CO405H

Computing in Space with OpenSPL

Topic 12: Numerics II

Oskar Mencer **Georgi Gaydadjiev**

Department of Computing
Imperial College London

http://www.doc.ic.ac.uk/~oskar/
http://www.doc.ic.ac.uk/~georgig/

CO405H course page: http://cc.doc.ic.ac.uk/openspl16/
WebIDE: http://openspl.doc.ic.ac.uk
OpenSPL consortium page: http://www.openspl.org

o.mencer@imperial.ac.uk g.gaydadjiev@imperial.ac.uk

Lecture Overview

- Recap from Numerics I
- Distribution of Variable Values
- Examples in Seismic Imaging and Weather Simulation
- Representation of Variables in Spatial Computing

Number Representation in OpenSPL

- OpenSPL supports floating point and fixed point/ integer arithmetic
  - Depends on the *type* of the SCSVar
- Can type inputs, outputs and constants
- Or can *cast* SCSVars from one type to another
- Types are Java objects, just like SCSVars,

```
// Create an input of type t
SCSVar io.input(String name, SCSType t);

// Create an SCSVar of type t with constant value
SCSVar constant.var(SCSType t, double value);

// Cast SCSVar y to type t
SCSVar x = y.cast(SCSType t);
```

OpenSPL Floating Point - scsFloat

- Floating point numbers with base 2, flexible exponent and mantissa
- Compatible with IEEE floating point **except** does not support denormal numbers
  - In spatial computers choose to use a larger exponent

```
SCSType t = scsFloat(int exponent_bits, int mantissa_bits);
```

Including the sign bit

- Examples:

	Exponent bits	Mantissa bits
IEEE single precision	8	24
IEEE double precision	11	53
DFE optimized low precision	7	17

Why scsFloat(7,17)…?

OpenSPL Fixed Point – scsFixOffset

- Fixed point numbers
- Flexible integer and fraction bits
- Flexible sign mode
  - SignMode.UNSIGNED or SignMode.TWOSCOMPLEMENT

```
SCSType t = scsFixOffset(int num_bits, int offset, SignMode sm);
```

- Common cases have useful aliases

	Integer bits	Fraction bits	Sign mode
scsInt(N)	N	0	TWOSCOMPLEMENT
scsUInt(N)	N	0	UNSIGNED
scsBool()	1	0	UNSIGNED

Rounding

- When we remove bits from the RHS of a number we may want to perform *rounding*.
  - Casting / type conversion
  - Inside arithmetic operations
- Different possibilities
  - TRUNCATE: throw away unwanted bits
  - TONEAR: if >=0.5, round up (add 1)
  - TONEAREVEN: if >0.5 round up, if <0.5 round down, if =0.5 then round to the nearest even number
- Lots of less common alternatives:
  - Towards zero, towards positive infinity, towards negative infinity, random uniform, uniform with distribution….
- Very important in iterative calculations – affects convergence behaviour

Rounding in MaxCompiler

- Floating point arithmetic uses TONEAREVEN
- Fixed point rounding is flexible, controlled by the *RoundingMode*
  - TRUNCATE, TONEAR and TONEAREVEN are in-built

```
DFEVar z;
...
optimization.pushRoundingMode(RoundingMode.TRUNCATE);

z = z.cast(smaller_type);

optimization.popRoundingMode();
```

Numerics II: Variables, Values and Distribution

STEP 1: For all variables in the application which are moving to the DFE, record all values of each particular variable.

STEP 2: look at histogram, distribution, average, min, max, variance, etc.

STEP 3: pick representation to match range and distribution of values for the variable.

Ex 1: Number Representations in Geoscience

Case Study: Complex Exponential in Downward Continued Based Migration

$$U(w,k_s,k_g,z+\Delta z)=\exp[-iwv(\sqrt{1-\frac{vk_g}{w}}+\sqrt{1-\frac{vk_s}{w}})]\cdot U(w,k_s,k_g,z)$$

Double Square Root (DSR) condition (the complex exponential step) with frequency w
- a small table holds the vk/w values
- next an approximate value of the square root is looked up

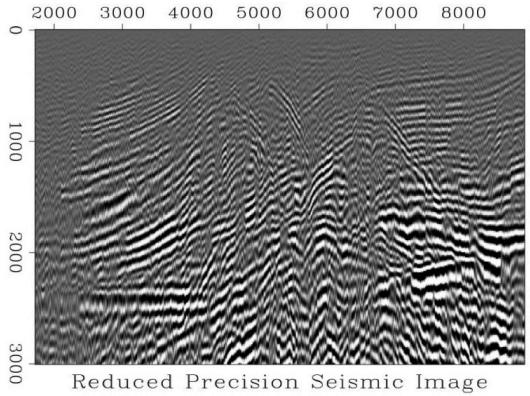
The design consists of three parts:
- *square root calculation*
- *sine/cosine evaluation*, and
- *complex multiplication*.

The above three parts have quite different range of variables → different bit-widths in each part: SQRT (square root), SINE (sine/cosine evaluation) and WMUL (wave-field complex multiplication) bit-width.

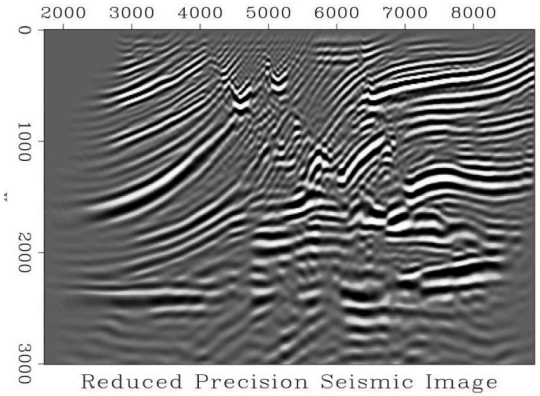
[H.Fu, W. Osborne, R. G. Clapp, O. Pell, Accelerating Seismic Computations [], 70th EAGE Conference, Italy, 2008]

Fixed-point bit-width exploration

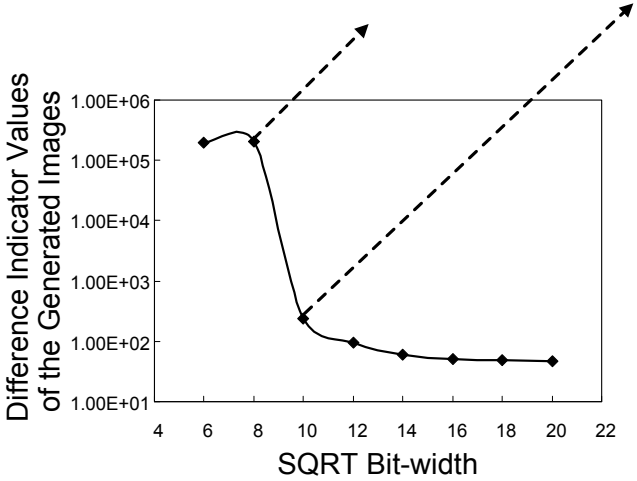
8-bit fixed-point



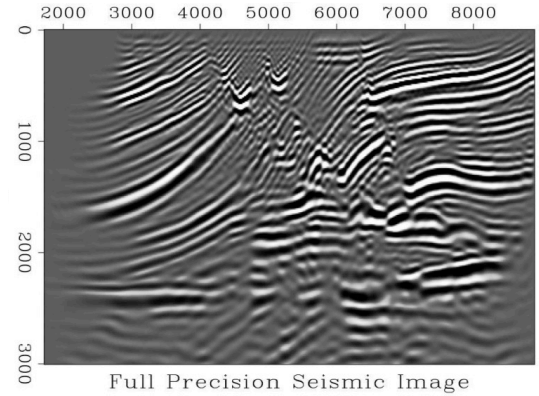
10-bit fixed-point



Different parts are explored *separately*, i.e., when we investigate one part, we keep the bit-widths in other parts a constant high value



