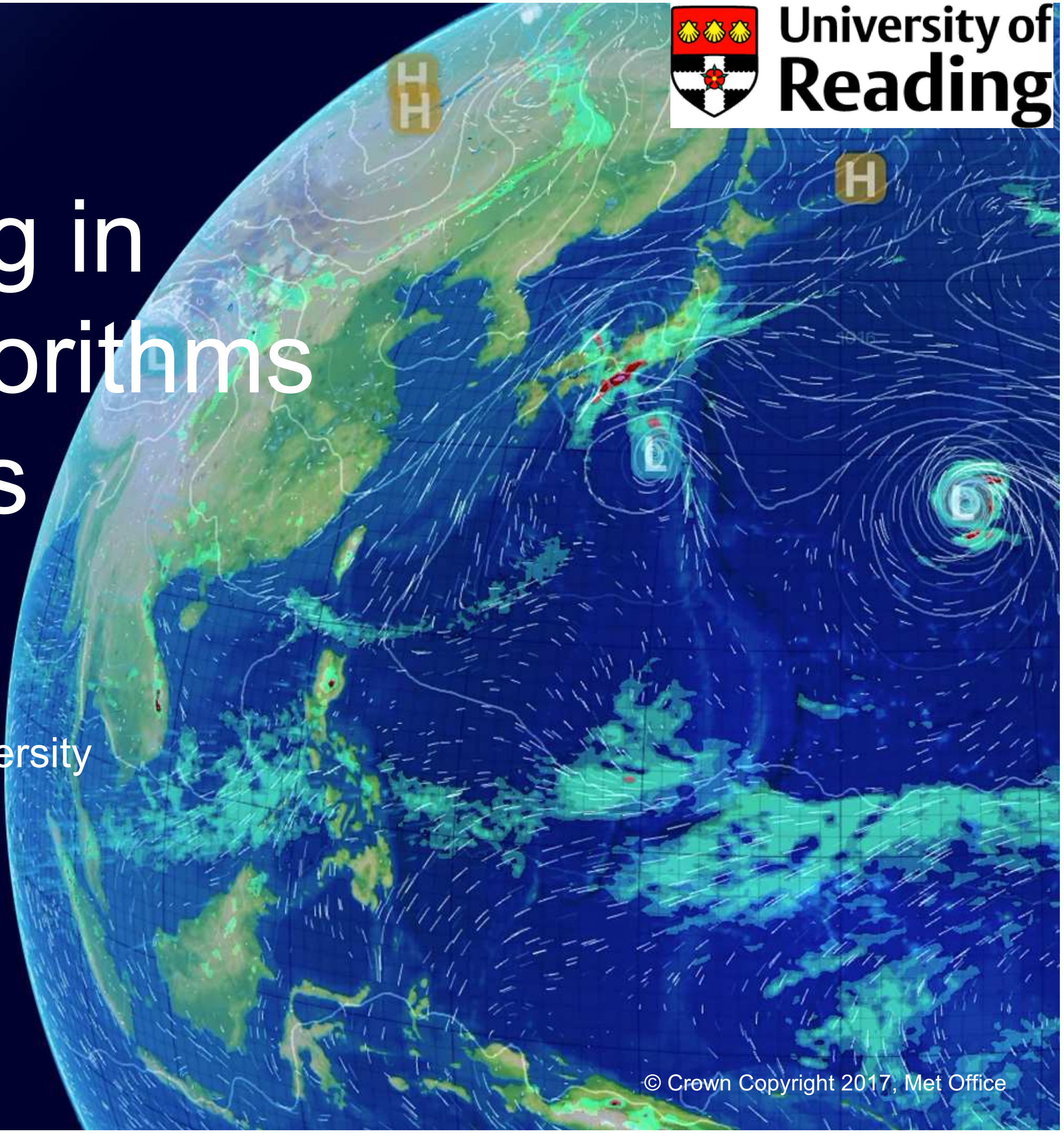
Parallel programming in practice: Scaling algorithms and Coupling models

Dr Chris Maynard

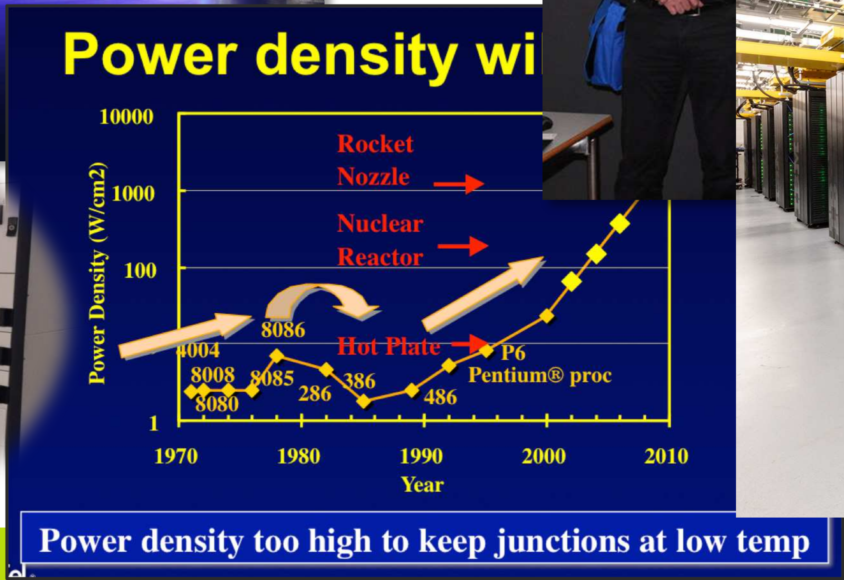
Acting HPC Optimisation Manager - Met Office

Associate Professor of Computer Science – University of Reading

www.aces.cs.reading.uk



End of the Free Lunch



Computation

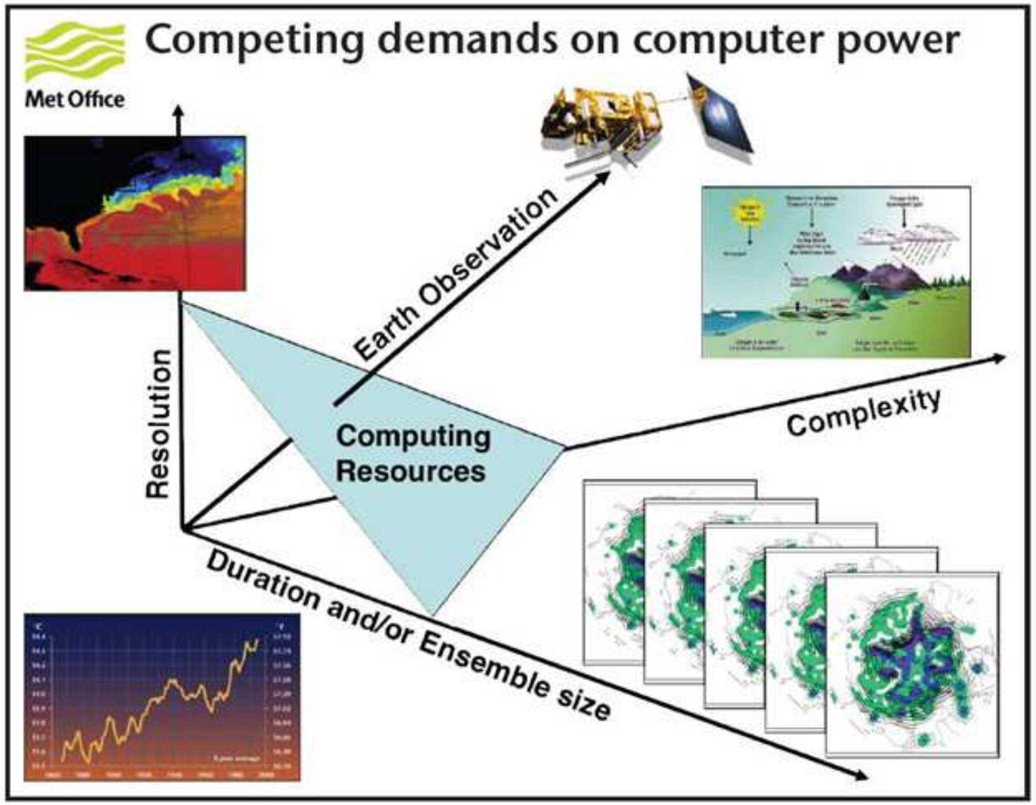
Scientific innovation is limited by computation
- Size and speed of calculation

Speed of computation – End of Moore's Law

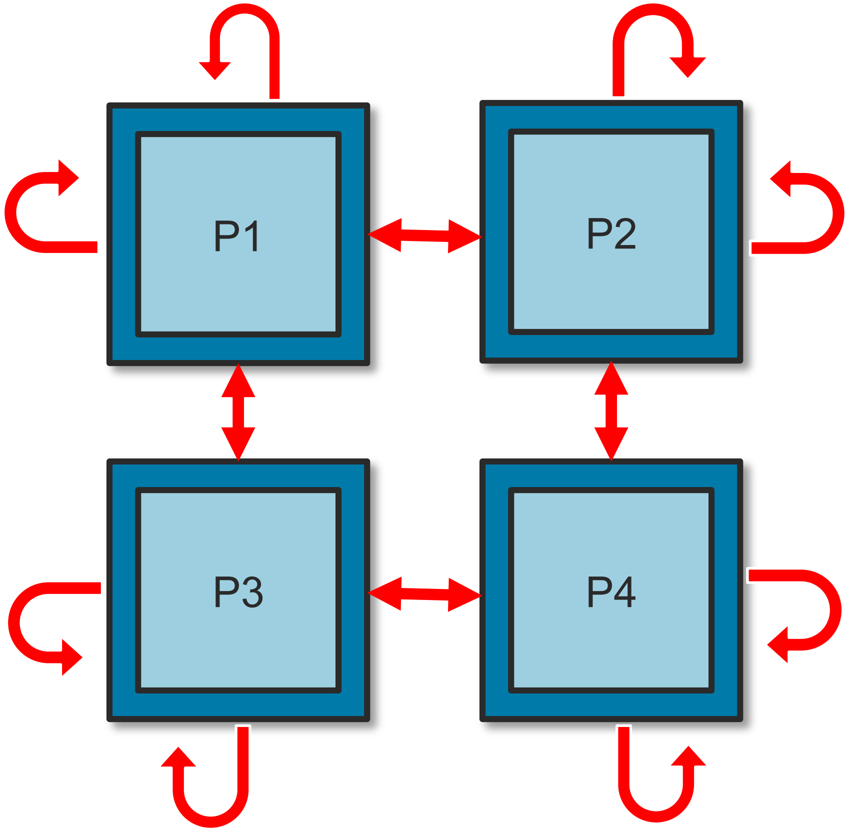
If we cannot compute X and Y faster

Can we compute X and Y simultaneously?

Dependency Y depends on X – cannot compute simultaneously



Parallelism



Data Parallelism

Data decomposed across parallel elements (PEs)

PEs perform same action on different data

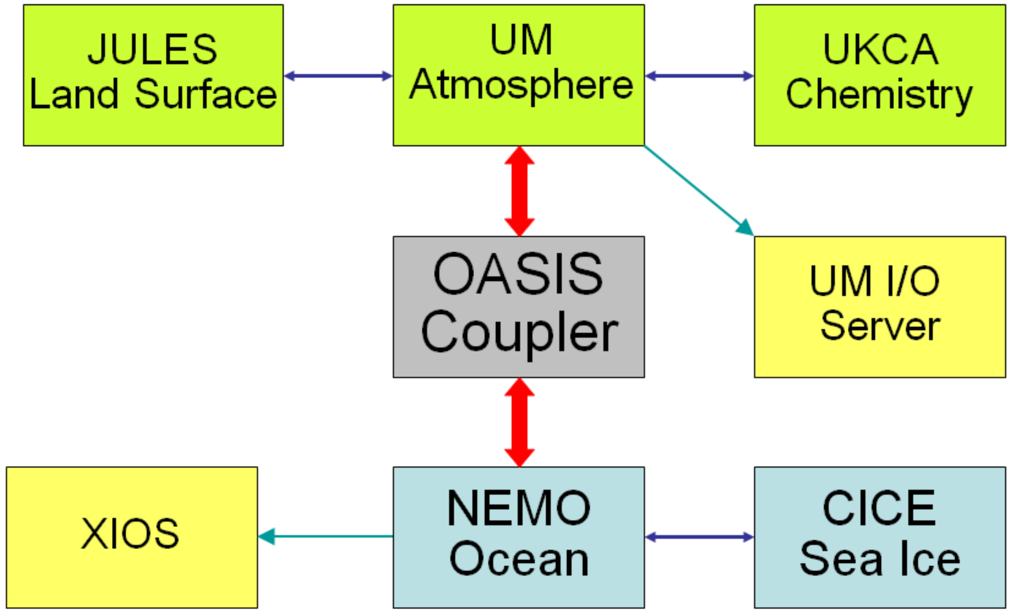
Single Program Multiple Data (SPMD) over MPI

- this example includes data movement – halo exchange

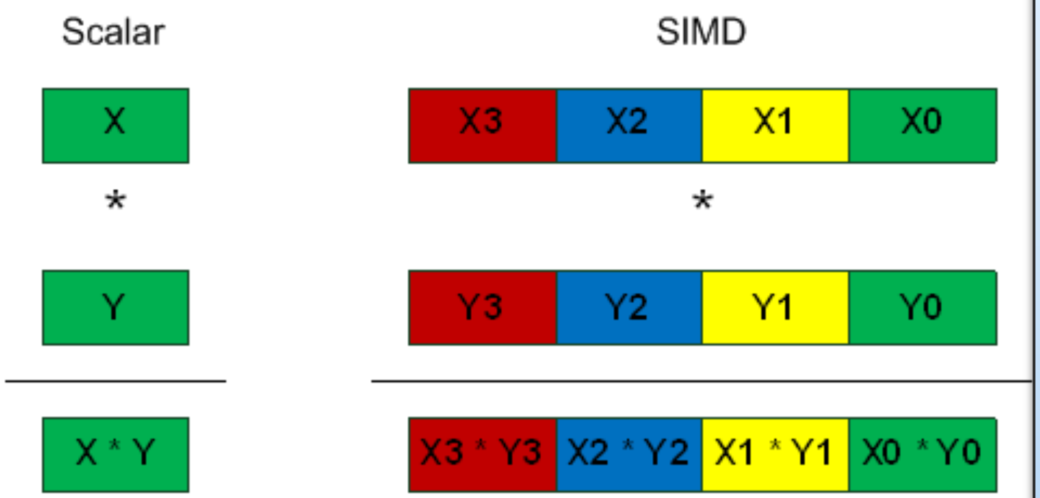
Parallelism II

Task parallelism

Also known as functional parallelism
Decompose problem into independent pieces
Coupled models (ATM + Ocean)
Ensembles (many models on perturbed data)
I/O server – Asynchronous offload
CPU + GPU -- kernel offload



Parallelism III



Instruction Level Parallelism (ILP)

Fused multiply add (single op)

```
F = a*x + y
```

Single Instruction Multiple Data (SIMD) – combined with data parallelism

Vectorisation on modern CPU

Distinct from *pipeline vectorisation* on true vector processors

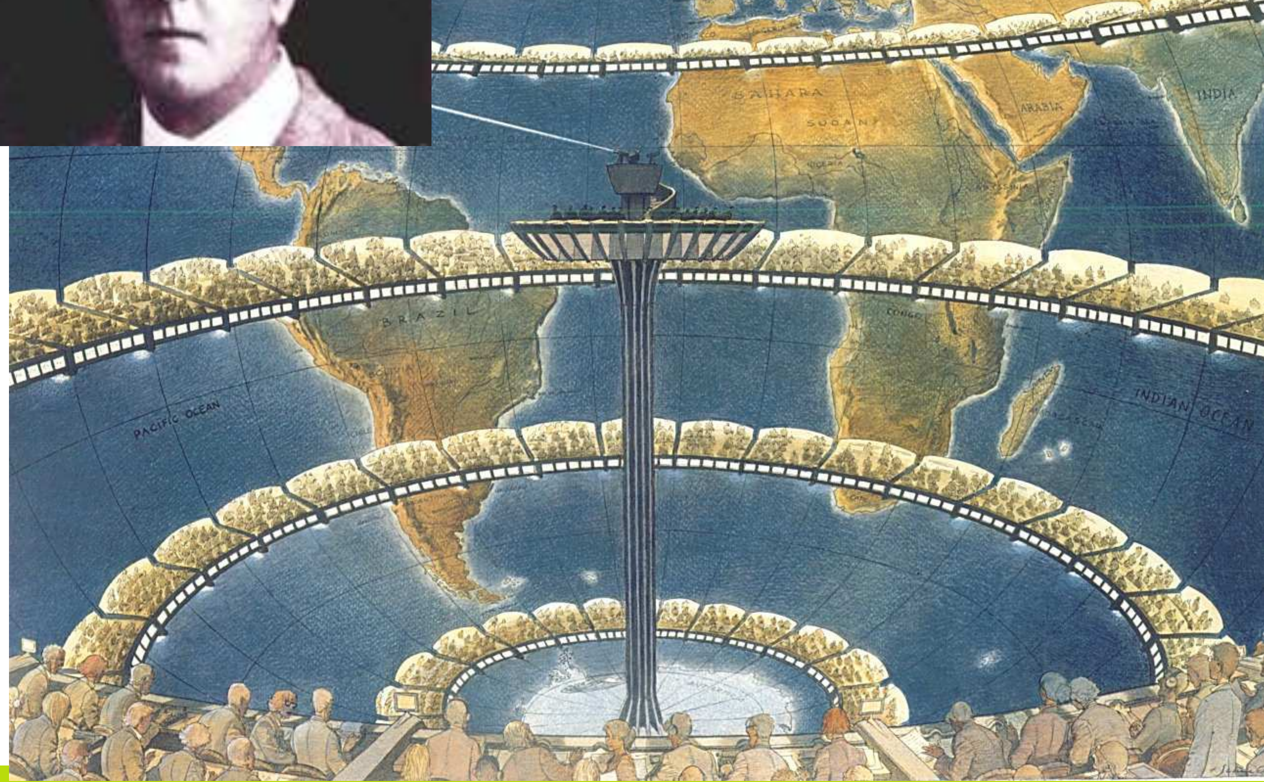
c.f. coalesced memory access on GPU Single Instruction Multiple Thread (SIMT)

Met Office

Lewis Fry Richardson

University of Reading

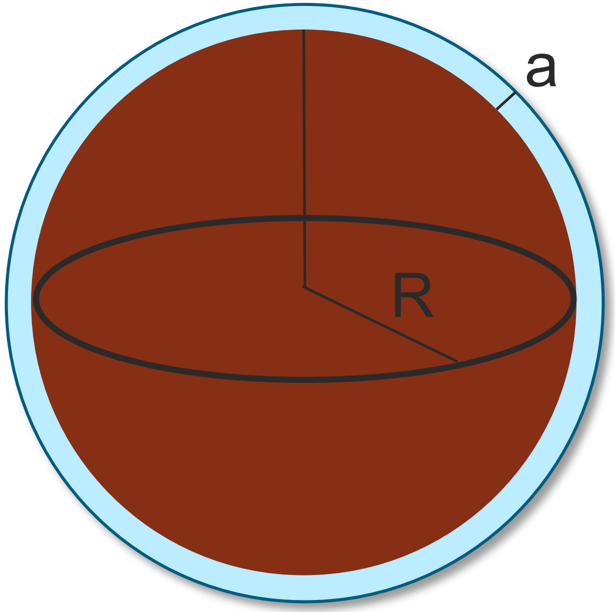
LFR attempted first NWP calculation 1916-191. Volunteer ambulance unit on the Western front.
7x7x5 grid, (250km resolution over Europe) two 3-hour timesteps.
Completely wrong – bad input data and CFL condition violation (not known).
1922 paper *Weather Prediction by Numerical Process*
Recognised problem was parallel (64,000 computers)



Building a model

Designing a dynamical core

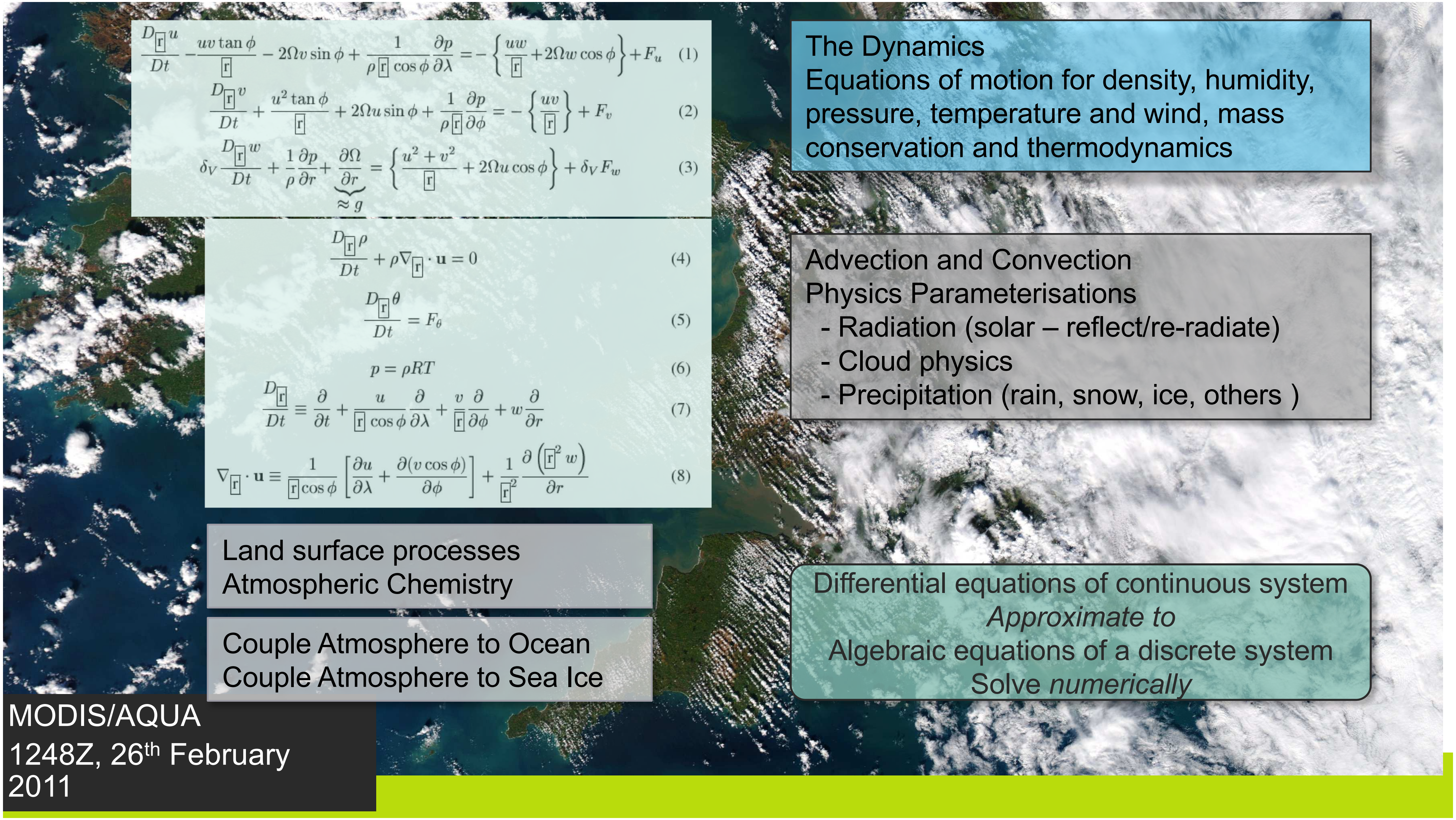
The domain specifics



Geometry: Spherical, Orography (Mountains)
Atmosphere is thin and vertically stratified
Kármán line ~ 100Km (99.99997% atm)
Diagram drawn to scale ~ 600Km atm. $R \gg a$
Rotation, not in thermal equilibrium
Atmosphere, Ocean, (Sea) Ice, land surface
Moist, Chemical and Biological processes

Very complex domain
multi-component
multi-scale

Models:
Large, $O(10^5)$ – $O(10^6)$ LoC
Legacy, 10+ years to develop, lifetime 25+ years
Continuous development (e.g. UM 20% PA)
Operations (and production) – Conservative, scientifically prudent!



$$\frac{D_{\boxed{r}} u}{Dt} - \frac{uv\tan\phi}{\boxed{r}} - 2\Omega v\sin\phi + \frac{1}{\rho\boxed{r}\cos\phi}\frac{\partial p}{\partial\lambda} = -\left\{\frac{uw}{\boxed{r}} + 2\Omega w\cos\phi\right\} + F_u \quad (1)$$

$$\frac{D_{\boxed{r}} v}{Dt} + \frac{u^2\tan\phi}{\boxed{r}} + 2\Omega u\sin\phi + \frac{1}{\rho\boxed{r}}\frac{\partial p}{\partial\phi} = -\left\{\frac{uv}{\boxed{r}}\right\} + F_v \quad (2)$$

$$\delta_V\frac{D_{\boxed{r}} w}{Dt} + \frac{1}{\rho}\frac{\partial p}{\partial r} + \underbrace{\frac{\partial\Omega}{\partial r}}_{\approx g} = \left\{\frac{u^2+v^2}{\boxed{r}} + 2\Omega u\cos\phi\right\} + \delta_V F_w \quad (3)$$

$$\frac{D_{\boxed{r}}\rho}{Dt} + \rho\nabla_{\boxed{r}}\cdot\mathbf{u} = 0 \quad (4)$$

$$\frac{D_{\boxed{r}}\theta}{Dt} = F_\theta \quad (5)$$

$$p = \rho RT \quad (6)$$

$$\frac{D_{\boxed{r}}}{Dt} \equiv \frac{\partial}{\partial t} + \frac{u}{\boxed{r}\cos\phi}\frac{\partial}{\partial\lambda} + \frac{v}{\boxed{r}}\frac{\partial}{\partial\phi} + w\frac{\partial}{\partial r} \quad (7)$$

$$\nabla_{\boxed{r}}\cdot\mathbf{u} \equiv \frac{1}{\boxed{r}\cos\phi}\left[\frac{\partial u}{\partial\lambda} + \frac{\partial(v\cos\phi)}{\partial\phi}\right] + \frac{1}{\boxed{r}^2}\frac{\partial\left(\boxed{r}^2 w\right)}{\partial r} \quad (8)$$

The Dynamics
Equations of motion for density, humidity, pressure, temperature and wind, mass conservation and thermodynamics

Advection and Convection
Physics Parameterisations
- Radiation (solar – reflect/re-radiate)
- Cloud physics
- Precipitation (rain, snow, ice, others)

Land surface processes
Atmospheric Chemistry

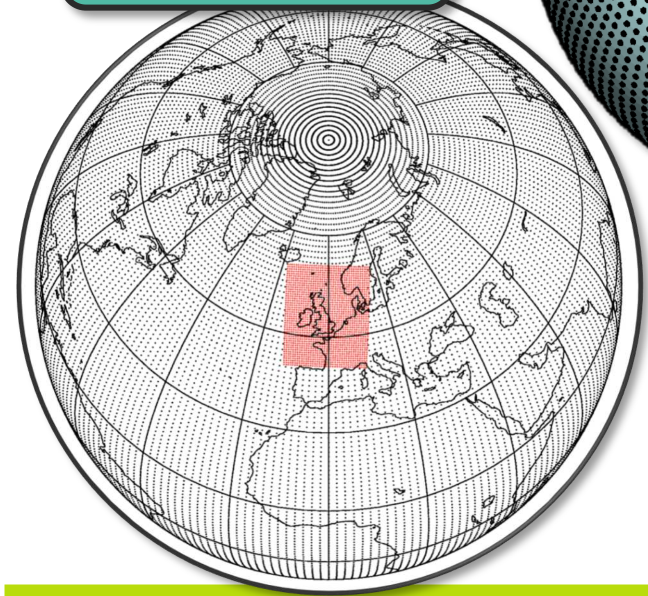
Couple Atmosphere to Ocean
Couple Atmosphere to Sea Ice

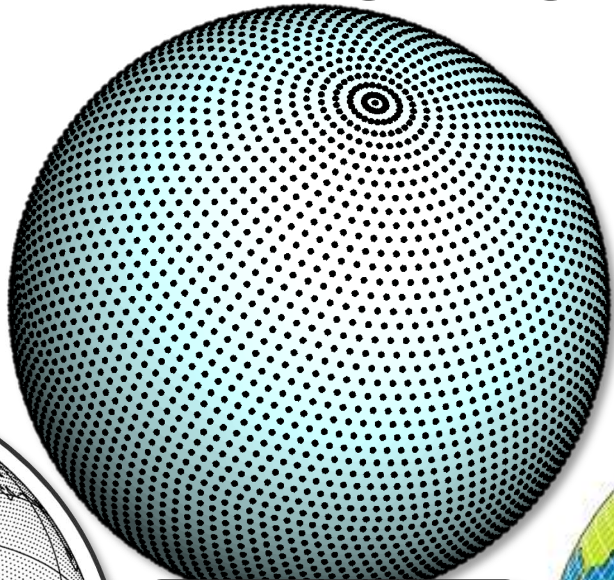
Differential equations of continuous system
Approximate to
Algebraic equations of a discrete system
Solve *numerically*

MODIS/AQUA
1248Z, 26th February 2011

Choosing a grid

Lat-Lon grid
Structured
Unified Model
(Met Office)





Octahedral
Gaussian grid
Structured
IFS (ECMWF)

Cubed Sphere
GungHo/LFRic
unstructured
(MO)
FV3 structured
(NOAA)

Icosahedral mesh
Unstructured
ICON (DWD)
MPAS Voronoi
stretch meshes

Choice of grid

Choice of grid based on Numerical Analysis
Symmetry properties
Consequences for

1. Accuracy
2. Stability

of numerical method

Structured

Neighbouring grid-points/cells known
Direct memory access

```
u(i) = u(i-1) + u(i+1)
```

Good for data locality and caching
Geometry of sphere → problematic communication patterns

Unstructured

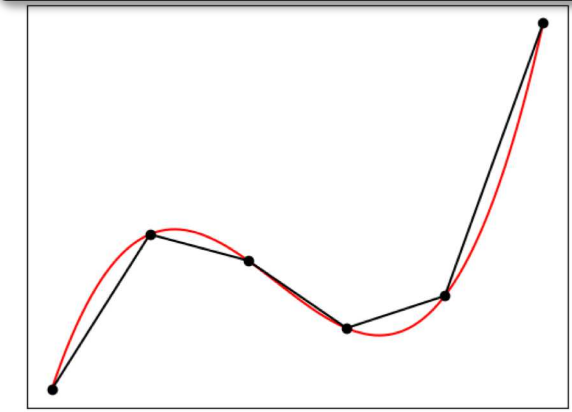
Neighbour grid-points/cells not known. Use look up table → indirect memory access

```
u(m(cell)) = u(st(cell,1)) + u(st(cell,2))
```

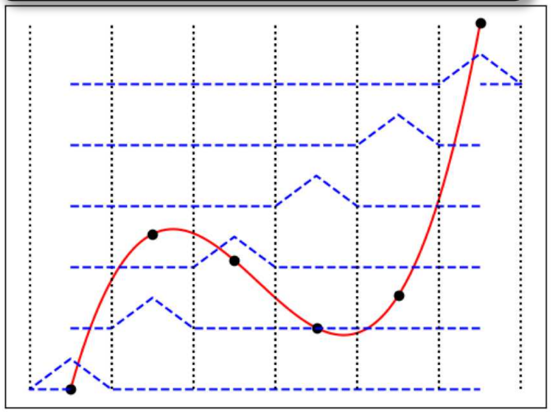
Bad for data locality, can avoid problematic communication patterns

Discretisation method

i) 1st order forward difference



iii) Linear Finite Element



i) Finite Difference - values a grid-points

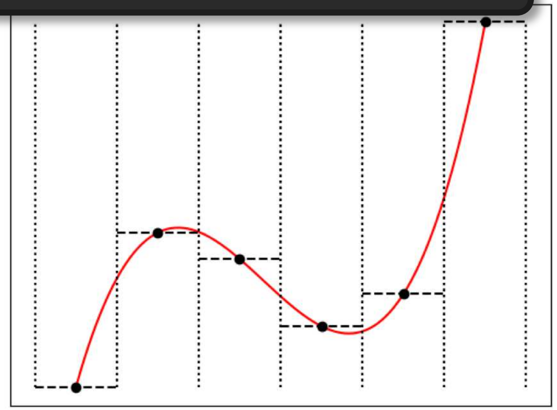
ii) Finite Volume - values over volume

iii) Finite Element - functions over cell

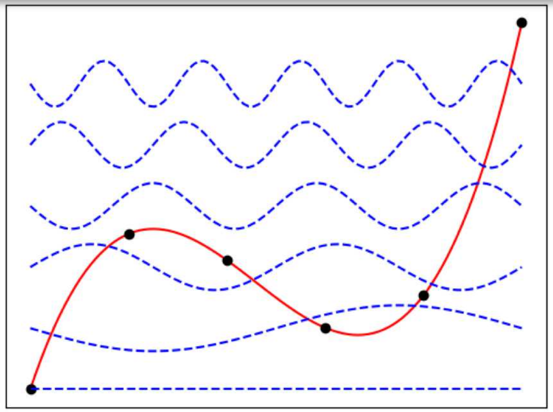
iv) Spectral - periodic (trigonometric/hyperbolic) functions over whole domain

All can be at higher order (lowest shown)

i) Constant Finite Volume



iv) Spectral method



Time-stepping and CFL

Also need to discretise time
– again choices of “grid” and
time-stepping scheme.
Courant-Friedrichs-Lewy (CFL)
condition (stability)

$$\frac{u\Delta t}{\Delta x} \leq C$$

1-d advection where u is wave-velocity
C depends on u and discretisation scheme $C \sim \mathcal{O}(1)$

Different atmospheric waves
Acoustic, Gravity, Rossby
different wavelengths and can have different treatments
i) Explicit $u(t_n) \sim f(u(t_{n-1}))$
ii) Implicit $u(t_n) \sim g\left(u(t_{n-1}, u(t_n)\right)$
Also advection, i) Eulerian versus ii) Semi-Lagrangian

i) Cheap to compute, small time-step
ii) Costly to compute, large time-step

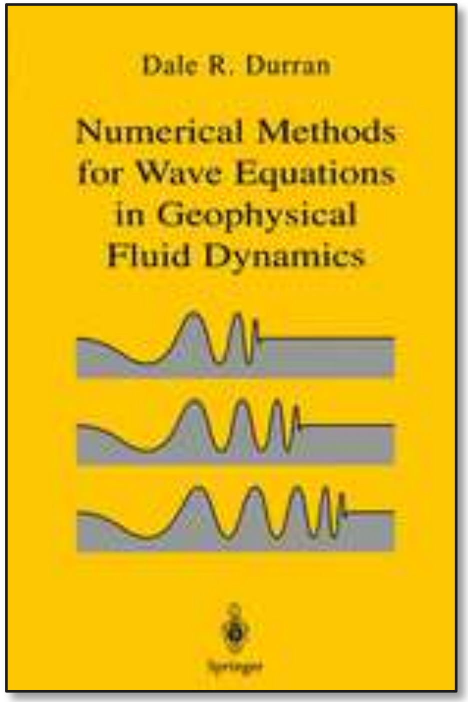
Dynamics summary

This is not a course on the dynamics nor numerical analysis

Choices of

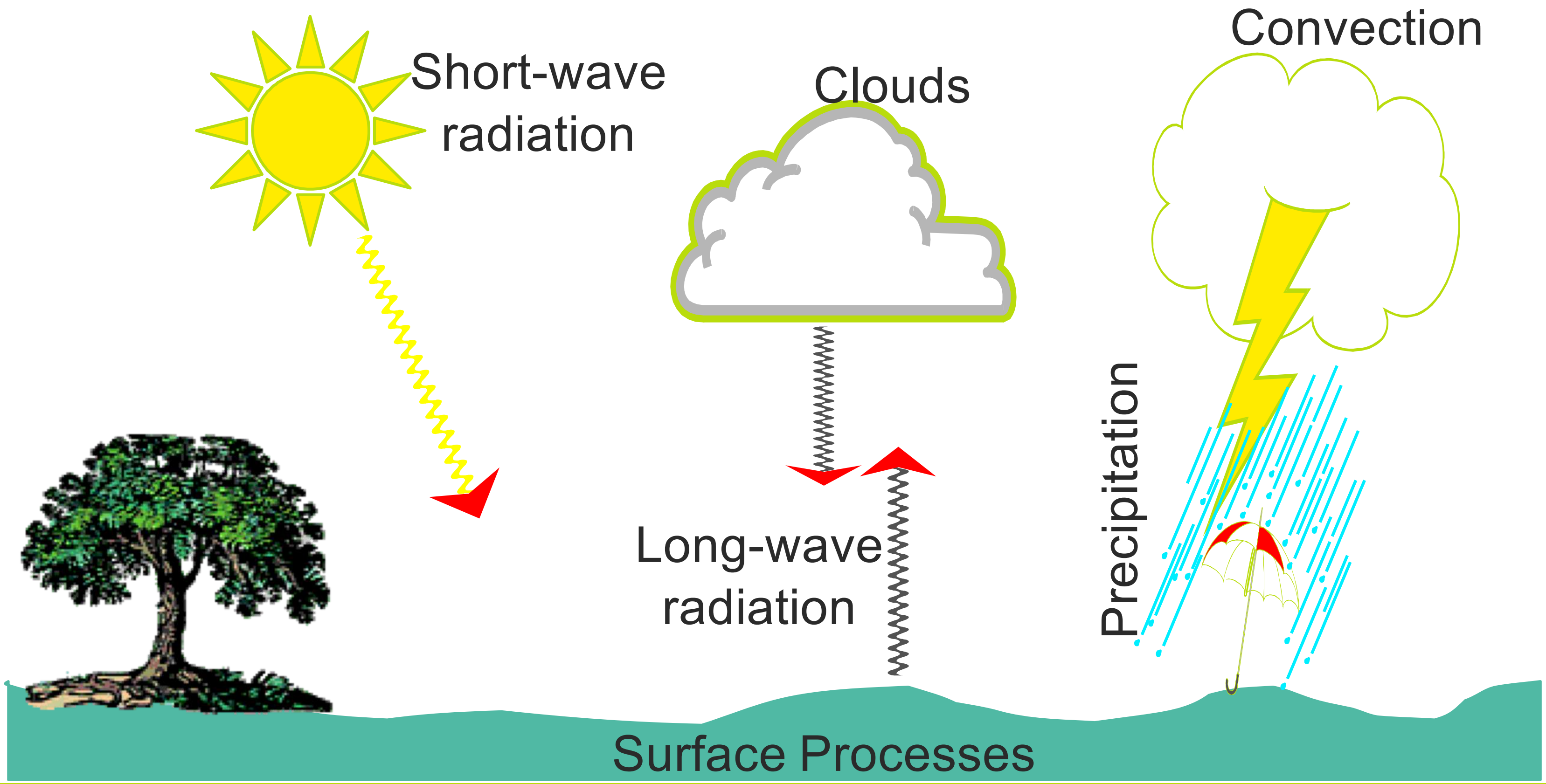
i) Discretisation (space - grid, and time)
ii) Method
iii) Order
iv) Time-stepping scheme
v) Solution method