'true' image: single-precision floating-point



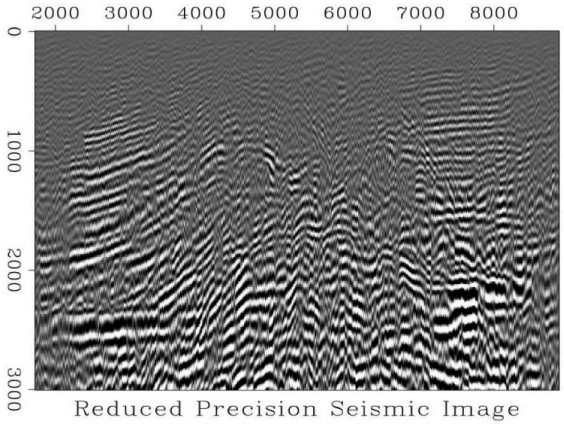
Similarly, we observe a significant drop of the error when the SQRT bit-width increases from 8 to 10

Similar precision thresholds observed in both synthetic and field results. This behavior enables an automatic tool to determine the minimum precision that still keeps the result *good enough*

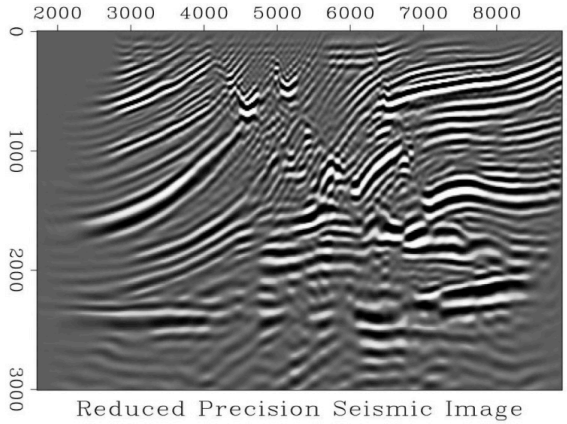
[H.Fu, W. Osborne, R. G. Clapp, O. Pell, Accelerating Seismic Computations [], 70th EAGE Conference, Italy, 2008]

Floating-point bit-width exploration

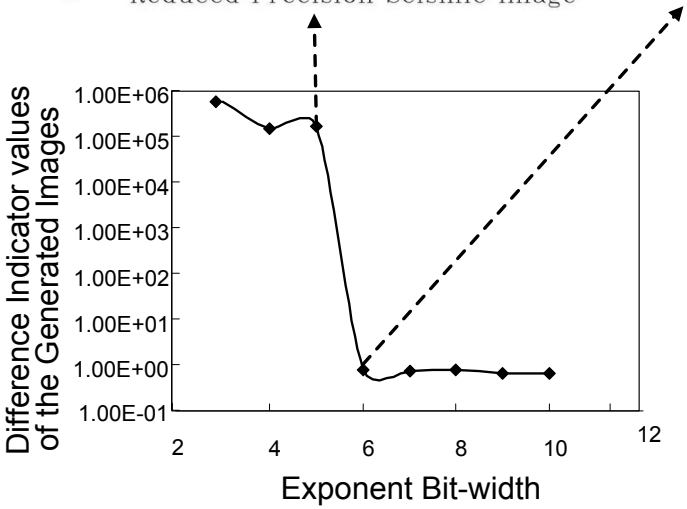
floating-point: 5-bit exponent



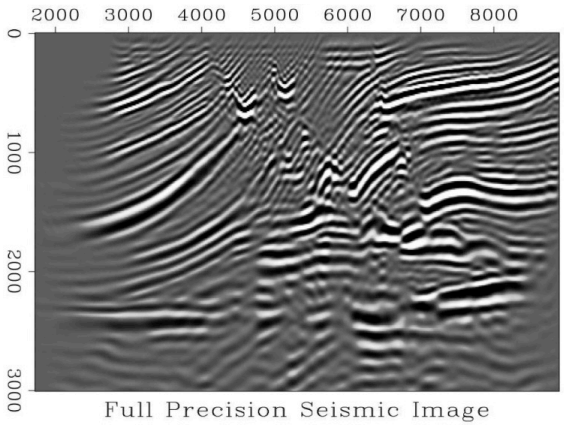
floating-point: 6-bit exponent



We use the Marmousi synthetic data set as the test data, and explore different combinations of exponent and mantissa bit-width



'true' image: single-precision floating-point



A precision threshold at exponent width of 6 bits:

- The error drops significantly when we increase the exponent width from 5 bits to 6 bits
- The image also turns from nearly random noise at 5 bits, to almost identical to the 32-bit image at 6 bits

[H.Fu, W. Osborne, R. G. Clapp, O. Pell, Accelerating Seismic Computations [], 70th EAGE Conference, Italy, 2008]

Acceleration results

- Based on exploration results, we use 12, 16, and 16-bit fixed-point numbers for the SQRT, SINE and WMUL parts

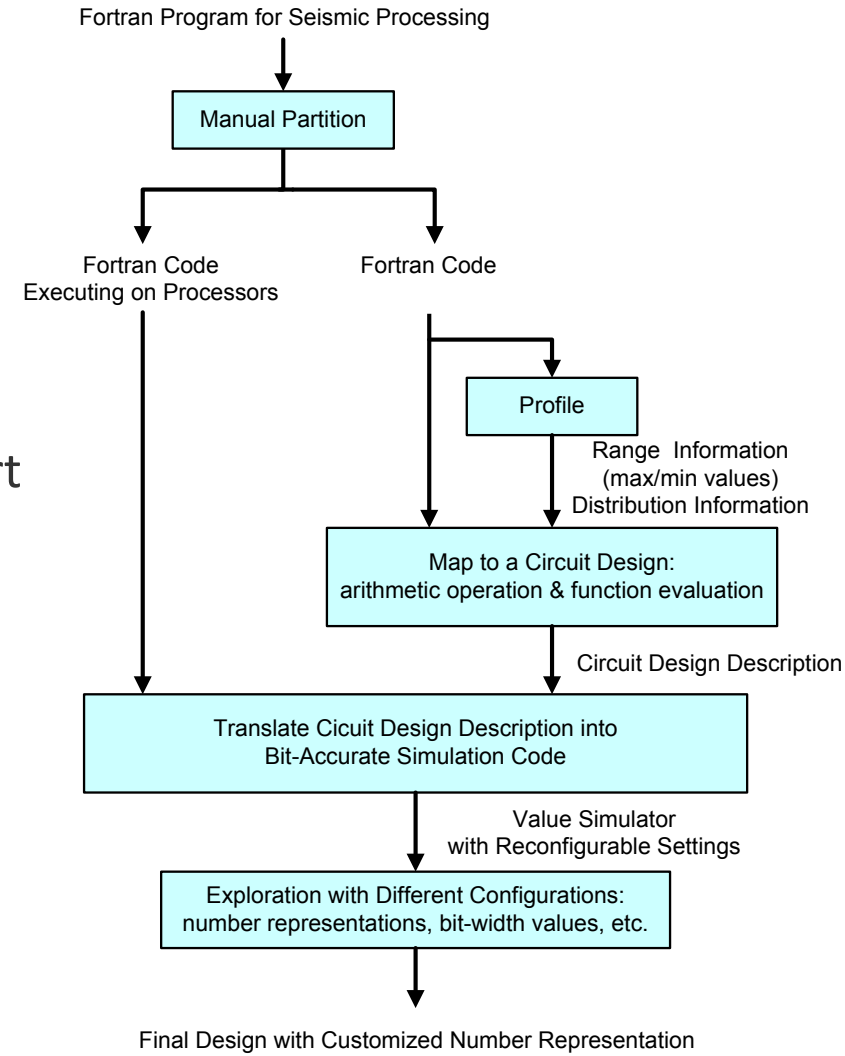
size of data set	processing time		speedup
	software	DFE	
43056	5.32 ms	0.84 ms	6.3
216504	26.1 ms	3.77 ms	6.9

- Implemented on Maxeler Vectis DFE with 192 MULTs
- The design consumes 28% of logic elements, 15% of FMEM, 10% of MULTs. Mapping 6 cores into the card (with additional effort) would provide up to 40x of node-to-node acceleration.

[H.Fu, W. Osborne, R. G. Clapp, O. Pell, Accelerating Seismic Computations [], 70th EAGE Conference, Italy, 2008]

Tool for adapting number presentations

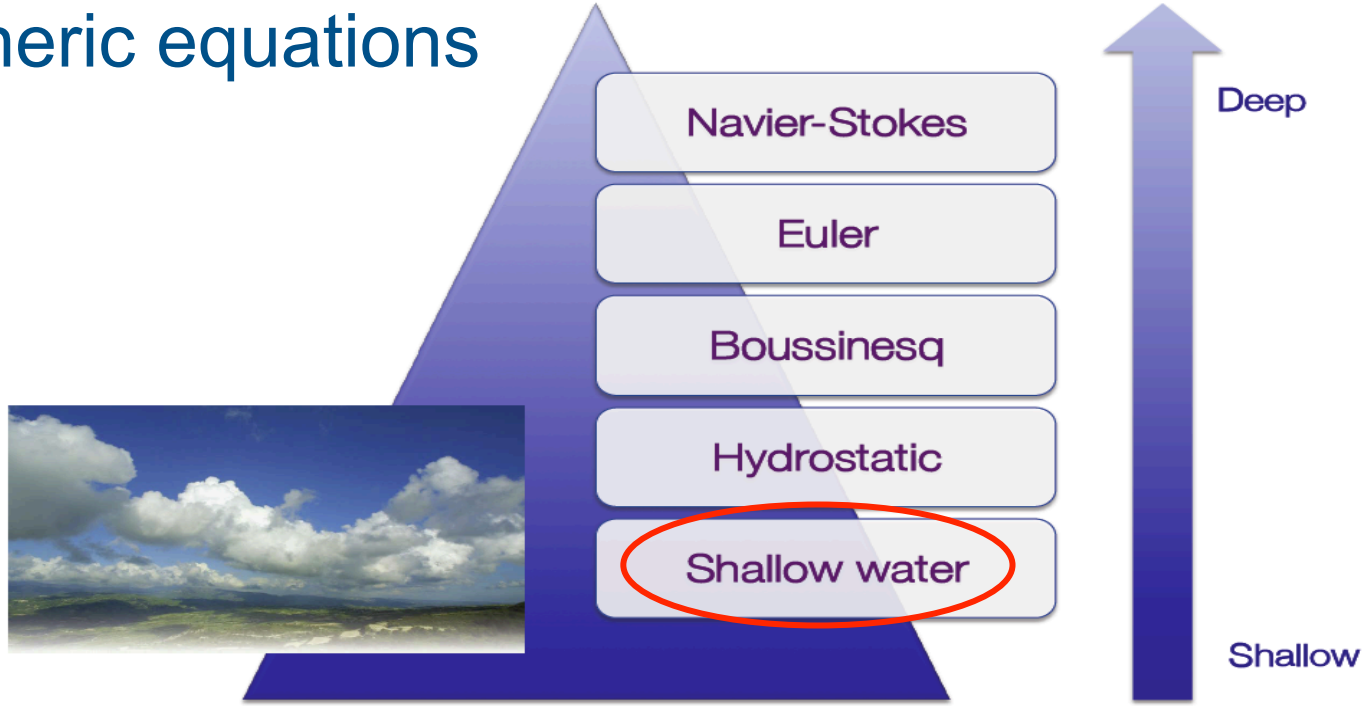
- Partition the program into software and hardware part
- Profile the variables in the hardware part
- Map the dataflow part to the DFE
- Explore the design space with different precisions



[H.Fu, W. Osborne, R. G. Clapp, O. Pell, Accelerating Seismic Computations [], 70th EAGE Conference, Italy, 2008]

Example 2: Global Weather Simulation

- Atmospheric equations



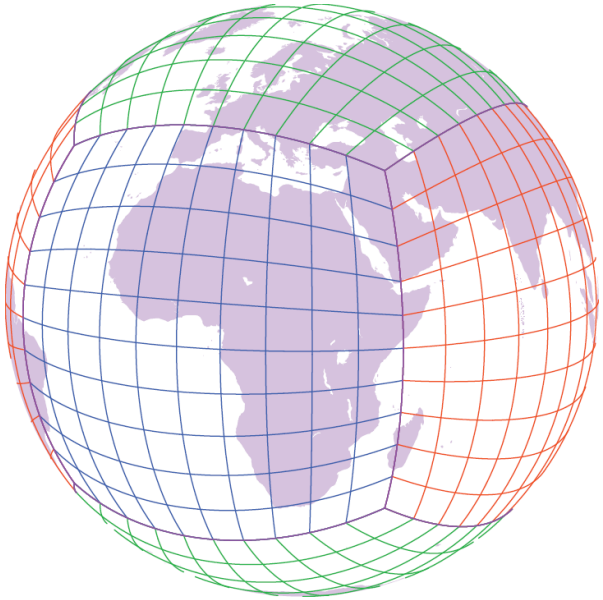
- Equations: Shallow Water Equations (SWEs)

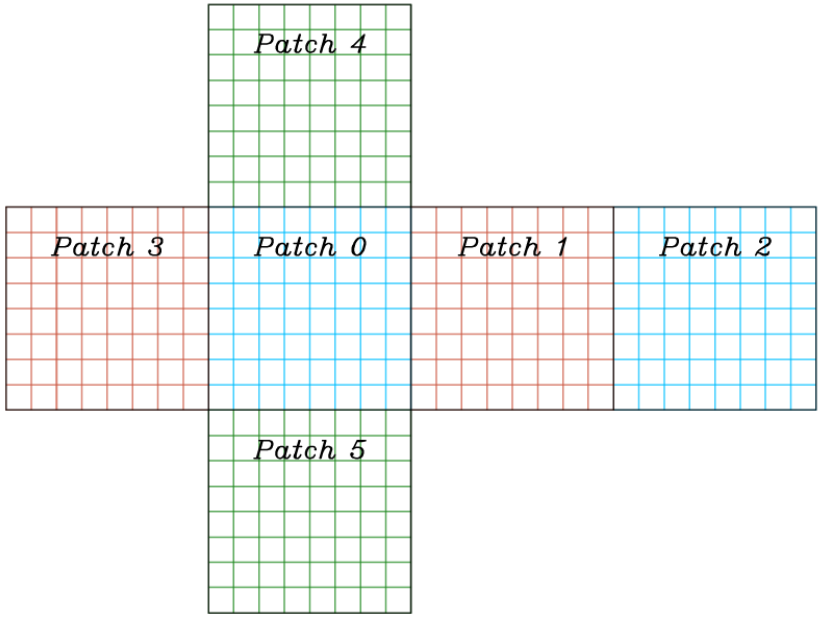
$$\partial Q/\partial t + 1/\Lambda\ \partial(\Lambda F^1)/\partial x^1 + 1/\Lambda\ \partial(\Lambda F^1)/\partial x^2 + S = 0$$

[L. Gan, H. Fu, W. Luk, C. Yang, W. Xue, X. Huang, Y. Zhang, and G. Yang, Accelerating solvers for global atmospheric equations through mixed-precision data flow engine, FPL2013]