Different patterns of computation, computational and data dependency, and communication



Met Office Physical parameterisations University of Reading

Processes not resolved at grid scale



Met Office

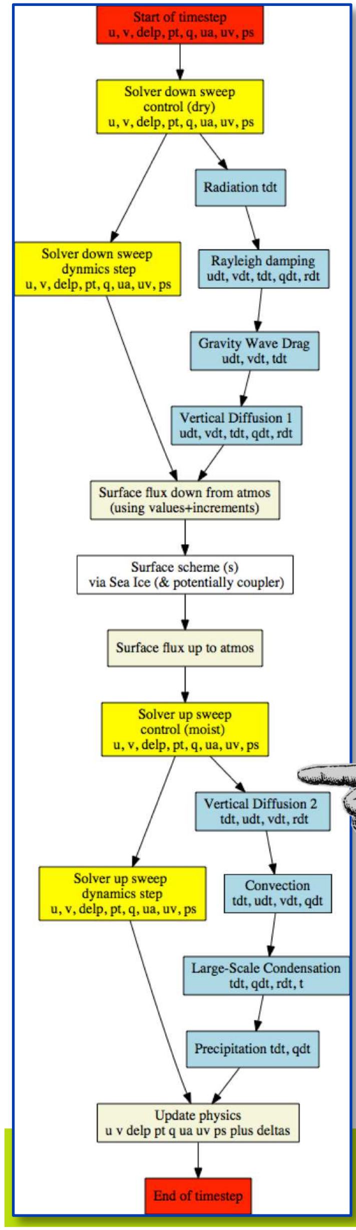
Time-step components

University of Reading

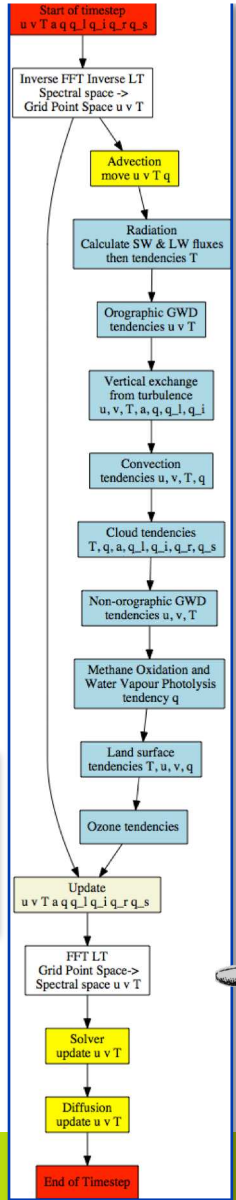
Dynamics
- Advection
- Solver

Physics
- Fast physics
  - cheap to compute, varies quickly
- Slow physics
  - costly to compute, varies slowly

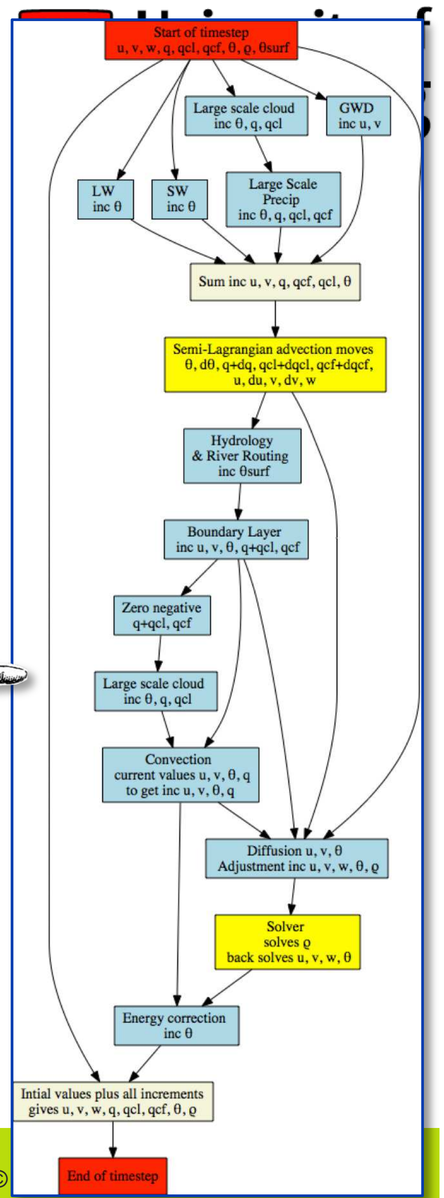
Different methods and algorithms for solving the problems. *Crossing the Chasm: how to develop weather and climate models for next generation computers?* B.N. Lawrence *et al.* (Geosci. Model Dev., 11, 1799–1821, 2018 https://doi.org/10.5194/gmd-11-1799-2018)



High level view of a time-step



Red: Start/end
Yellow: Dynamics
Blue: Increment Physics
Beige: Physics, full values
White: Coupling/library



GFDL Hiram

Met Office Unified Model

ECMWF IFS

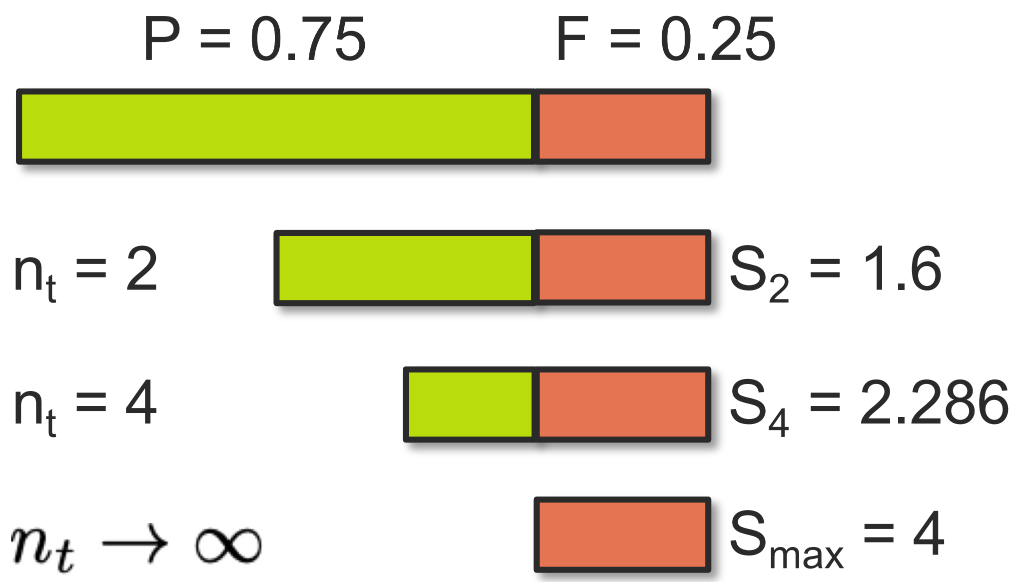
Parallel scaling

Amdahl's Law

P is proportion of program which is parallelisable

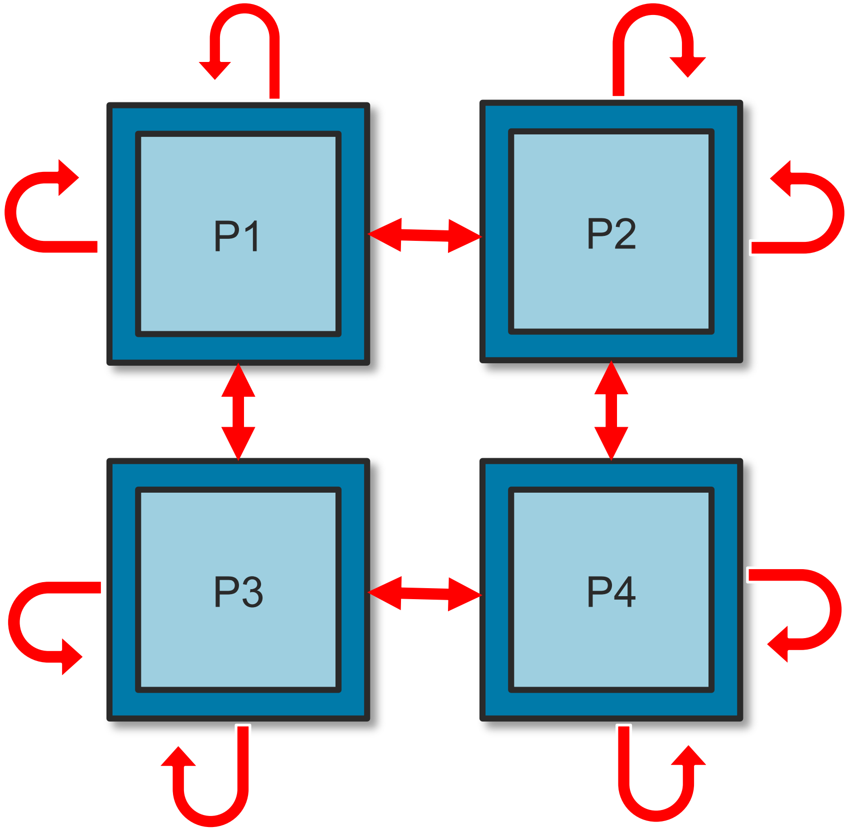
S, maximum speed up achievable compared to serial code is

$$S_{\max} = \frac{1}{1-P}$$



Even if all parts of program are parallelised, they have different scaling behaviour due to *communication* between parallel elements

Parallel Communication I



Local communication
Stencil-type calculations require data from neighbour
Halo exchange
Stencil size → halo depth
(e.g. Semi-lagrangian → large halos)
Point-to-point
Bandwidth limited

Parallel Communication II

Global communication
All parallel elements take part

- Reductions – global sum (iterative solvers)
    - latency bound
- All-to-all – spectral transforms
    - latency and bandwidth bound
- I/O -- Serial data to parallel memory and vice versa
    - latency, bandwidth and raw data rate bound

Supercomputer turns a compute bound problem into an I/O bound problem. *Ken Kennedy*

Strong and Weak scaling

Weak

Keep local problem size fixed

-- data size per parallel element is fixed – work rate constant

Local communication increases across whole problem, but not per PE

Global communication increases

Strong

Keep problem sized fixed

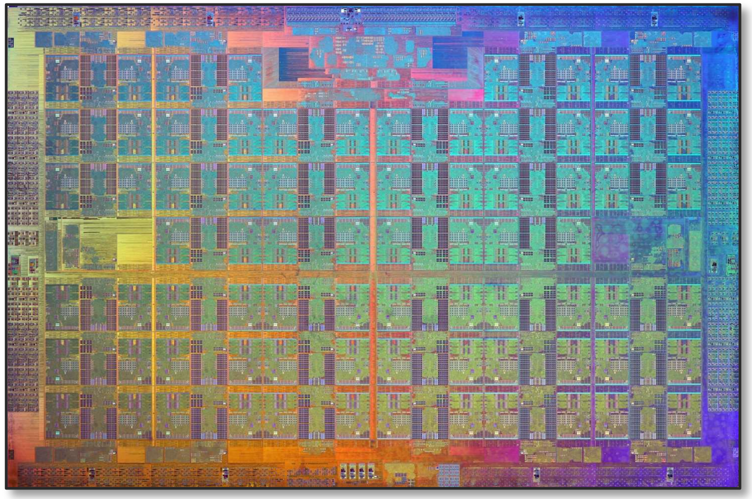
-- Size of globe is fixed, but resolution is not

- Amount of work per parallel element decreases (solve faster)
- Local communication decreases (but slower)
- Global communication increases

Strong scaling regime -- communication dominates

Levels of parallelism

This simple model of parallelism doesn't map onto modern, complex processors. Typically exhibiting multiple levels of parallelism and requiring multiple programming models to exploit them.

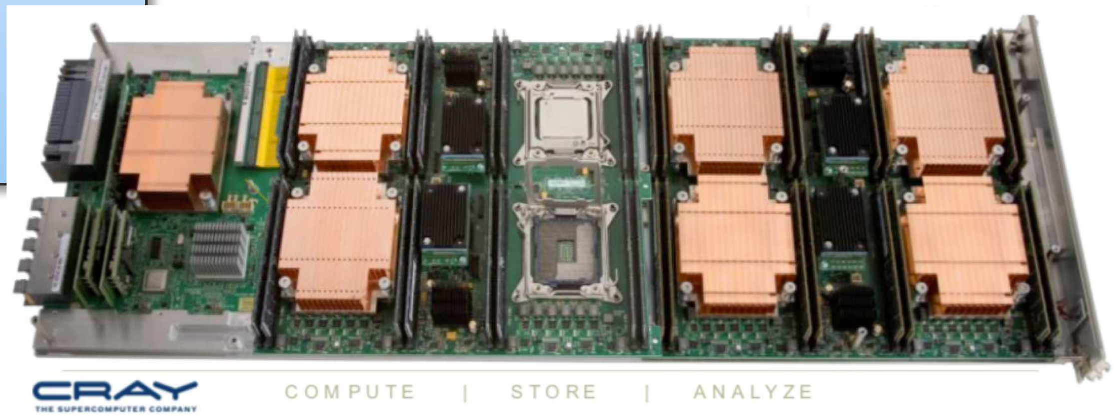


MPI + X
Where MPI is used for inter-node distributed memory
X is intra-node parallelism
Usually OpenMP/OpenACC

Node Comparison

Met Office Cray XC40 (32 on top500)
Dual socket 18-core Intel Xeon
Non-Uniform Memory Access
256 bit AVX SIMD ILP
6000+ nodes
Can Program MPI only
MPI + OpenMP is common

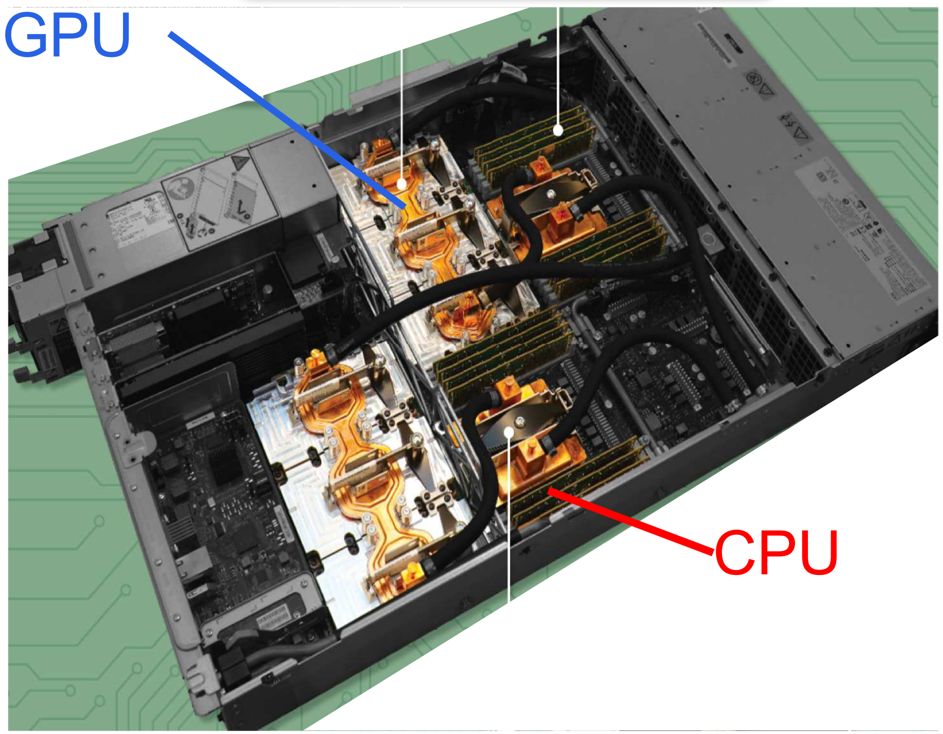
Whole machine
3 MW



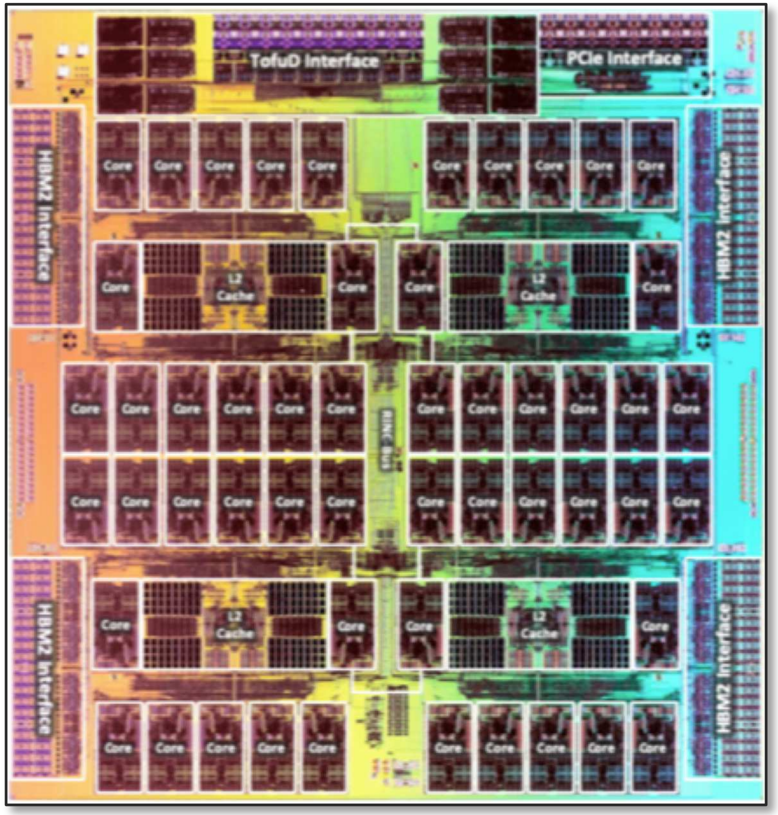
Summit

Oak Ridge National Lab (ORNL), USA
2 on Top 500
dual IBM Power 9 22-core CPU
+ 6 NVIDIA VOLTA GPU (4000+ nodes)
Host and device memory
NV-link connections
84 streaming multiprocessors
Each SM has 64/32 32/64-bit cores
Hierarchy blocks, warps and threads
Oversubscribed concurrency
Tens of thousands of SIMT threads per GPU