Global Weather Simulation, mesh

- Cubed-sphere mesh
  - Mapping inscribed cube to the surface of the earth
- Computational domain
  - Six patches covered with rectangular meshes

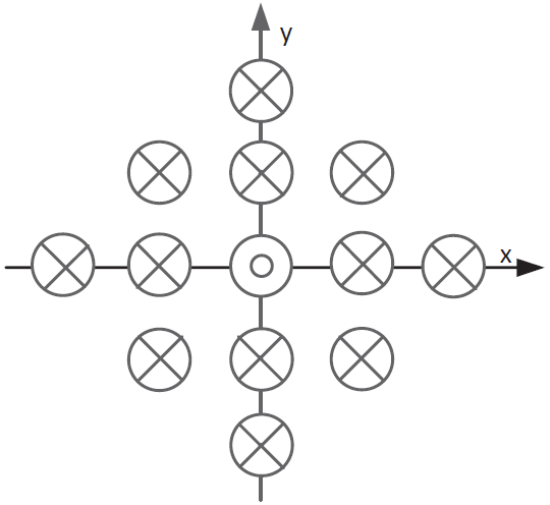




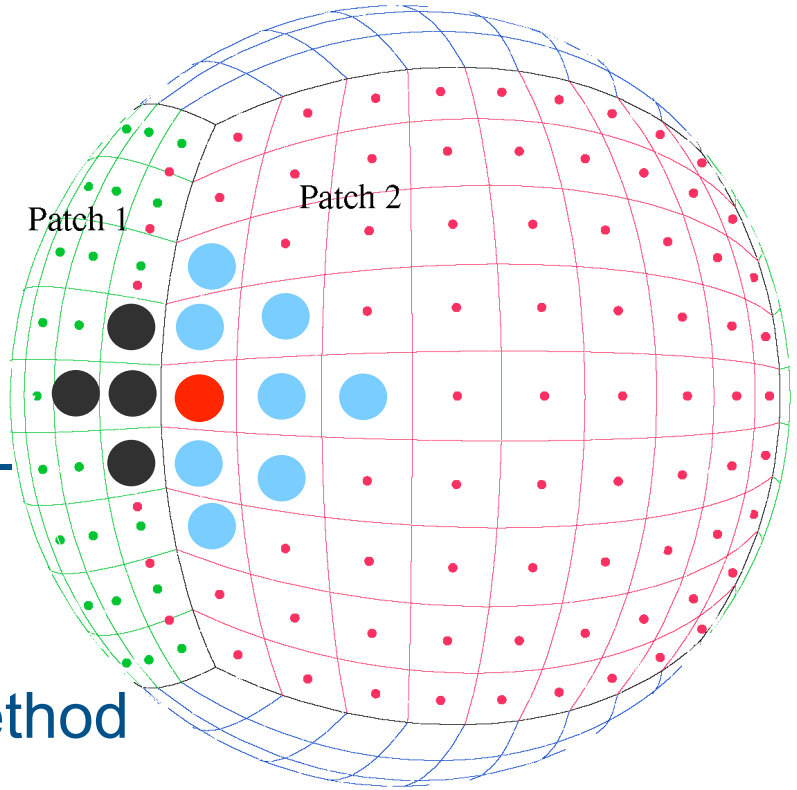
[L. Gan, H. Fu, W. Luk, C. Yang, W. Xue, X. Huang, Y. Zhang, and G. Yang, Accelerating solvers for global atmospheric equations through mixed-precision data flow engine, FPL2013]

Global Weather Simulation (cont)

- 13-point stencil



- Interp. Across patches
  - 1-d linear interpolation



- Spatially discretized with a cell-centred finite volume method
- Integrated with a second-order accurate TVD Runge-Kutta method

[L. Gan, H. Fu, W. Luk, C. Yang, W. Xue, X. Huang, Y. Zhang, and G. Yang, Accelerating solvers for global atmospheric equations through mixed-precision data flow engine, FPL2013]

Hybrid DFE-CPU implementation

For each stencil cycle

DFE side:

①Inner-part stencil

CPU side:

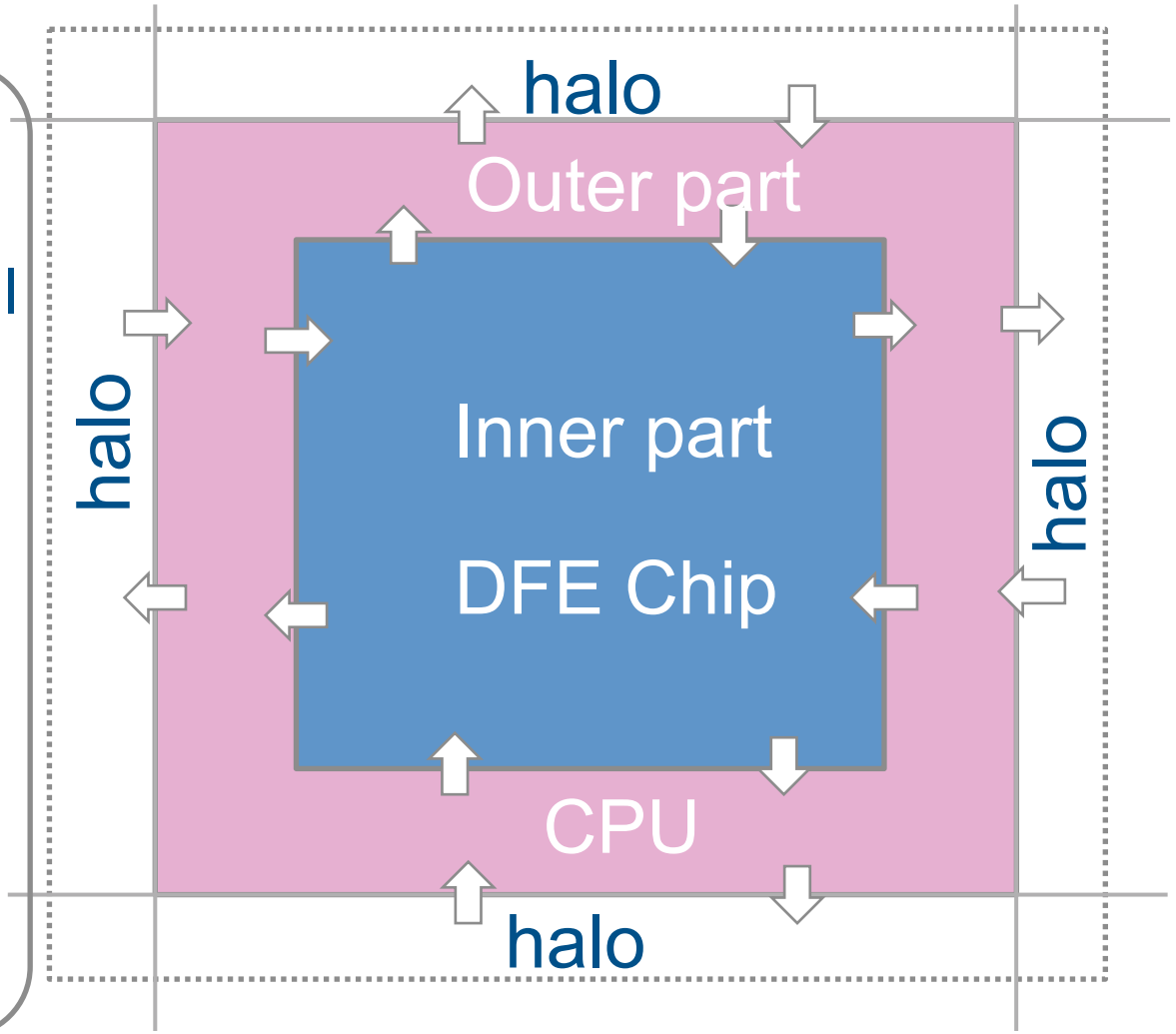
①Update halos

②Interpolate if necessary

③Outer-part stencils

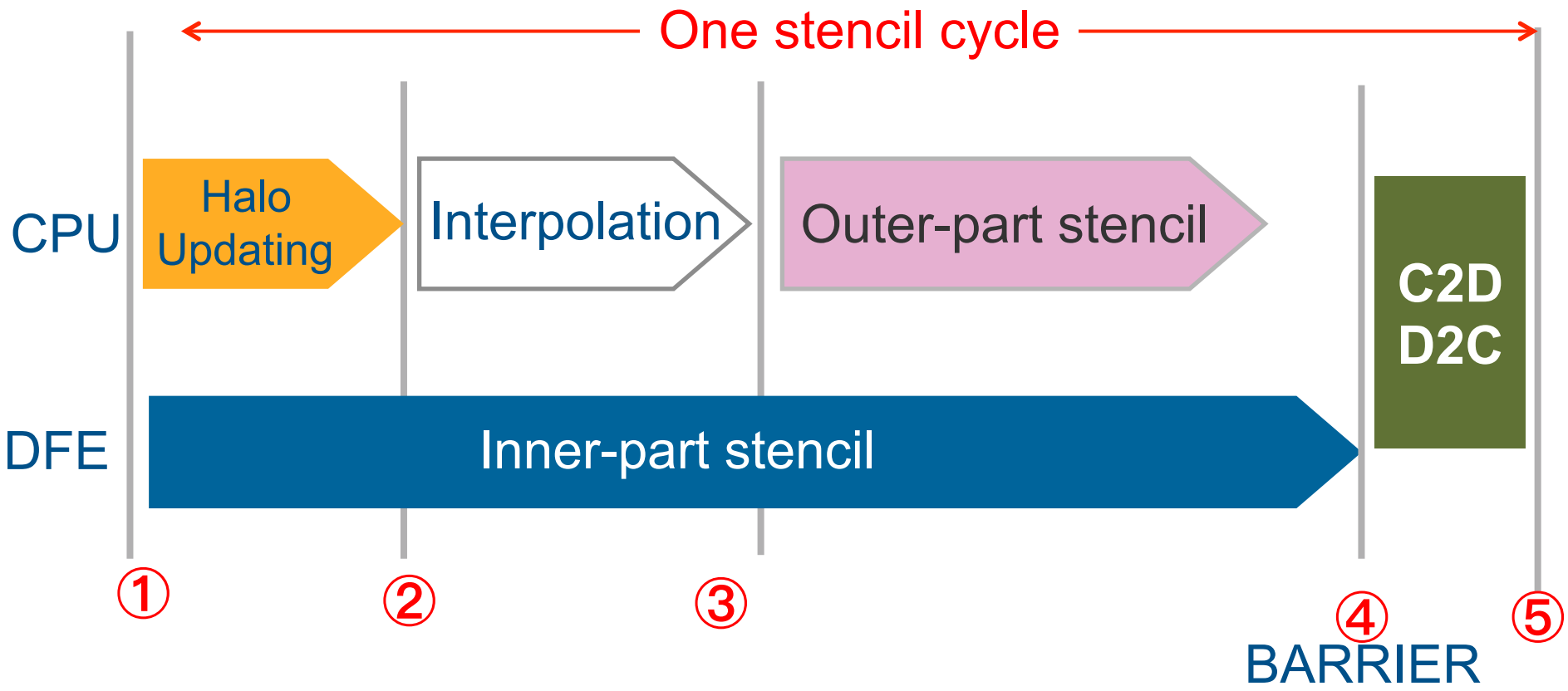
BARRIER:

CPU-DFE exchange



[L. Gan, H. Fu, W. Luk, C. Yang, W. Xue, X. Huang, Y. Zhang, and G. Yang, Accelerating solvers for global atmospheric equations through mixed-precision data flow engine, FPL2013]

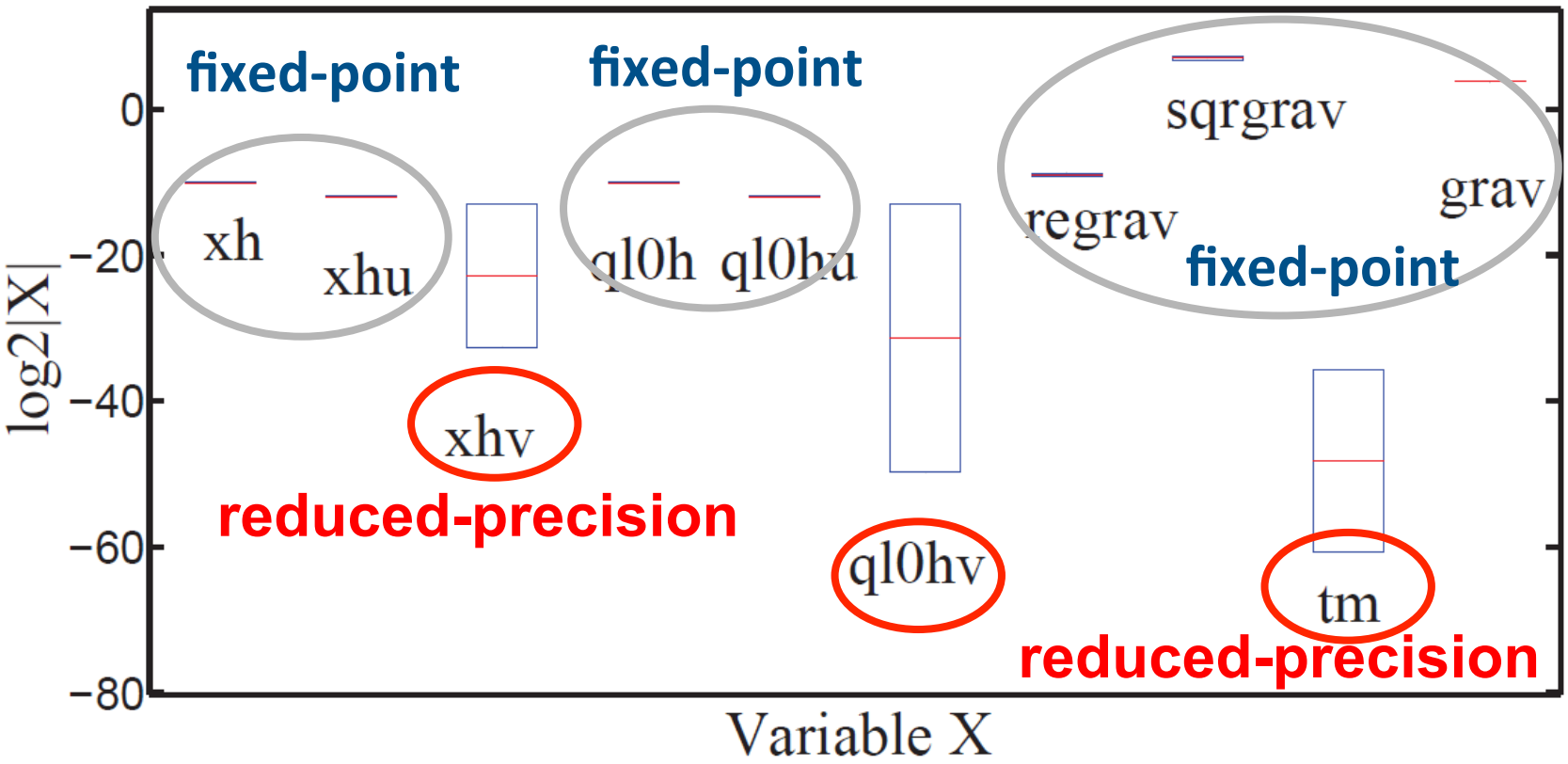
DFE-CPU Workflow



[L. Gan, H. Fu, W. Luk, C. Yang, W. Xue, X. Huang, Y. Zhang, and G. Yang, Accelerating solvers for global atmospheric equations through mixed-precision data flow engine, FPL2013]

Always double-precision needed?