Whole machine
15 MW



Fugaku



Riken, Japan, 1 Top 500
Fujitsu 64-bit ARM processor
48 cores, 4*12 mini-NUMA
Each has 512-bit Scalable Vector Extension SIMD (ILP)
150,000+ nodes
7M+ cores

Whole machine
28 MW

Real model scaling

Unified Model

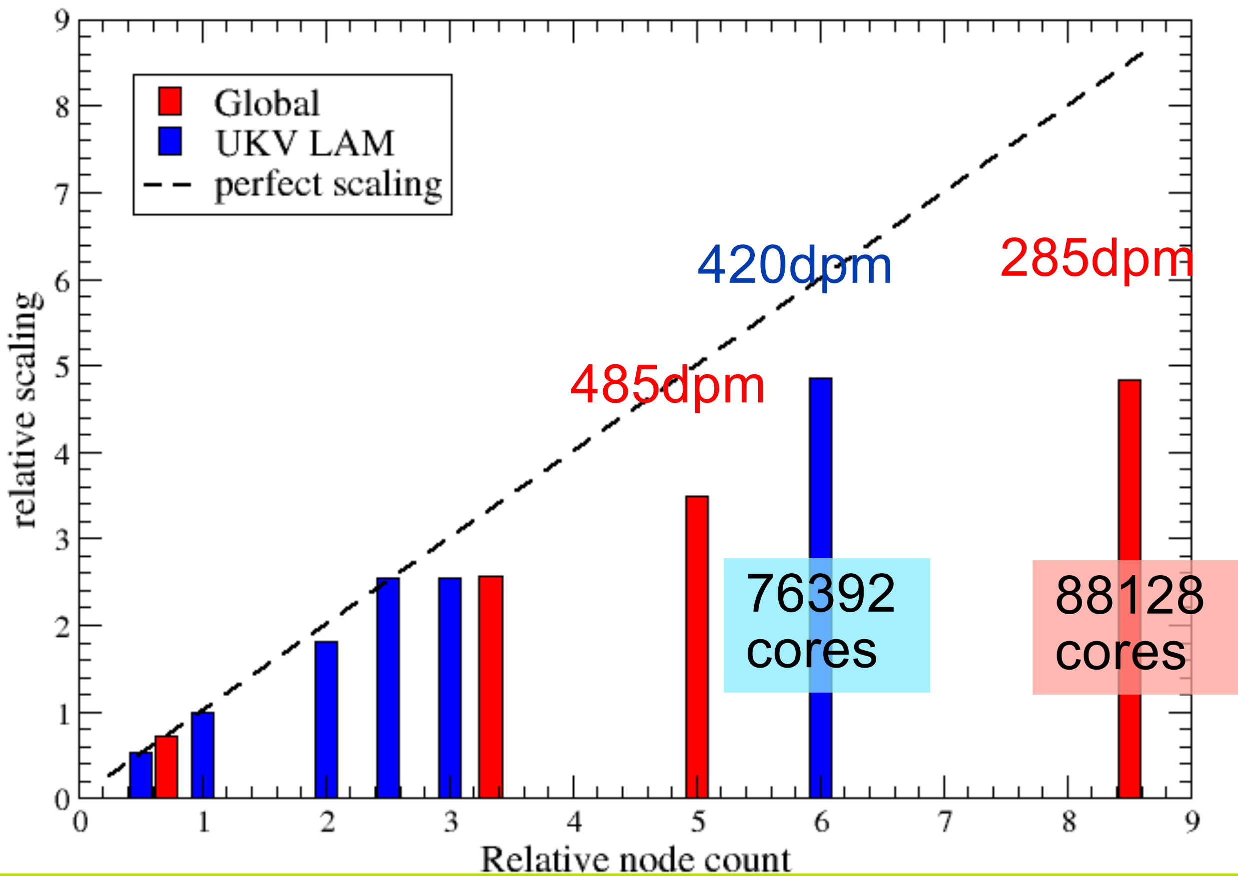
Other models are available!

UM scaling

Comparing relative scaling of Global Model (N2048 ~6km) and Limited Area Model (LAM)

Normalise relative to 2nd datum
Both have similar points per MPI task ~ 2500
LAM is scaling much better than Global

(Selwood & Malcolm)



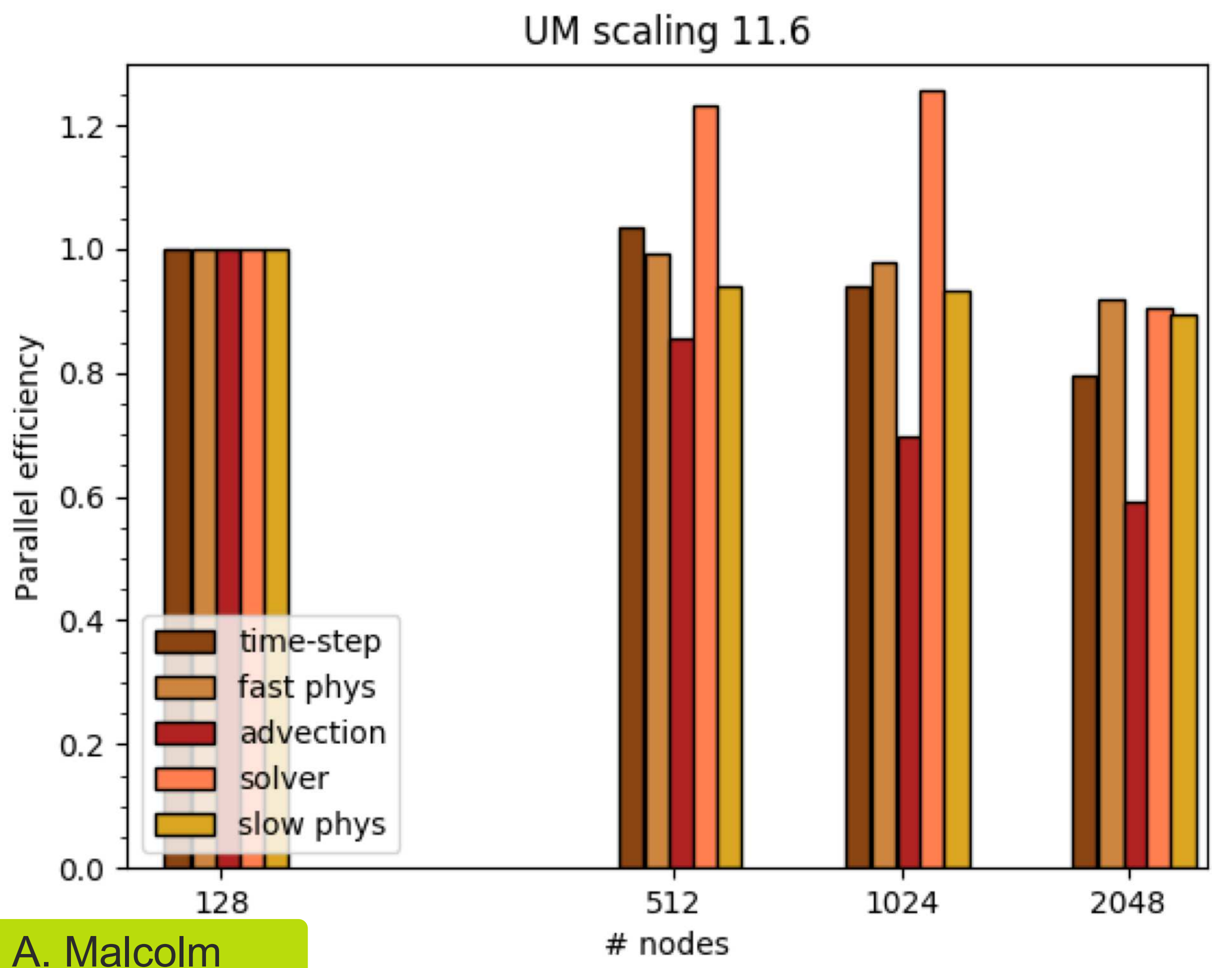
Parallel efficiency

Met Office

University of Reading

$$PE = \frac{T_n/T_0}{N_n/N_0}$$

UM11.6 Global N2048 ~ 6km resolution
Met Office XC40
MPI+OMP – 3 (6) threads
Time-step scales OK.
Physics scales well (no comms)
Solver –super-linear (memory)
Advection scales poorly

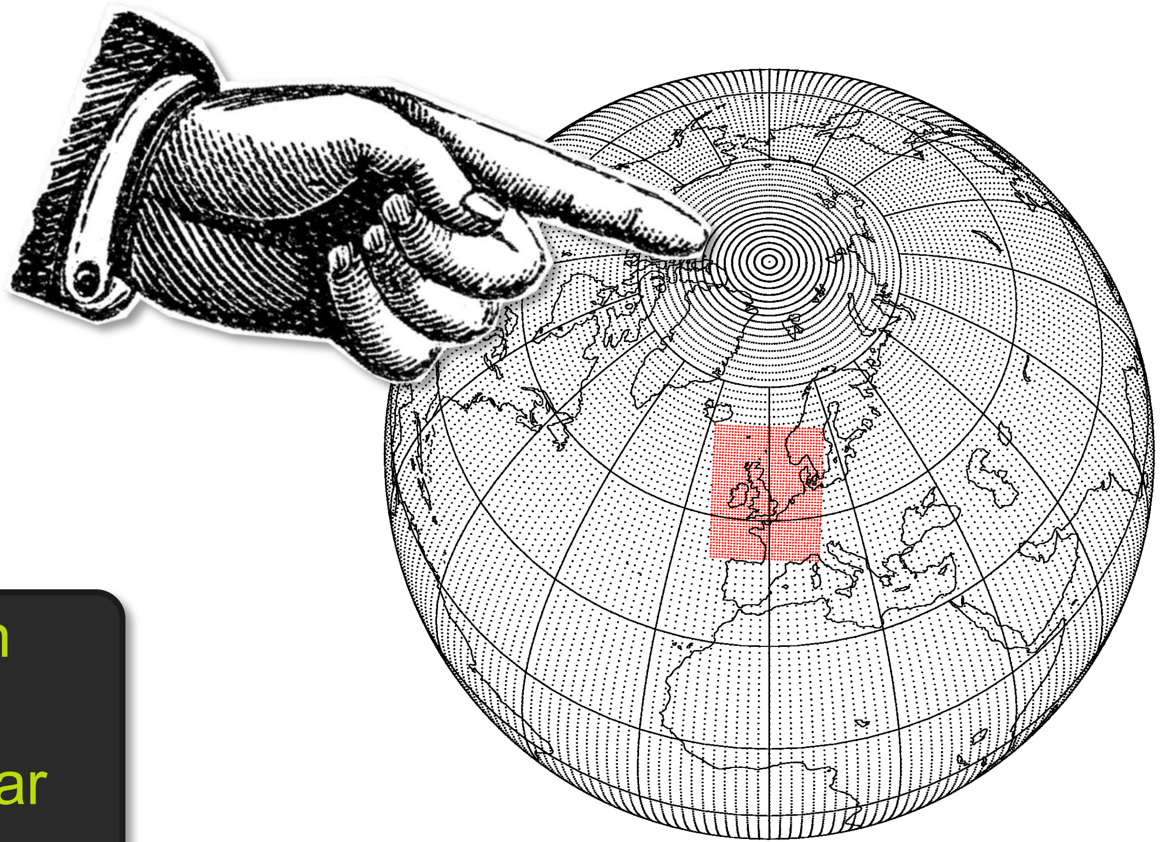


A. Malcolm

The Finger of Blame …

At 25km resolution, grid spacing near poles = 75m

At 10km reduces to 12m!



Semi-Lagrangian Advection
→ Large halos
→ Lots of communication near poles

MPI+OMP

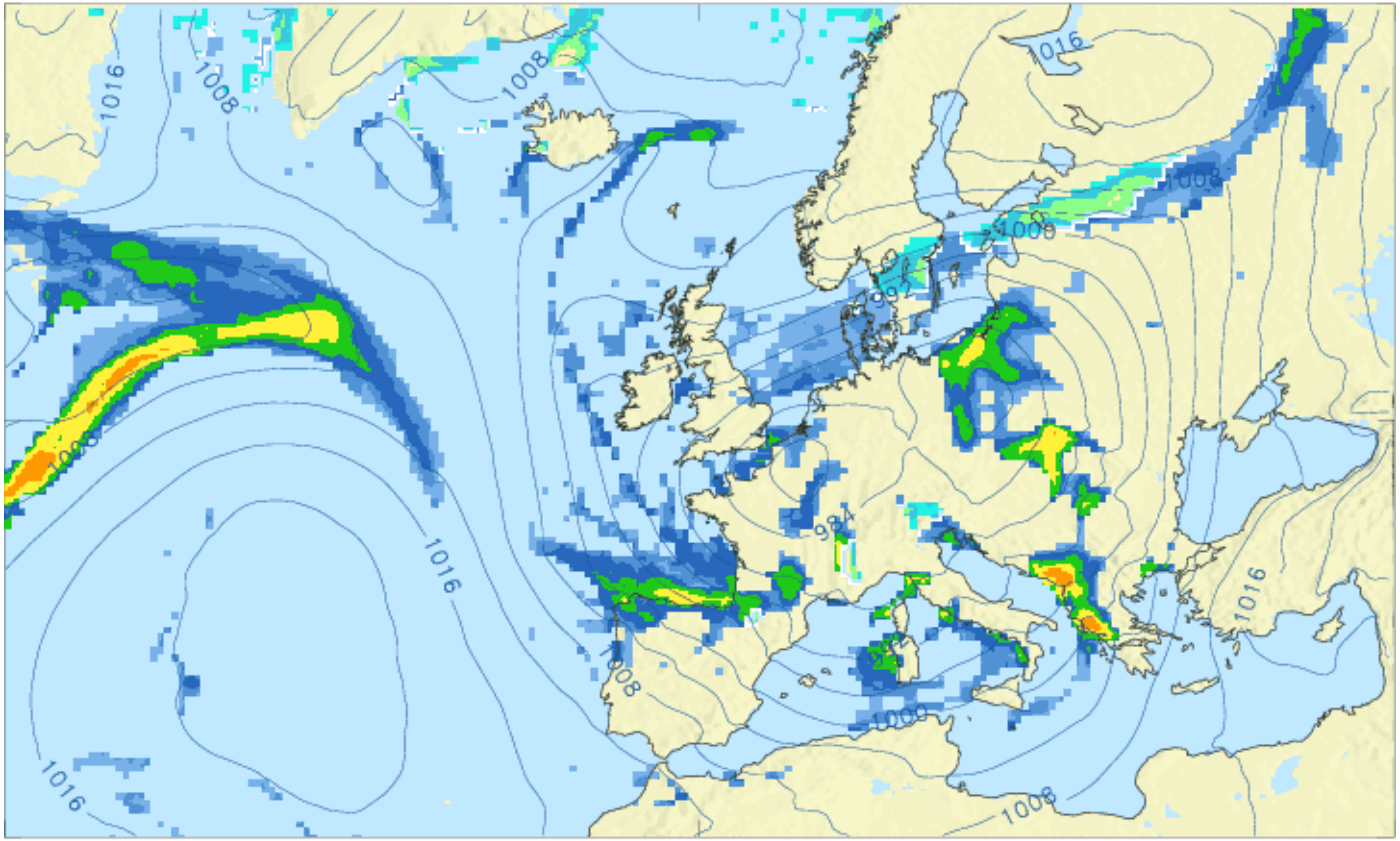
MPI on-node communication is efficient
However, OpenMP reduces the amount of OpenMP required and balance of computation
Glover et al CUG 2016

Code Section	36x60_2 Baseline	72x90_2 MPI	36x60_6 Threads
U_MODEL4A	1792	829	825
ATM_STEP_4A	1571	624	599
AS SOLVER	510	176	158
AS S-L Advect	356	183	153
AS Atmos_phys2	344	121	135
AS Atmos_phys1	283	105	114
INITIAL	209	197	218

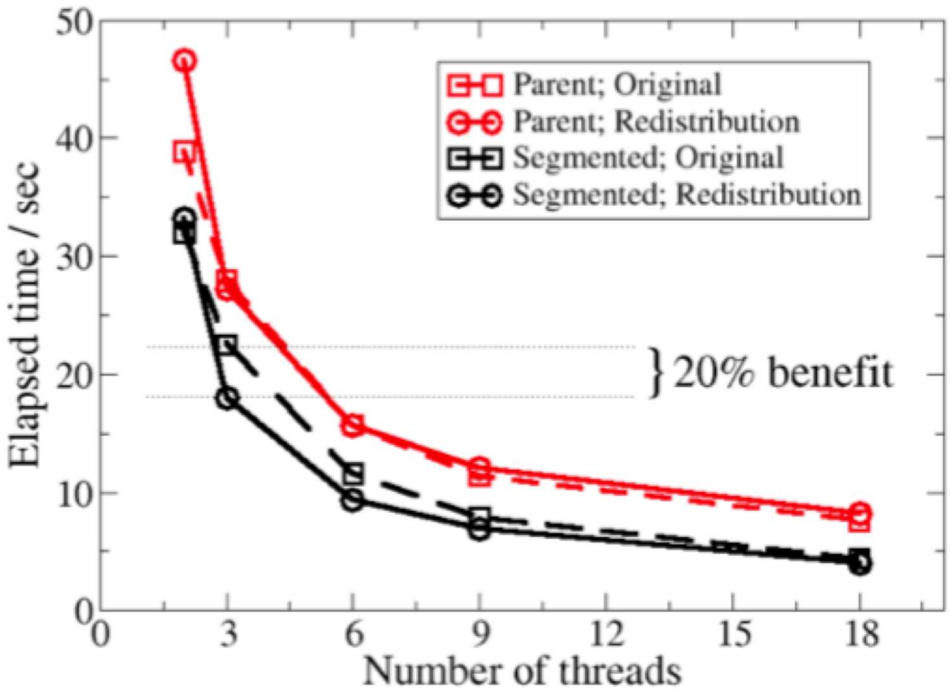
Glover et al CUG 2016 time in seconds (MPI E-WxN-S_NOMP)

Solver and Advection both have lots of communication → OMP benefit
Physics has not much, but lots of loops to thread (2016) and potentially poor load balance

Weather is not uniform



SW radiation



Lower is better

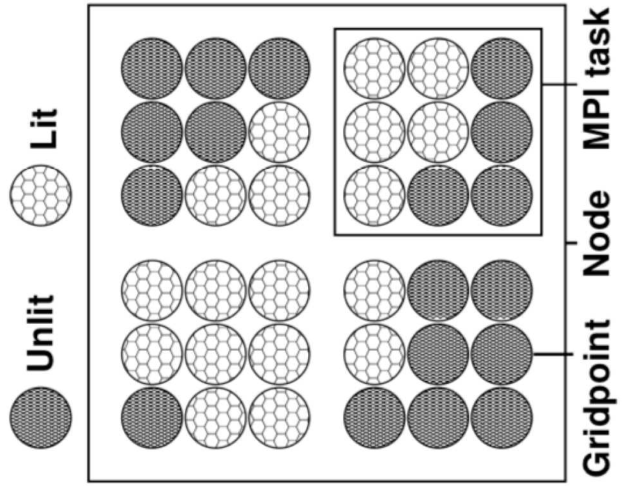


Fig. 6. Illustration of on-node load imbalance. Only lit points require computation.