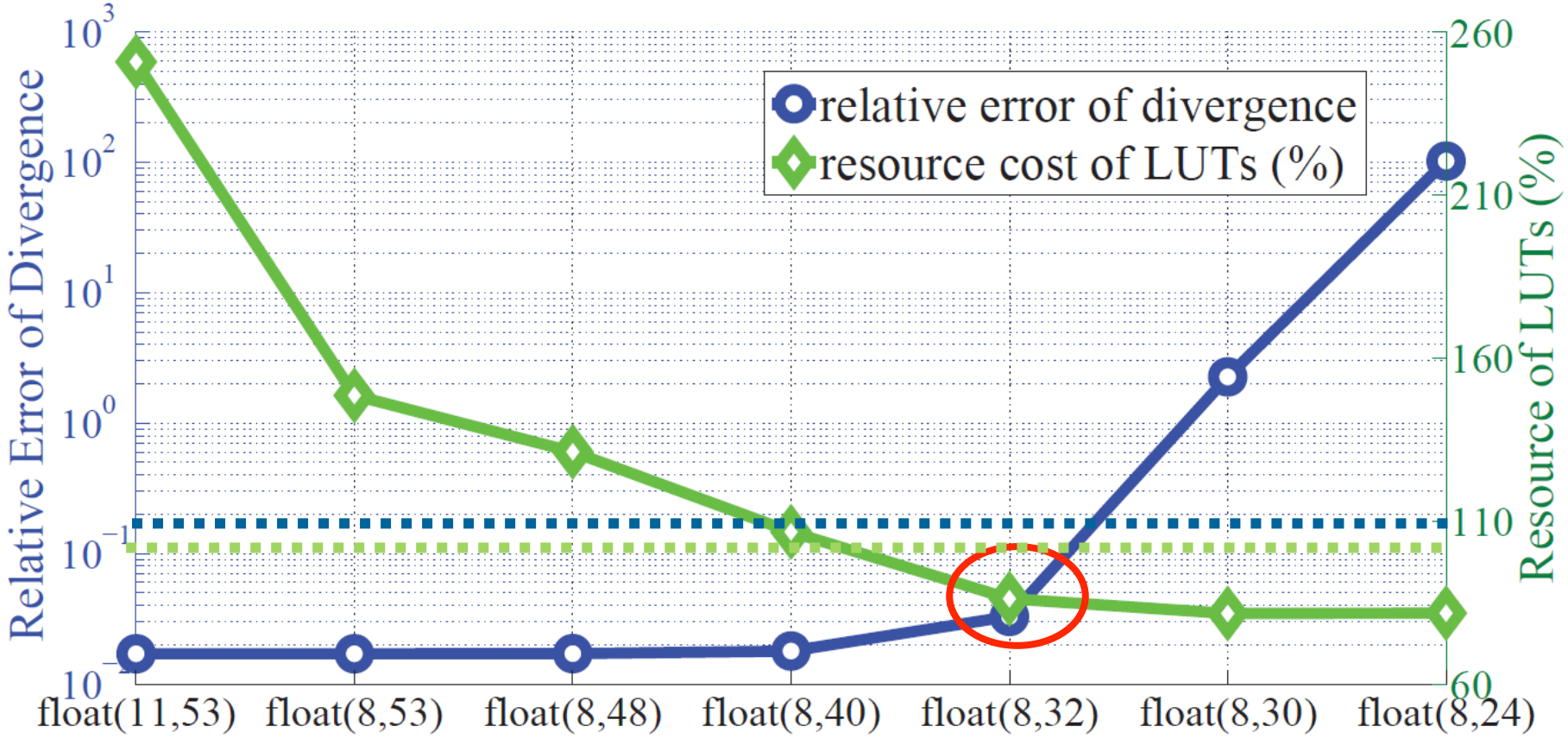
- Range analysis to track the absolute values of all variables



[L. Gan, H. Fu, W. Luk, C. Yang, W. Xue, X. Huang, Y. Zhang, and G. Yang, Accelerating solvers for global atmospheric equations through mixed-precision data flow engine, FPL2013]

What about error vs area tradeoffs

- Bit accurate simulations for different bit-width configurations.



[L. Gan, H. Fu, W. Luk, C. Yang, W. Xue, X. Huang, Y. Zhang, and G. Yang, Accelerating solvers for global atmospheric equations through mixed-precision data flow engine, FPL2013]

Accuracy validation

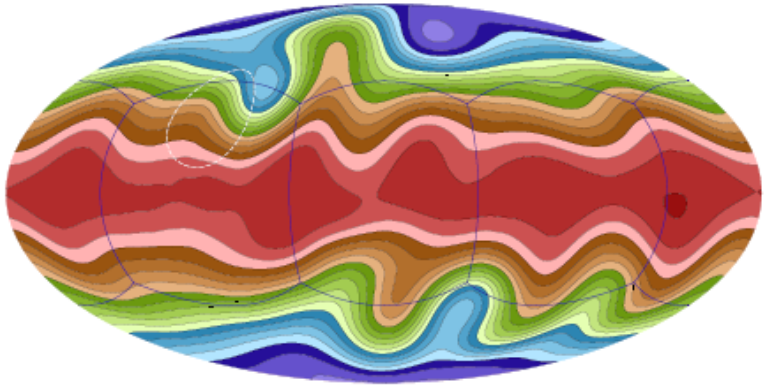


Figure 15. Surface level distribution of the atmosphere at day 15 in the isolated mountain test. Results are obtained on a 10,240 10,240 6 cubed-sphere mesh using 1,536 nodes of the Tianhe-1A. The conical mountain is outlined by the dotted circle in the figure.

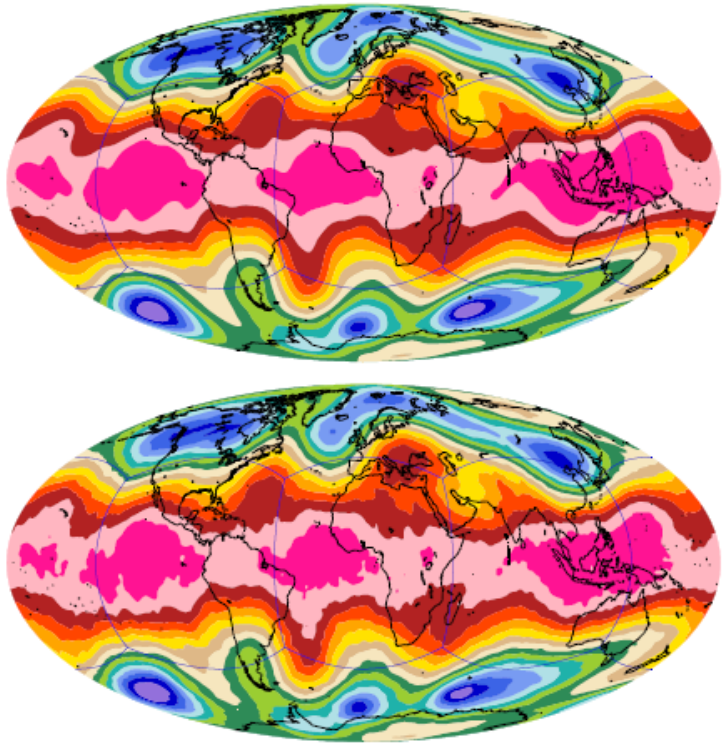


Figure 16. Surface level distribution of the atmosphere at day 15 in the real-topography test. We compare results at a 40-km resolution (upper panel) and a 1-km resolution (lower panel).

[Chao Yang, Wei Xue, Haohuan Fu, Lin Gan, et al. 'A Peta-scalable [] Algorithm for Global Atmospheric Simulations', PPoPP'2013]

And there is also performance gain

Platform	Performance	Speedup
6-core CPU	4.66K	1
Tianhe-1A node	110.38K	23x
MaxWorkstation	468.1K	**100x**
1U MaxNode	1.54M	**330x**

14x

Meshsize: 1024×1024×6
1U MPC-X MaxNode speedup over Tianhe node: 14 times

[Chao Yang, Wei Xue, Haohuan Fu, Lin Gan, et al. 'A Peta-scalable [] Algorithm for Global Atmospheric Simulations', PPoPP'2013]

Weather model -- power efficiency

Platform	Efficiency	Power Advantage
6-core CPU	20.71	1
Tianhe-1A node	306.6	14.8x
MaxWorkstation	2.52K	**121.6x**
1U MaxNode	3K	**144.9x**

9 x

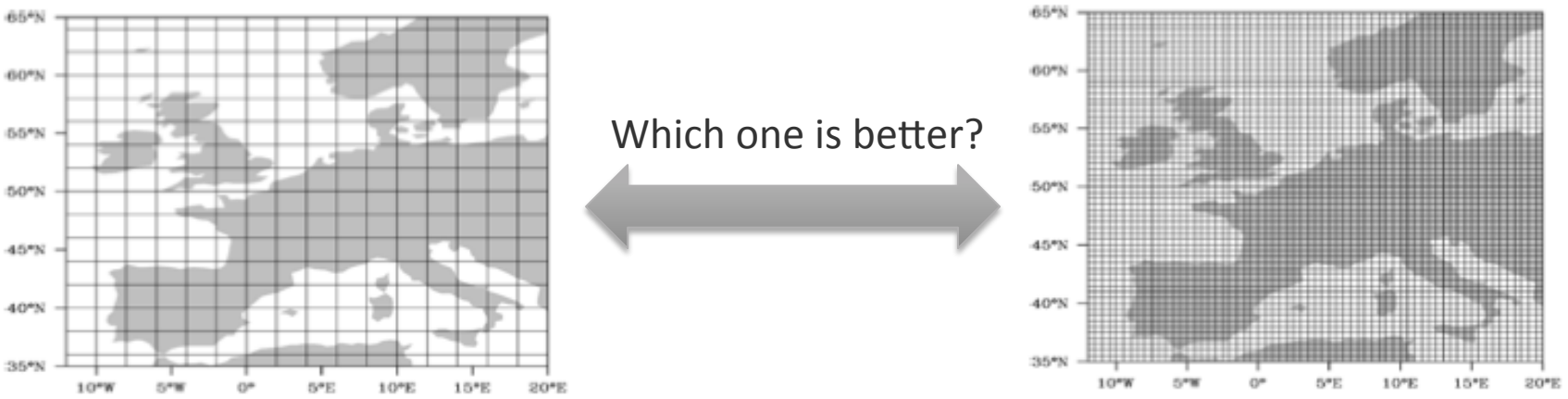
Meshsize: 1024×1024×6

1U MaxNode is 9 times more power efficient

Imperial College London

ISCAS

Example 3: Weather / climate models on DFEs

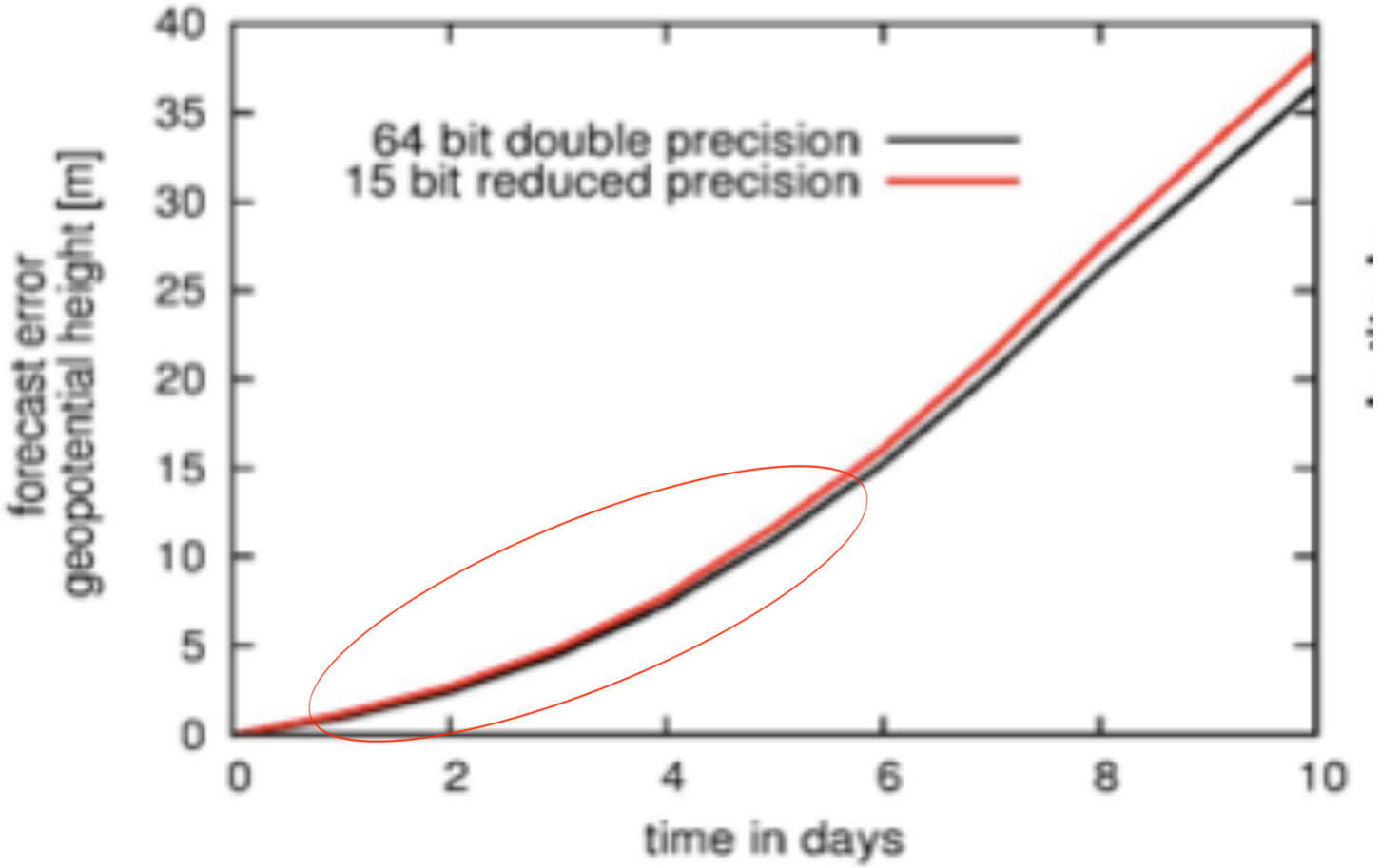


Finer grid and higher precision are obviously preferred but the computational requirements will increase ➔ Power usage → $$

What about using reduced precision? (15 bits instead of 64 double precision FP)