Lit points have to be determined. Redistribution for load balance (cost), all threads have similar amount of work.

Met Office

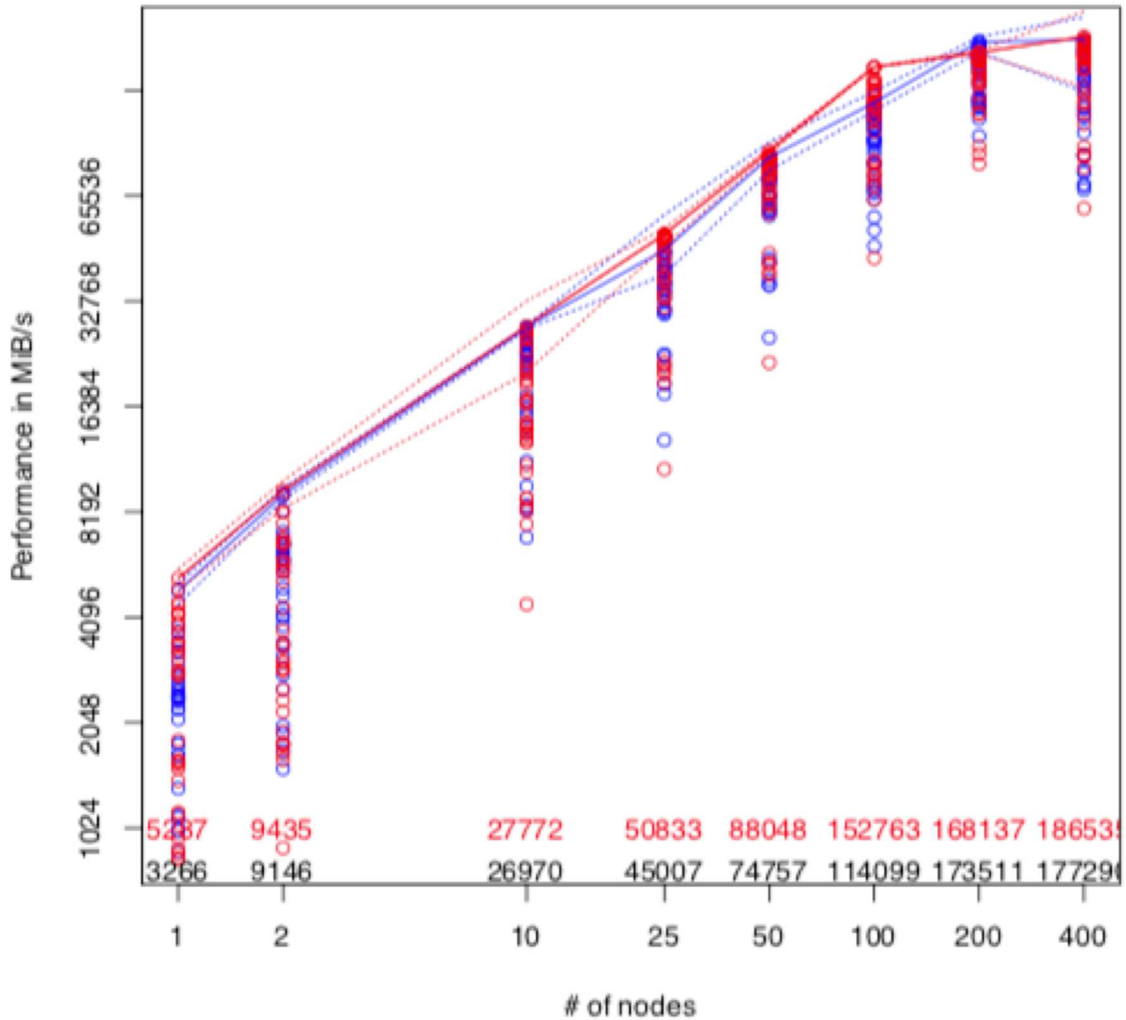
IO Server

University of Reading

Models produce lots of data.
Higher-resolution means more data.
IO Server avoids model computation waiting whilst output is done
Dedicated (MPI) resource to do output only
Most PEs do computation, asynchronous offload of data to IO server resources which write data, whilst computation continues.
How many IO servers to compute PEs depends on machine characteristics, problem size, diagnostics selected, compute speed compared to IO speed.

IO performance

Plot from JM Kunkel
DKRZ supercomputer
Showing different numbers IO throughput for different numbers of clients and servers, processes per node, tunable IO parameters, read/write
Best performance gives
~6GiB/s per node (small cfg)
~1.5 GiB/s per node (large cfg)



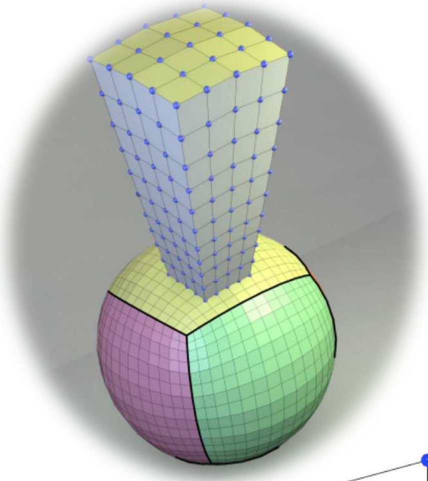
Gung Ho and LFRic

Lon-lat grid will ultimately prevent UM scaling → changing the grid changes everything

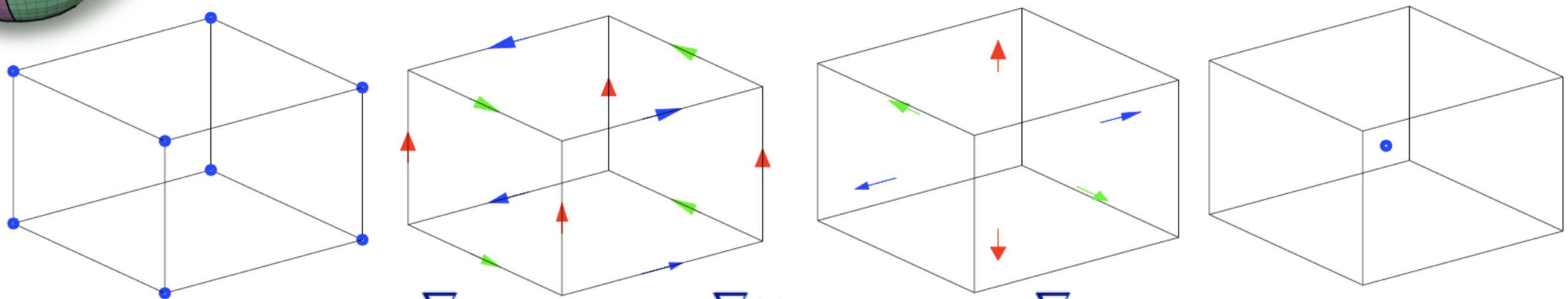
LFRic: Adams *et al* JDPC V132 (2019) 383-396 DOI:[10.1016/j.jpdc.2019.02.007](https://doi.org/10.1016/j.jpdc.2019.02.007)

Gung Ho: Melvin *et al* *Q J R Meteorol Soc*. 2019; 145: 2835- 2853. https://doi.org/10.1002/qj.3501

Gung Ho dynamical core



- Cubed Sphere → no singular poles lon-lat
- Unstructured mesh → can use other meshes
- Mixed finite element scheme – *C-Grid*
- Exterior calculus *mimetic* properties
- Semi-implicit in time



$$\begin{tikzcd}
\mathbb{W}_0 \arrow[r, shift left, "\nabla"] & \mathbb{W}_1 \arrow[l, shift left, dashed, "\tilde{\nabla}."] \arrow[r, shift left, "\nabla\times"] & \mathbb{W}_2 \arrow[l, shift left, dashed, "\tilde{\nabla}\times"] \arrow[r, shift left, "\nabla."] & \mathbb{W}_3 \arrow[l, shift left, dashed, "\tilde{\nabla}"]
\end{tikzcd}$$

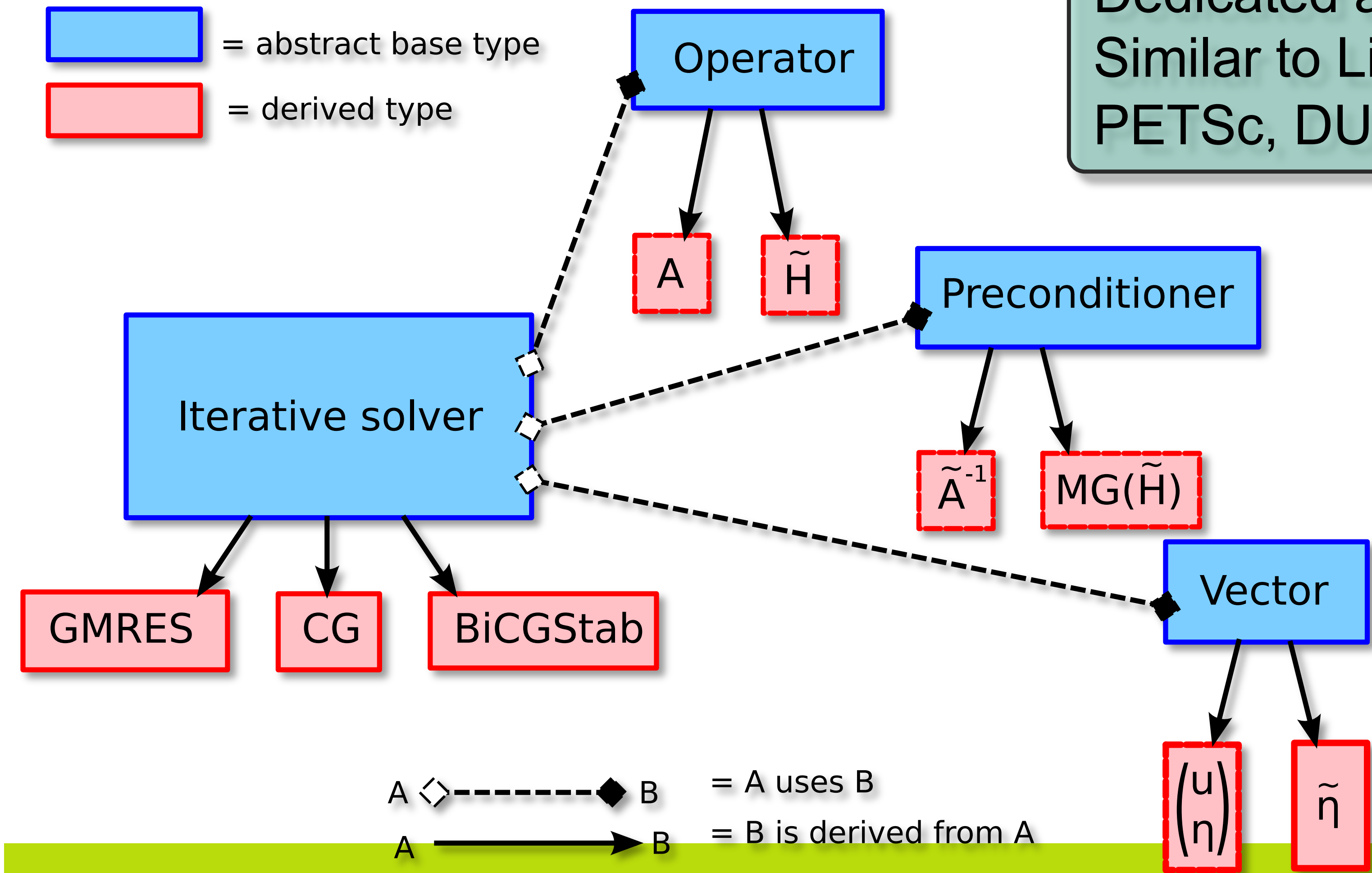
Krylov subspace solvers

$$\mathbf{A} \cdot \vec{x} = \vec{b}$$

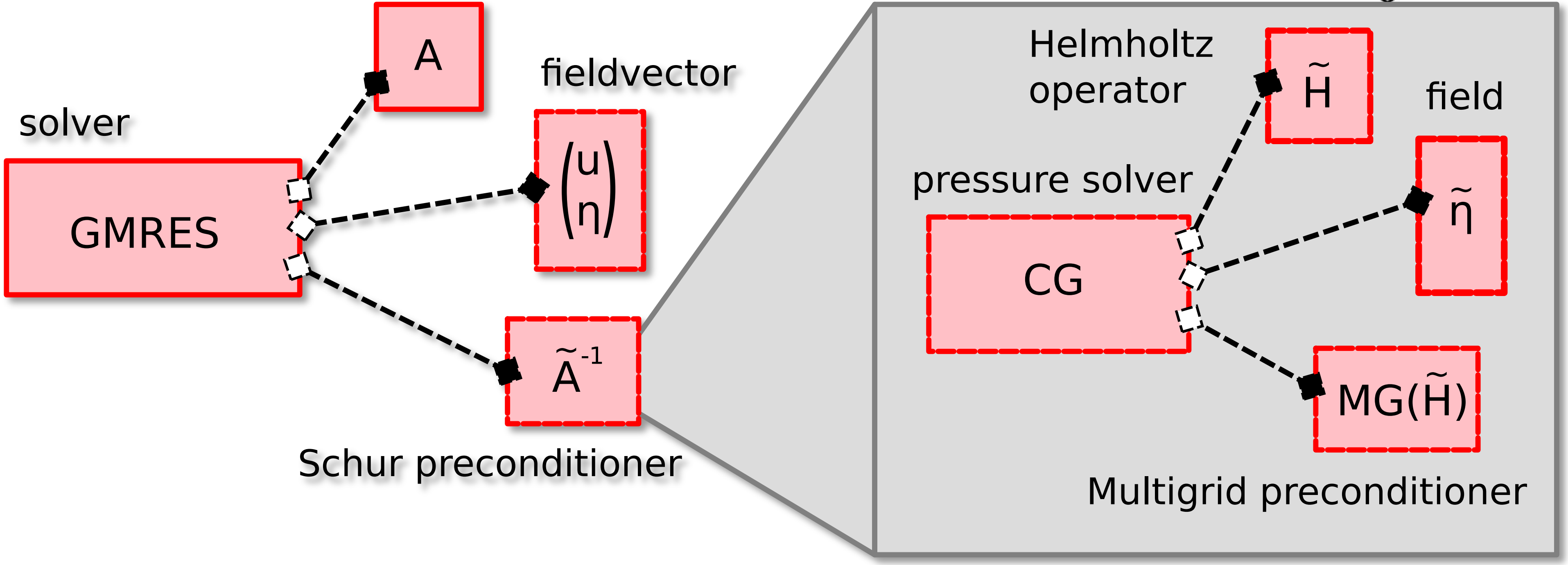
Many different types
Build an improved solution based on previous
Compute intensive Matrix $\times$ Vector (linear operator – part of the subspace)
Scalars computed from norms of vectors
Global sum (MPI_Allreduce)
Fewer iterations → fewer global sums
preconditioners

The solver

Dedicated abstraction in F2K3 OO
Similar to Lin. Alg Libs e.g.
PETSc, DUNE-ISTL, Trillinios



The solver II



Allows for easy implementation of sophisticated nested solver
Multigrid preconditioner - reduce work for iterative solver
- faster and less global sums (better scaling)

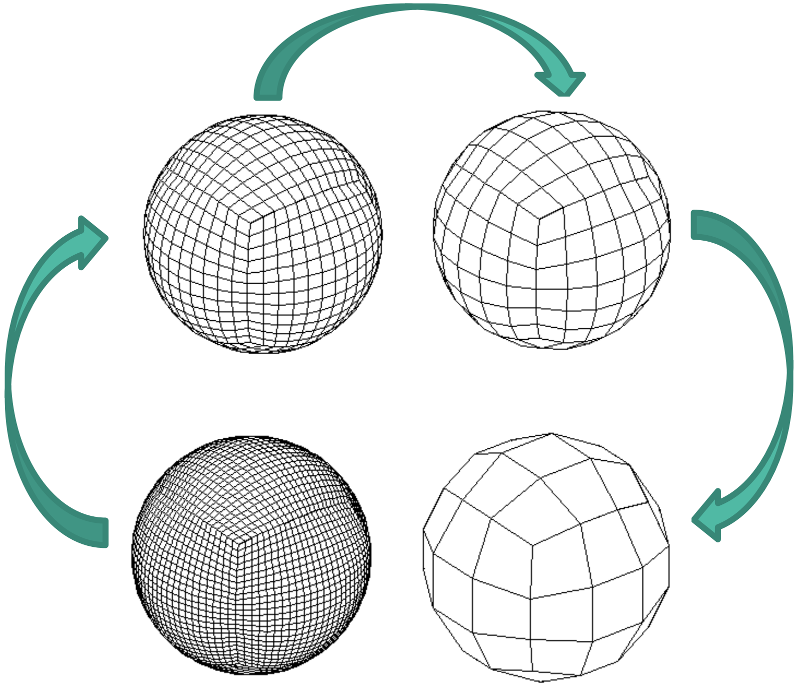
LFRic Multigrid

- Helmholtz system $H\Pi' = R$ solved using a single Geometric-Multi-Grid V-cycle with block-Jacobi smoother

$$H = M_3^{\Pi^*} + \left(P_{3\theta}^{*}\mathring{M}_\theta^{-1}P_{\theta 2}^{\theta^*,z} + M_3^{\rho^*}M_3^{-1}D^{\rho^*}\right)\left(\mathring{M}_2^{\mu,C}\right)^{-1}G^{\theta^*}.$$

- Block-Jacobi smoother with small number (2) of iterations on each level
- Exact (tridiagonal) vertical solve: $\widehat{H}_z^{-1}$

$$\widetilde{\Pi}' \leftarrow \widetilde{\Pi}' + \omega \widehat{H}_z^{-1}\left(\mathcal{B} - H\widetilde{\Pi}'\right)$$

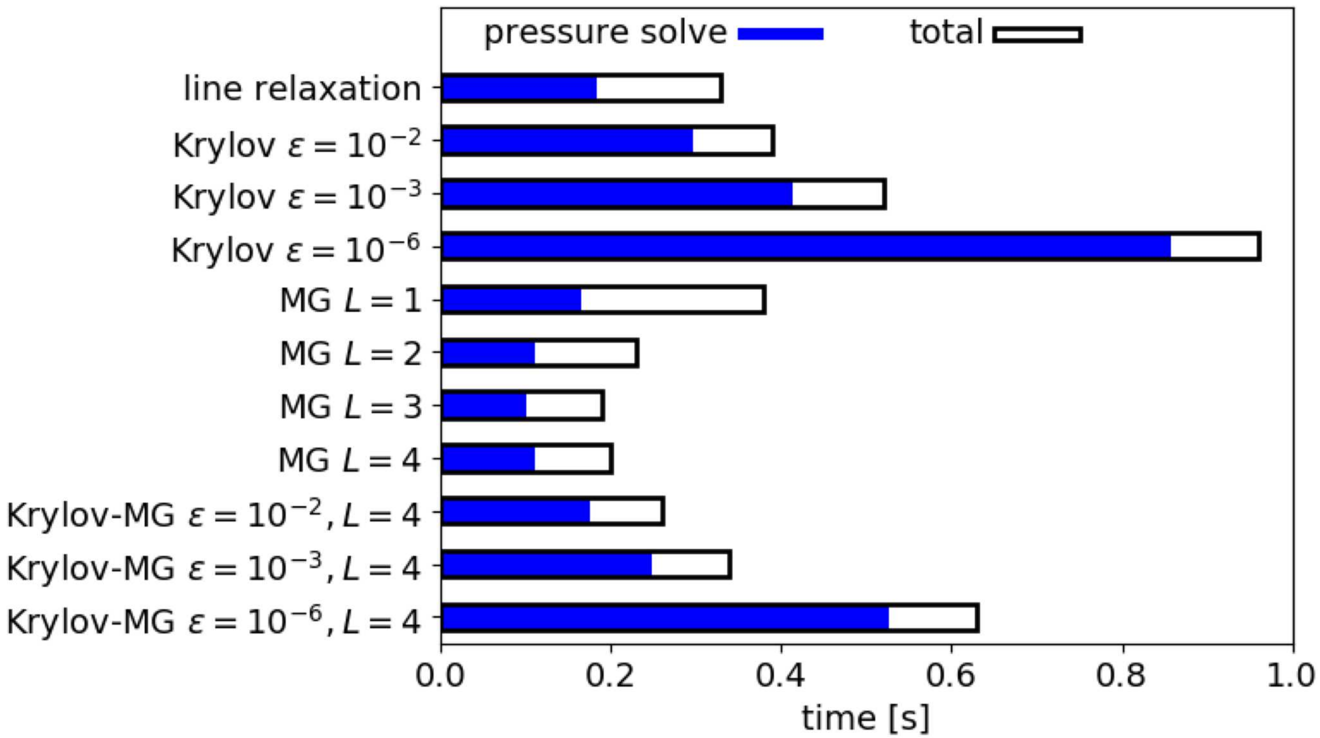


Maynard, Melvin and Mueller, QJRMetS
https://rmets.onlinelibrary.wiley.com/doi/10.1002/qj.3880

Initial Results

C192 cubed sphere with 30 L (~50Km) Baroclinic wave test Met Office Cray XC40 64 nodes (2304 cores) Mixed mode 6 MPI/6 OMP threads

c.f. $||r|| = ||\mathbf{A}x - b||$ Of Krylov 10^{-2}
Before and after MG 3-level V-cycle



Met Office Scaling runs

University of Reading

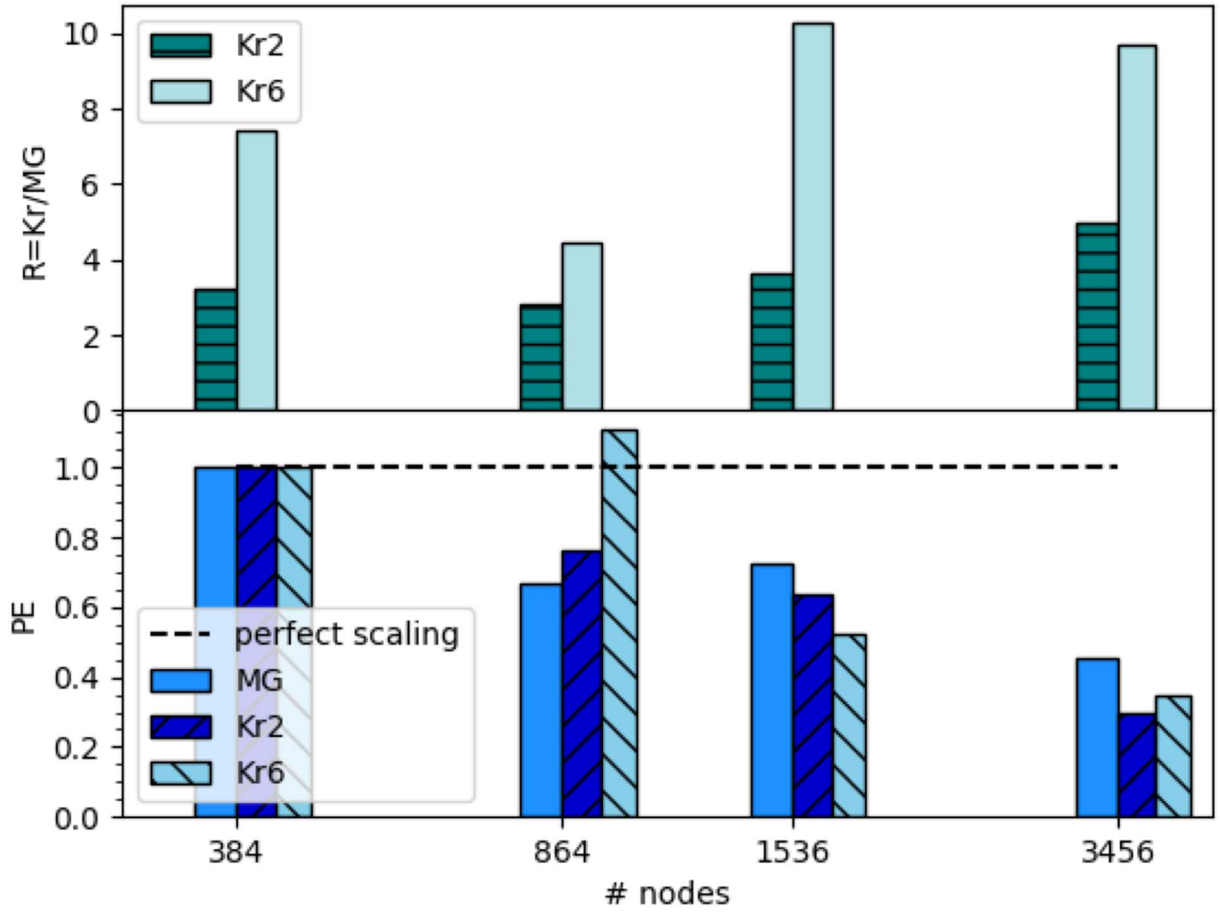
C1152 mesh → 1152 X 1152 X 6 mesh with 30 levels – 9Km resolution
Dynamics only, (Baroclinic Wave)
400 time-steps. Δt = 205 s CFL_H (acoustic) ~ 8
Intel 17 compiler
6 MPI ranks per node, 6 OpenMP threads per rank
384 nodes (13824 cores) – 3456 nodes (124416 cores)
Data per PE 24x24, 16x16, 12x12, 8x8
MG is 3-levels of MG inner solve is preconditioner only
KR2 is $||r||=10^{-2}$
KR6 is $||r||=10^{-6}$

Met Office

Semi-implicit solver

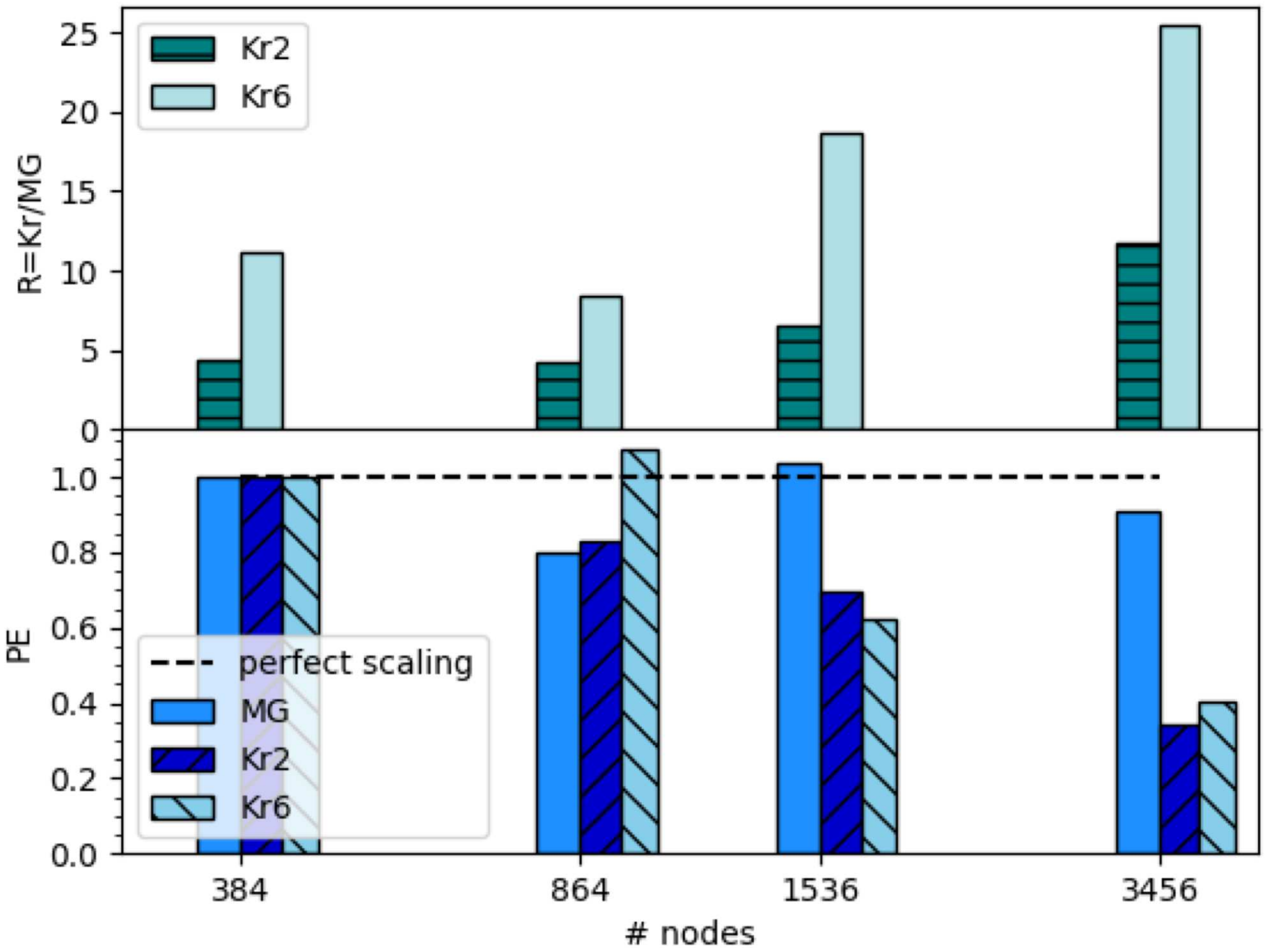
University of Reading

Bottom panel is parallel efficiency
1 is perfect scaling
Top panel shows relative cost of KR solvers *c.f.* MG
(higher is better)



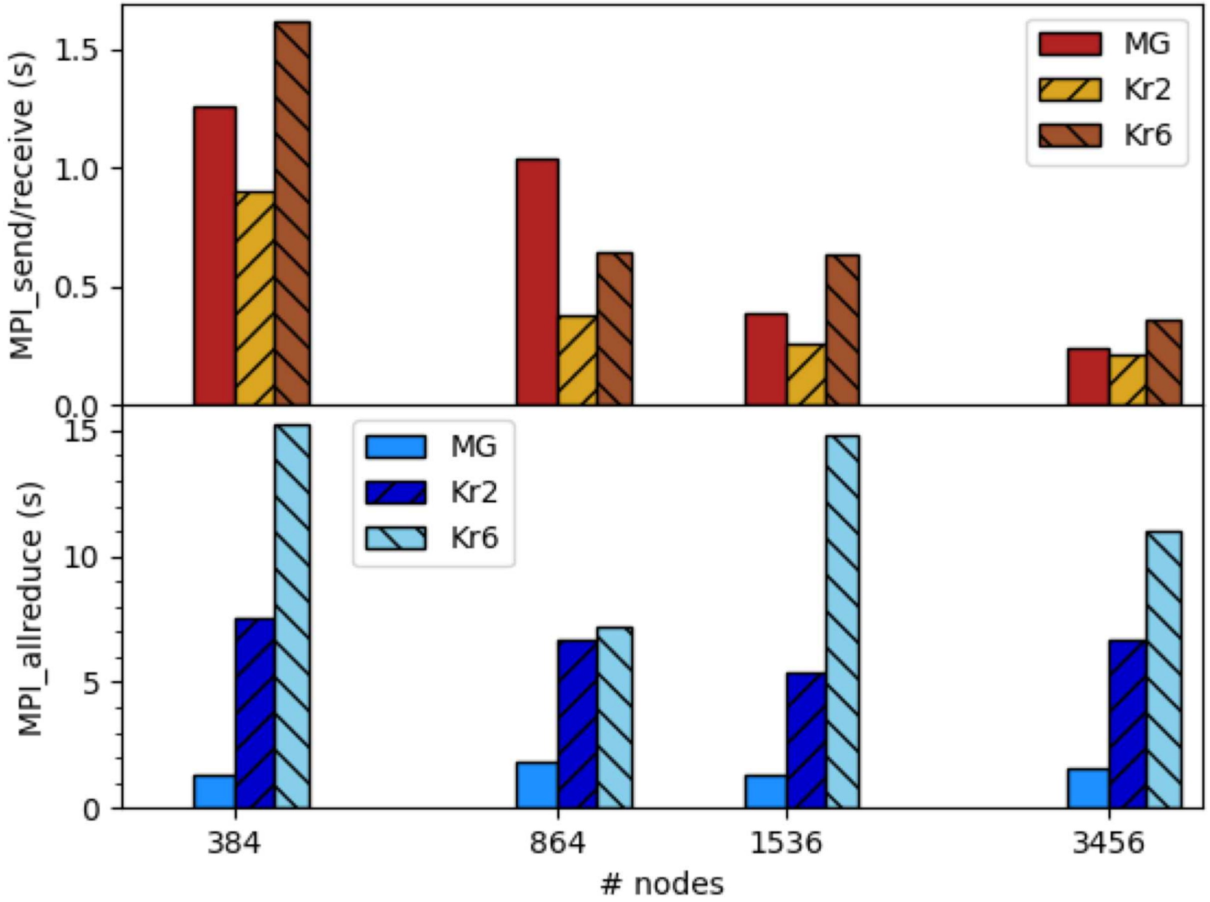
Pressure solve

Bottom panel is parallel efficiency
1 is perfect scaling
Top panel shows relative cost of KR solvers *c.f.* MG (higher is better)
MG is much faster and scales much better



Communication costs

- Time per time-step in communication
- Bottom panel is MPI allreduce (global sum) cost
- Massive reduction for MG
- Upper panel shows local comms scale with data size

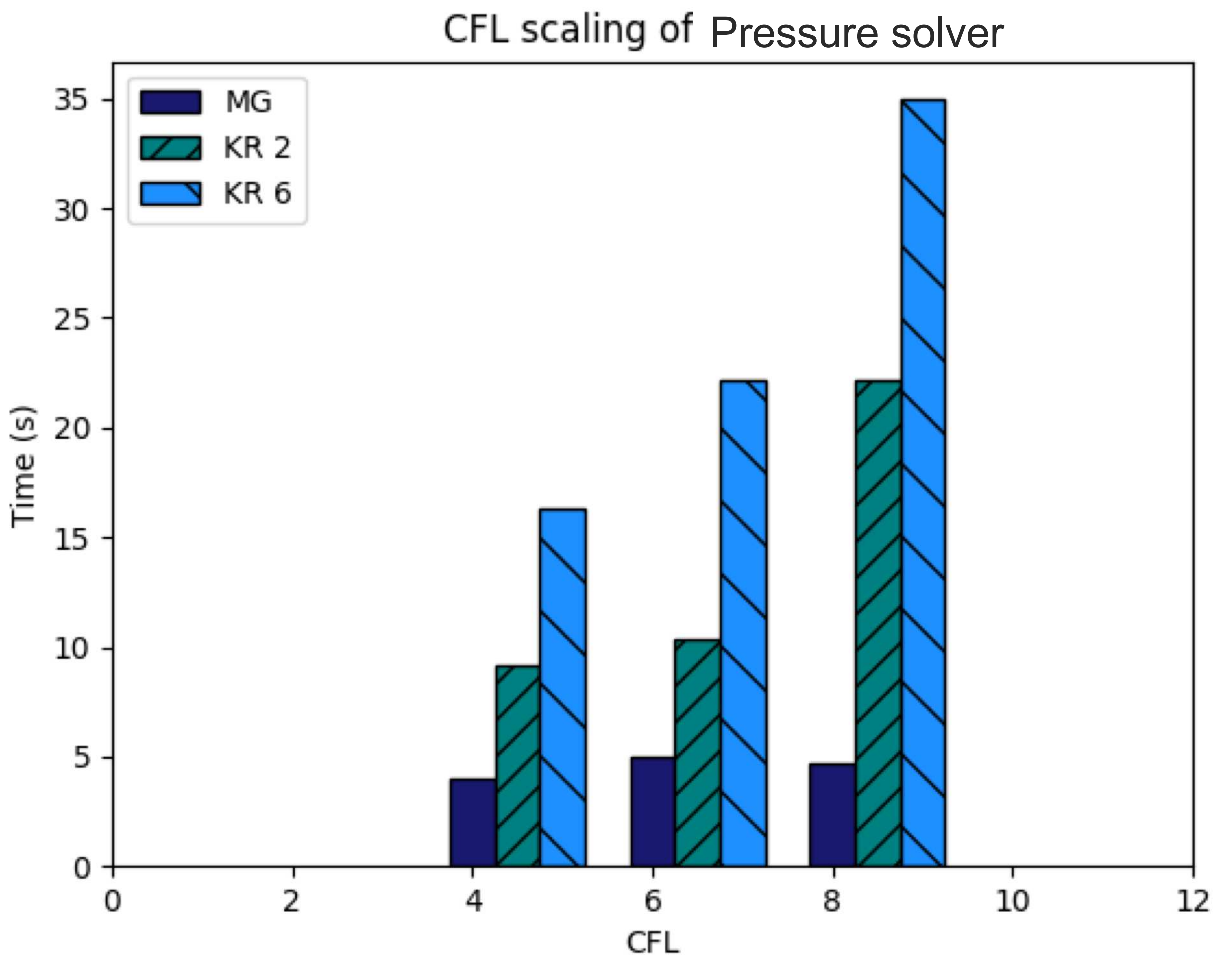


Size of time-step

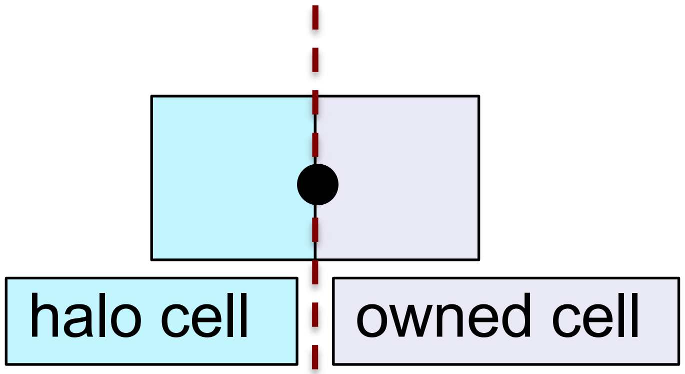
$\mathrm{CFL_h}$	Solver	# iter mixed	# iter pressure	$T^{(\mathrm{m})}_{\mathrm{solve}}$
4	MG	12.9	–	3.96
	Kr2	13.2	9.4	9.13
	Kr6	12.0	26.2	16.29
6	MG	13.6	–	4.98
	Kr2	13.3	13.2	10.31
	Kr6	12.3	40.4	22.16
8	MG	14.0	–	4.70
	Kr2	13.3	17.3	15.15
	Kr6	12.6	54.2	34.96

CFL scaling

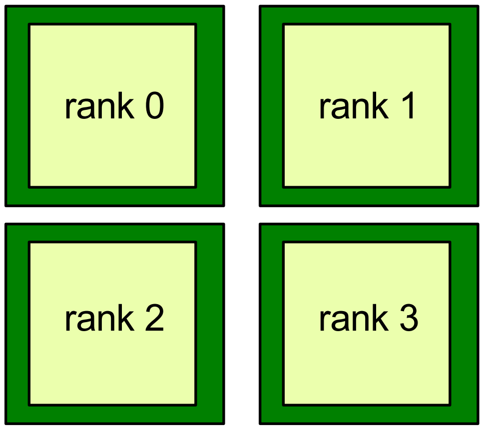
As time-step increases, condition number of matrix increases
→ more iterations
Multigrid V-cycle is fixed cost
As long as solution is good enough, no extra cost to increase time-step



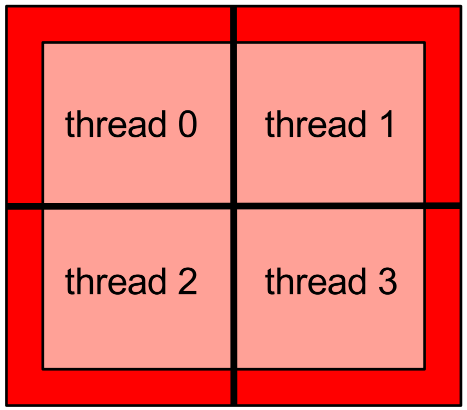
Redundant computation



Dof living on shared (partitioned) entity (edge).
Receive contribution from owned and halo cell.
Redundant compute contribution in halo to shared dof.
Less communication

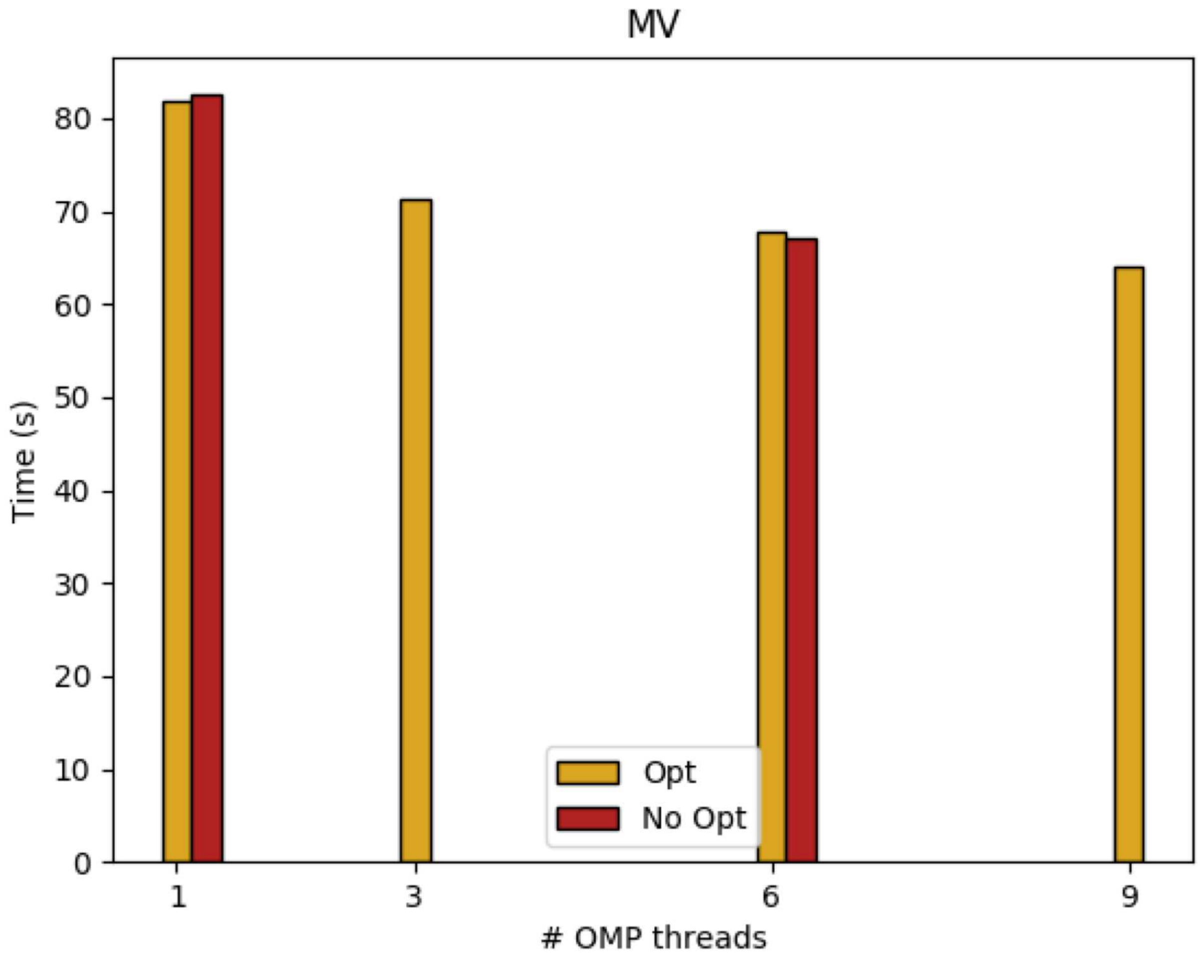


MPI only, 4 MPI ranks all have halos
Hybrid, 1 MPI task has a halo, 4 OpenMP threads share halo
boundary-to-area scaling
→ Less work for OpenMP threads



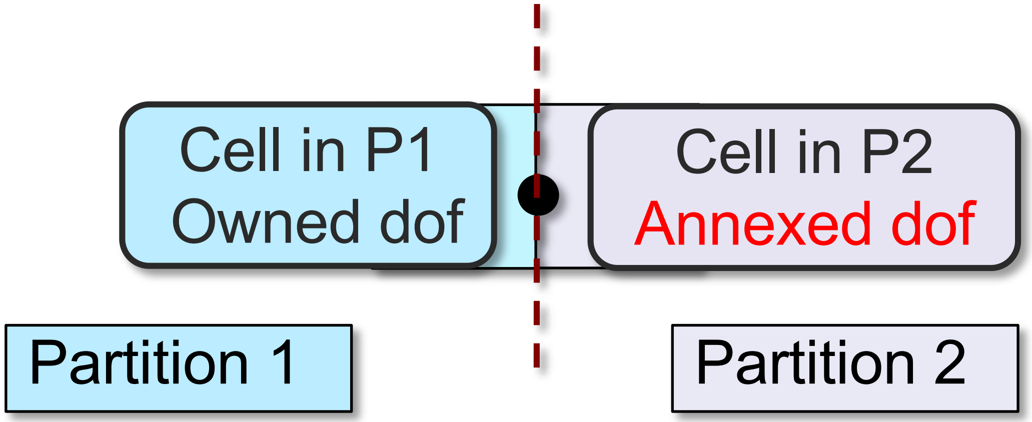
rank 0

Matrix-vector scaling



C288 mesh, 96 nodes
1 OMP thread is 36 MPI ranks per node
9 OMP threads is 4 MPI ranks per node
Redundant computation favours more threads
(opt/No Opt is MPI comms env variable – not relevant here)

Annexed dofs



Point-wise computations (e.g. set field to a scalar) loop over dofs
Looping to owned dofs ➔ halo exchange required for P2
Looping to annexed dofs is now transformation in Psyclone
Small increase in redundant computation
Large reduction in number of halo exchanges required

GPUs and other animals

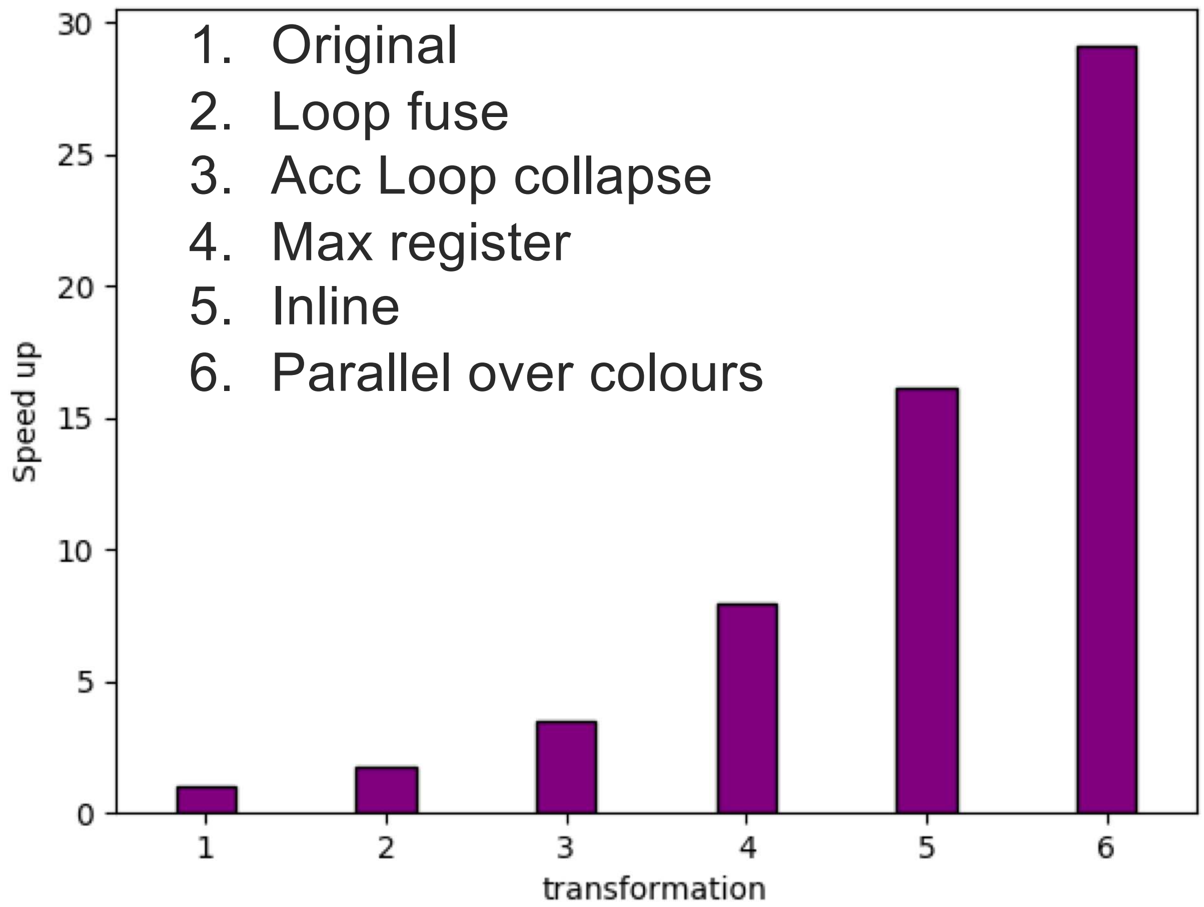
GPUs

Distinct memory spaces
Data transfer/synchronisation
ILP

Exploited data parallelism in horizonal for CPUs
Dynamics kernels tend to have limited data dependency in vertical
SIMD/SIMT GPU vector across vertical dofs – 128+ levels
Physics kernels often have dependency in vertical …
But have extra dofs, e.g. radiation bands
Exploiting CPU and GPU together with hard to synchronise
Simpler to reduce data movement and compute on GPU only

P9 + Volta GPU: MV-LMA

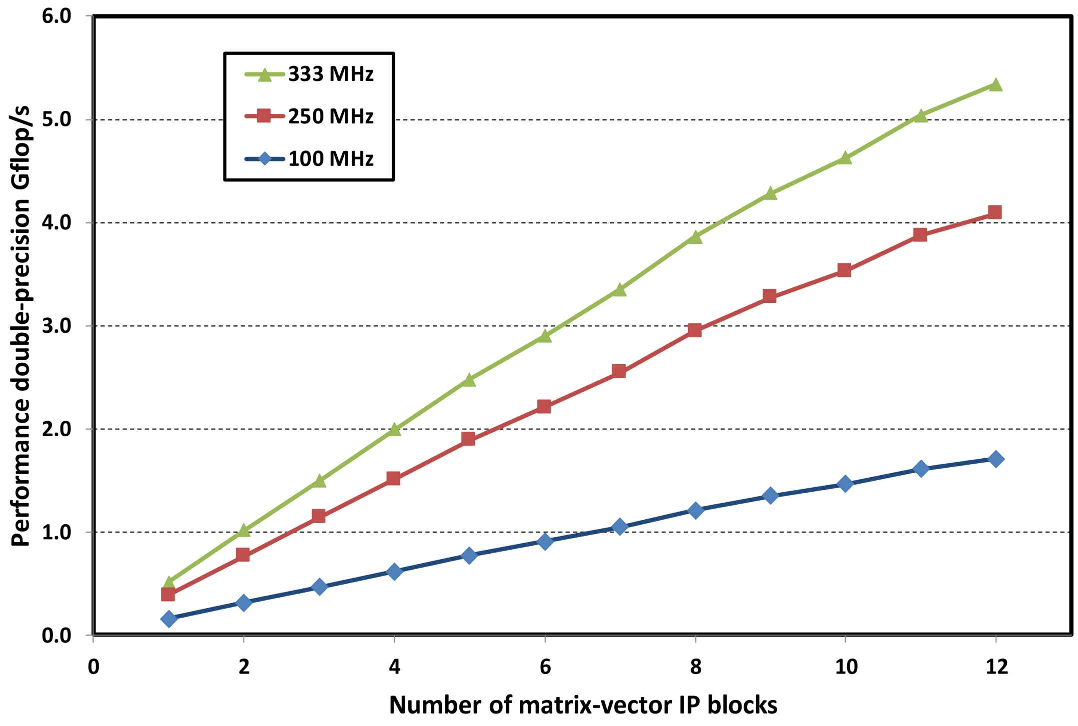
A. Gray (NVIDIA)
Cumulative speed up against original OpenACC code.
Problem size is too small for GPU.
Amortise cost of data movement by offloading multiple kernels



FPGAs

EuroExa project: ARM CPUs + FPGA accelerator prototype → low power
LFRic one of several applications. Mike Ashworth Uni of Manchester

Ported using High-level *Synthesis* tool from Xilinx Vivado.
Graph shows scaling versus IP block and clock speed.
Max 5.3 GFlop/s in double precision.
Comparable to CPU and GPU.
Significant benefits considering power.



Met Office

University of Reading

Conclusions

End of the *free lunch* – no faster processors → exploit ever more parallelism

Mathematics of problem dictates what can be computed in parallel

Choice of how to solve mathematics for weather and climate

Leading to different parallel algorithms and implementations

Interplay between implementations and parallel algorithms

Scaling of algorithmic components of time-steps

Newer architectures require exploitation of more parallelism

Met Office

University of Reading

Extra slides

If there is time

Projects

Gung Ho/LFRic/Psyclone Replace UM
UK Met Office + STFC + NERC
New FEM dynamical core + infrastructure
Code generator (automatic parallelism)
DSEL (Fortran)

Gridtools/Stella Rewrite
MeteoSwiss/CSCS (Cosmo/ICON)
Finite difference/structured mesh
Initially GPU code
DSEL (C++)

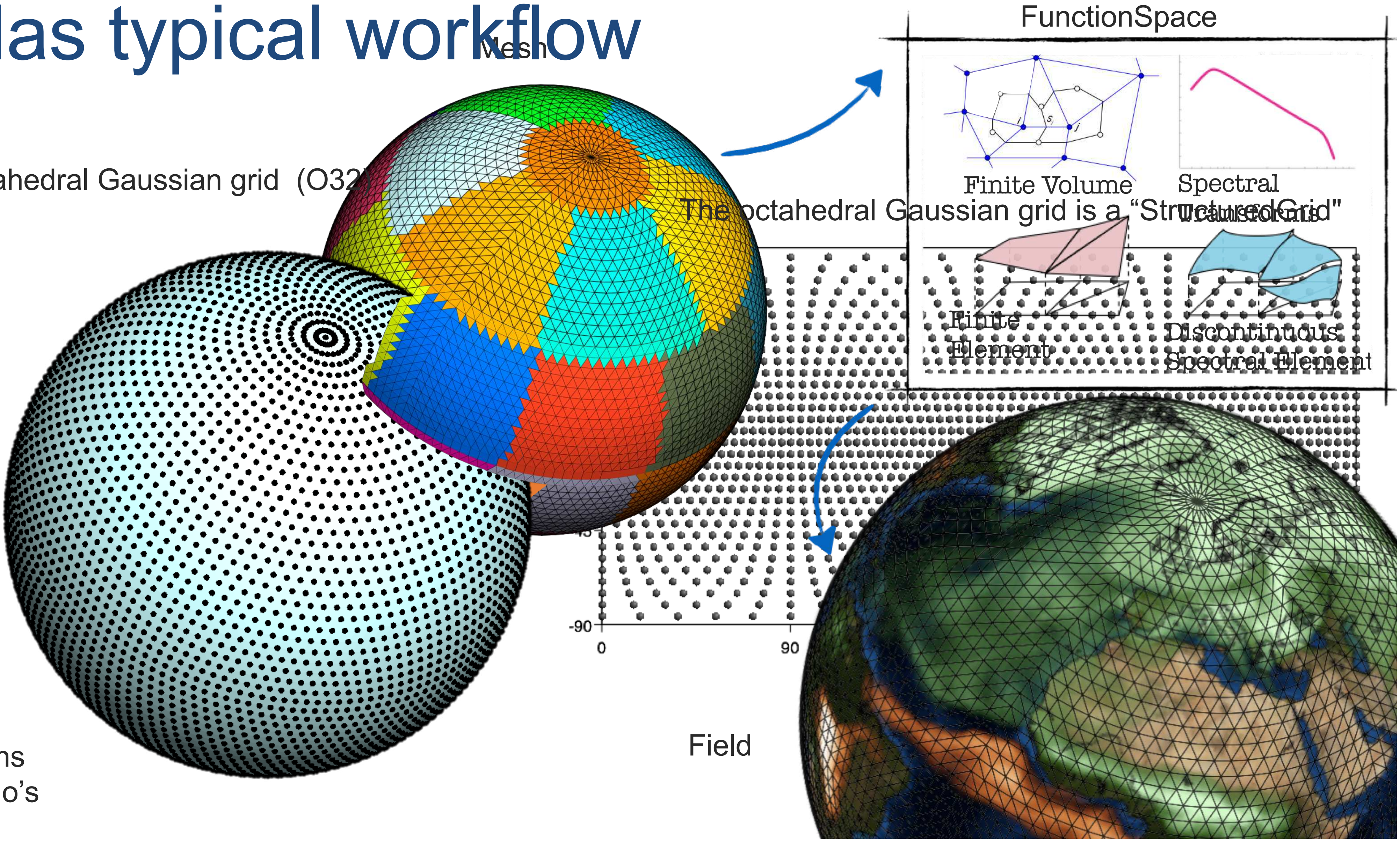
Atlas Library/Framework Change Alg
ECMWF
C++ with Fortran 2008 wrappers
Support for different grids structured and unstructured
Methods FD/FV/FEM

ESCAPE (ECMWF + lots of partners weather/climate and vendors)
Extract computational patterns (Dwarfs or mini-apps)
Explore optimisation space

Atlas typical workflow

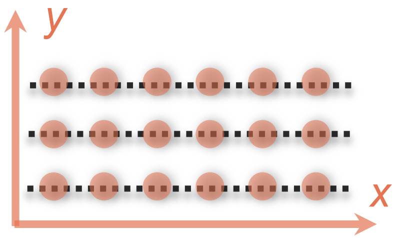
Octahedral Gaussian grid (O32)

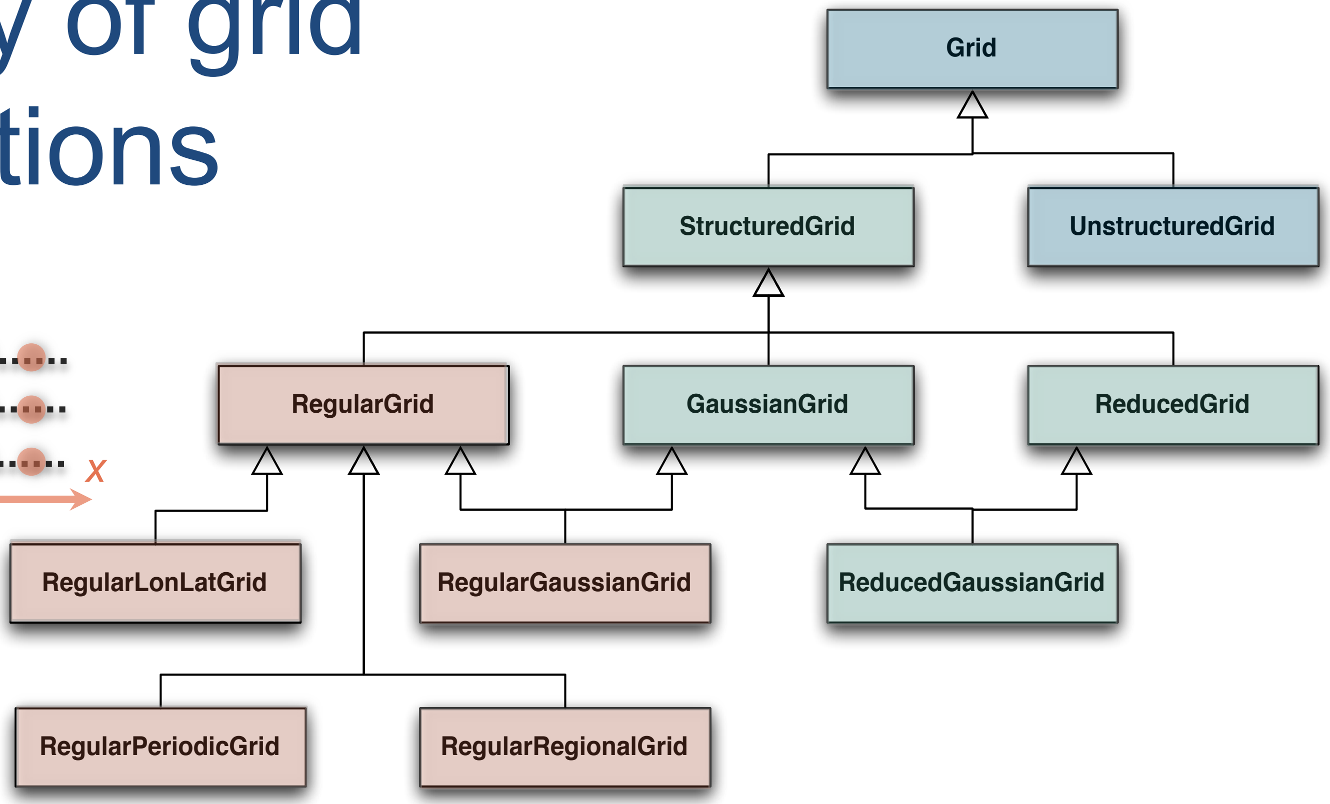
Grid

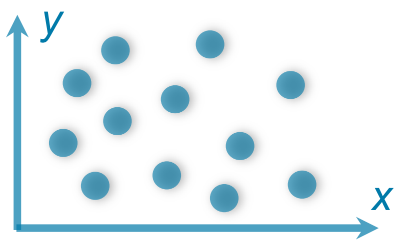


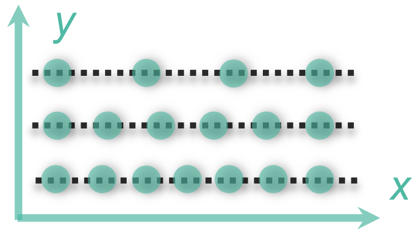
Partitions
with halo’s

A library of grid abstractions









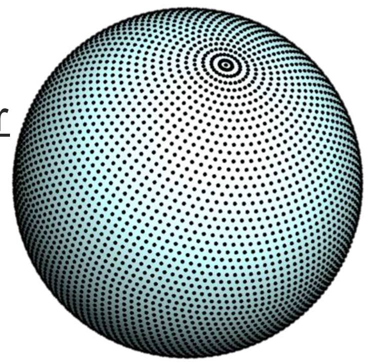
Example creation of operational octahedral reduced Gaussian grid using unique identifier

C++

```
atlas::Grid grid;
grid = atlas::Grid ( “O1280” )
```

Fortran

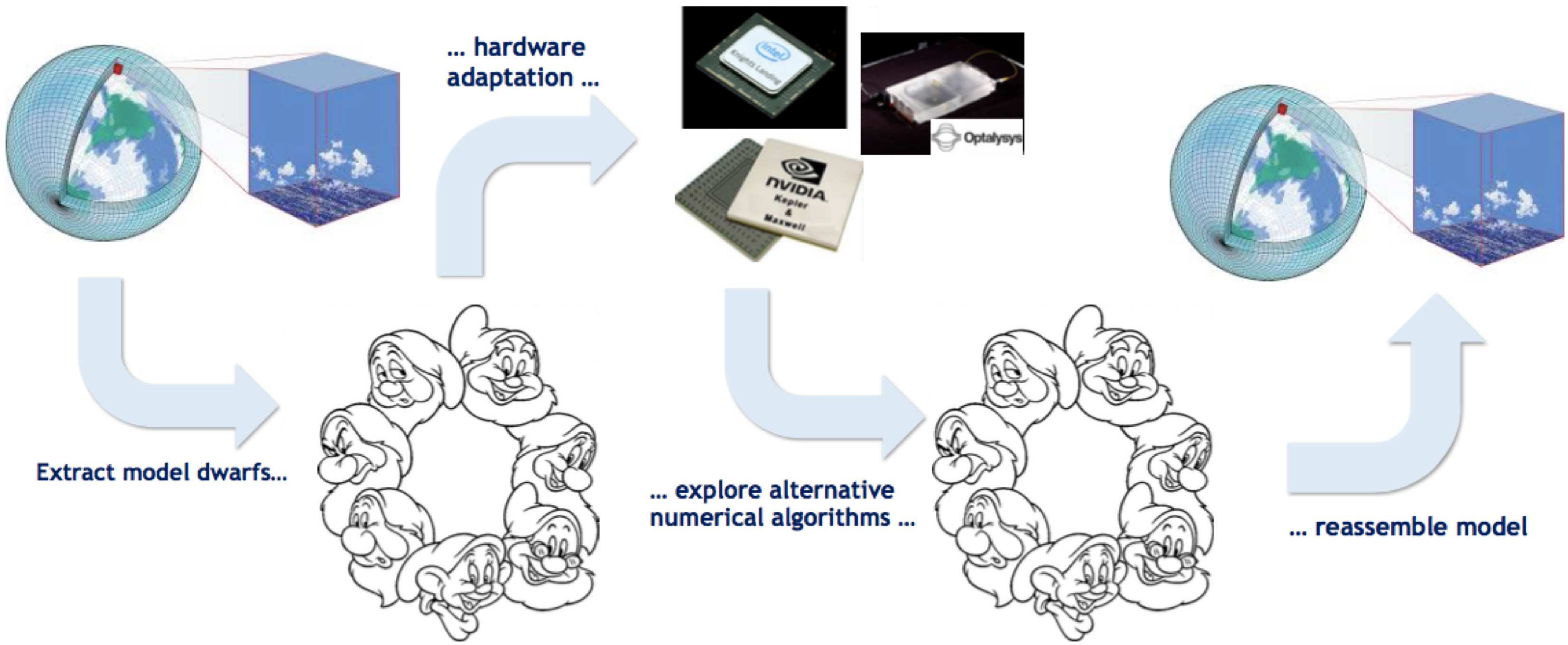
```
type(atlas_Grid) :: grid
grid = atlas_Grid( “O1280” )
```



Funded by the European Union

Idea behind ESCAPE

ESCAPE



Optimization of spectral transform dwarf

Funded by the European Union

ESCAPE

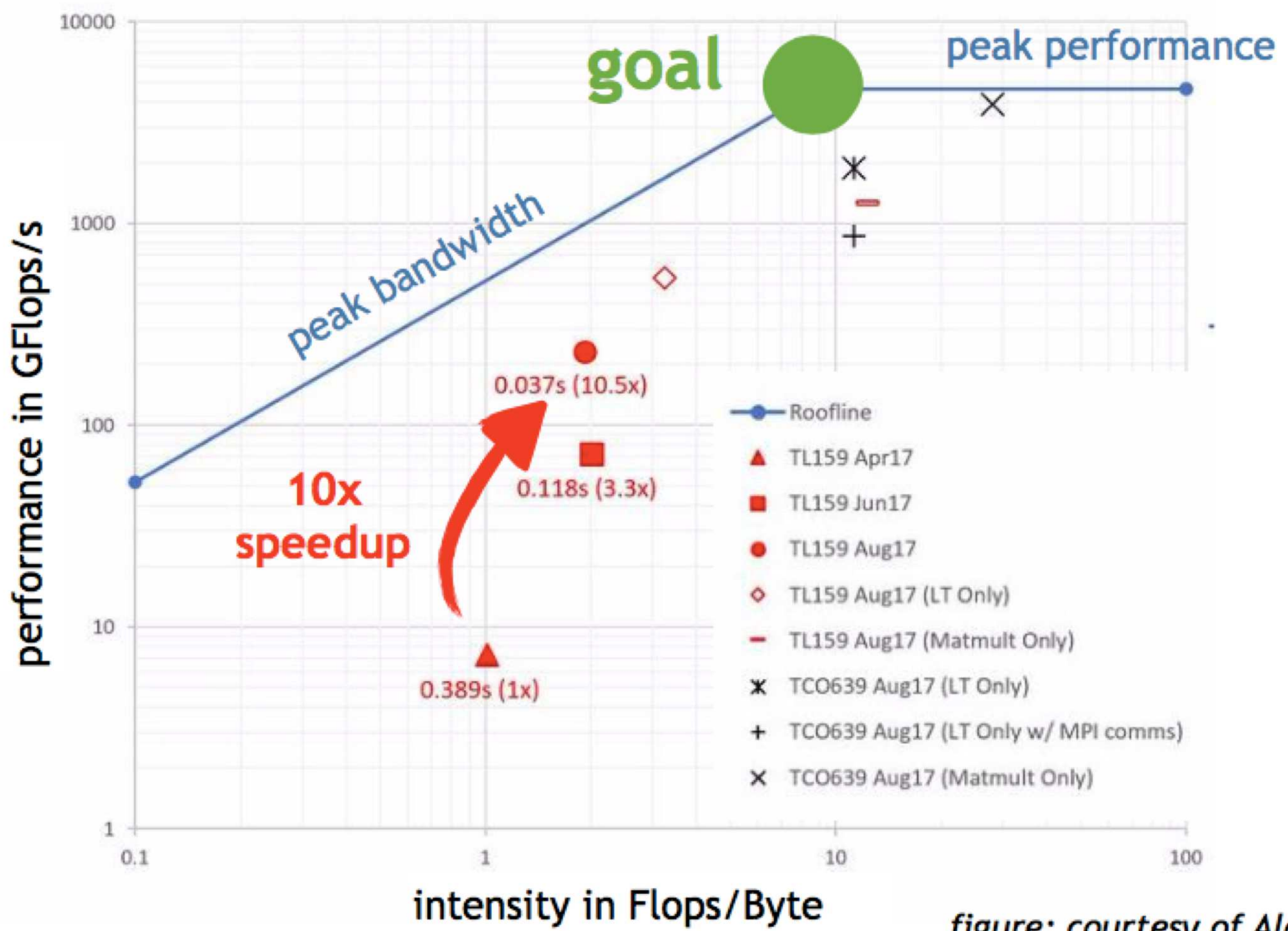
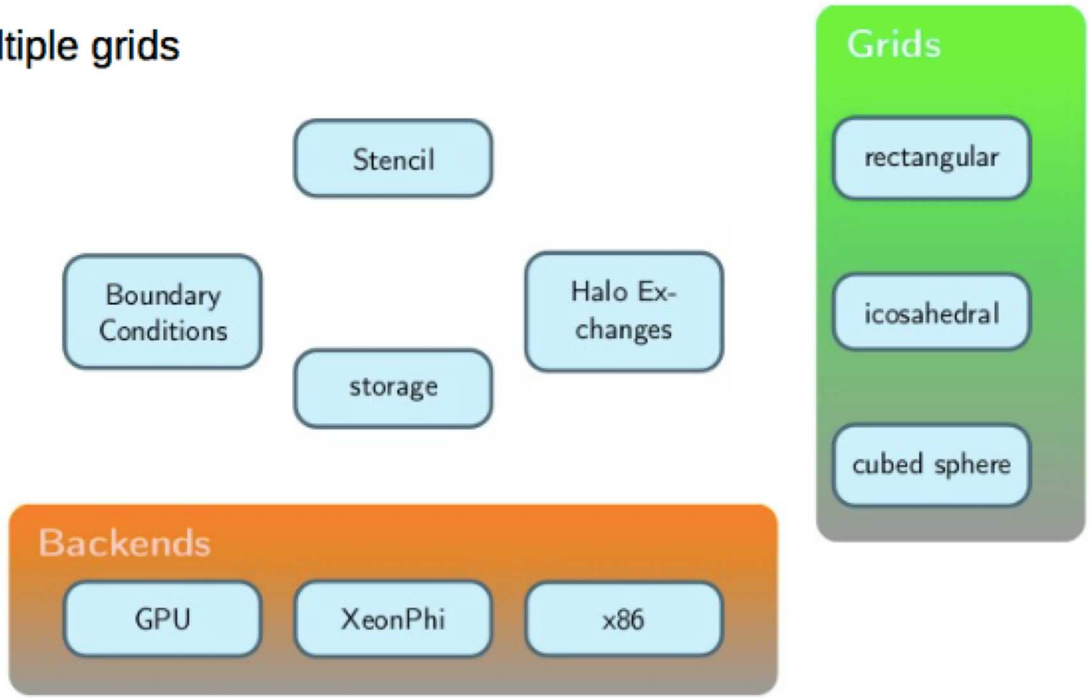


figure: courtesy of Alan Gray, Peter Messmer (NVIDIA)

GridTools

- Set of grid tools, including DSL for stencil codes, for solving PDEs on
  multiple grids

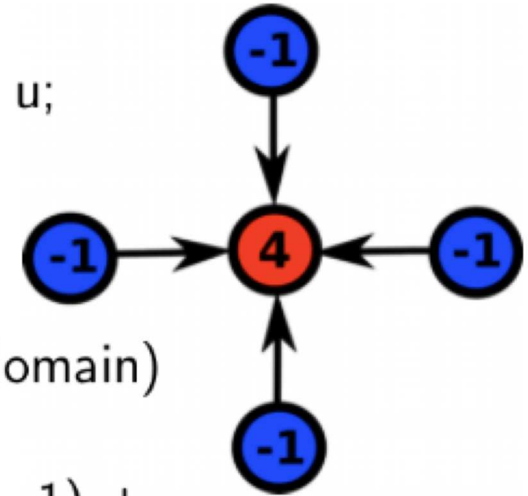


- Provides separation of concerns: Separates model and algorithm from hardware specific implementation and optimization
- Supports multiple hardware and grid backends.

Encoding Stencil Information in Types

```
struct Laplace
{
    typedef in_accessor<0, range<−1,1−1,1> > u;
    typedef out_accessor<1> lap;

    template<typename Evaluation>
    static void Do(Evaluation const& eval, full_domain)
    {
        eval(lap()) = eval(−4*u() + u(i+1) + u(i−1) +
            u(j+1) + u(j−1));
    }
};
```



Co-design: Extending Collaborations