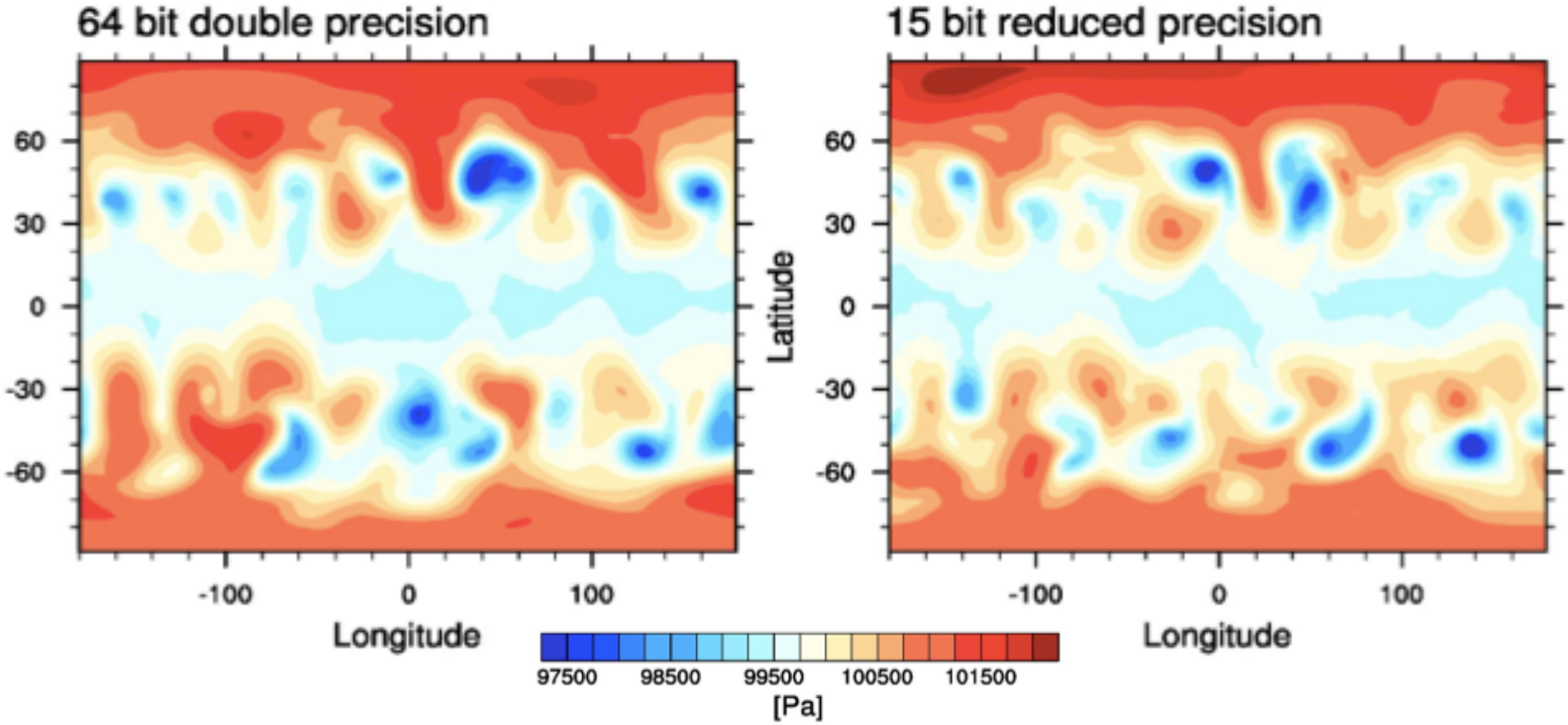
[P. Düben, T. Palmer, Oxford, OCCAM, "Weather and climate models on dataflow engines," CeBIT, 2013]

Weather models precision comparison



[P. Düben, T. Palmer, Oxford, OCCAM, "Weather and climate models on dataflow engines," CeBIT, 2013]

What about 15 days of simulation?



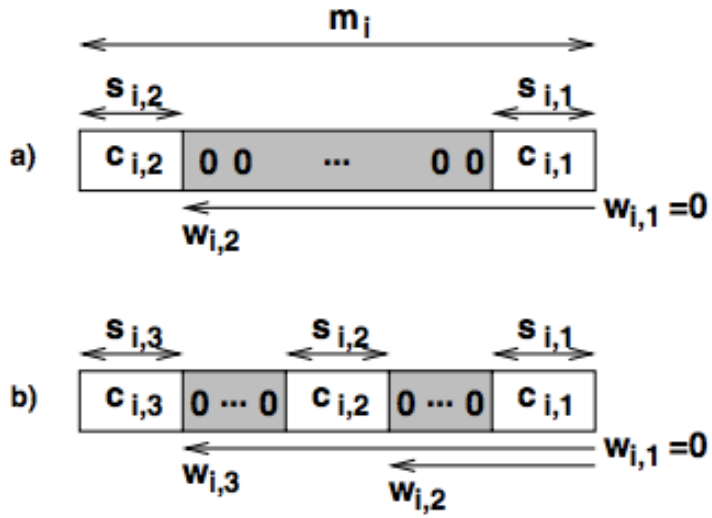
Surface pressure after **15 days** of simulation for the double precision and the reduced precision simulations (quality of the simulation hardly reduced)

[P. Düben, T. Palmer, Oxford, OCCAM, "Weather and climate models on dataflow engines," CeBIT, 2013]

More Advanced Number Representations

- Integer: unsigned, signed, one's/ two's complement
- Floating Point: single, double precision (also custom)
- Fixed point: (any custom radices / bitwidths)
- Logarithmic number representation
- Redundant number systems: use more bits
  - Signed-digit representation
  - Residue number systems (modulo arithmetic)
  - Decimal: decimal floating point, binary coded decimal
- Decimal: decimal floating point, binary coded decimal
- *(any more exotic application specific representation)*
- Static and dynamic adaptations
- OpenSPL helps to express these

Advanced Optimisation for Number Representation Minimise ‘1’s in Polynomial Coeffcients



$$p = \frac{32799}{32768} - \frac{609}{32768}x - \frac{14881}{32768}x^2.$$

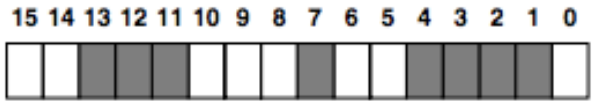


Fig. 2. Target format for cos function.

$$p = -\frac{75}{32768} + \frac{34538}{32768}x - \frac{6169}{32768}x^2.$$

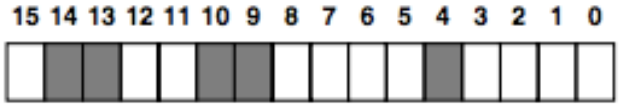


Fig. 3. Target format for sin function.

$$p = \frac{32793}{32768} + \frac{31836}{32768}x + \frac{21146}{32768}x^2.$$

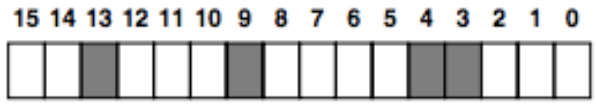


Fig. 4. Target format for exp function.

Nicolas Brisebarre, Jean-Michel Muller and Arnaud Tisserand
Sparse Coefficient Polynomial Approximations for Hardware Implementation,
Asilomar Conference, 2004.

Conclusions

- Understand your data requirements
- Know what fixed or floating point you need
- Mind rounding
- Different presentations have their costs in space!

Practice Exercises

1. Write a MaxCompiler kernel that takes one `dfeFloat(8,24)` input stream and adds it to a `dfeFloat(11, 53)` input stream to produce a `dfeFloat(11, 53)` result.

2. What will be the result of trying to represent $X=2^{32}$ and $Y=2^{-2}$ in each of the following number types:

```
dfeFixOffset(32, 0, SignMode.UNSIGNED)
dfeFixOffset(32, 0, SignMode.TWOSCOMPLEMENT)
dfeFixOffset(28, 4, SignMode.UNSIGNED)
dfeFixOffset(32, 4, SignMode.UNSIGNED)
dfeFloat(11, 53)
dfeFloat(8, 24)
dfeFloat(8, 32)
dfeFloat(8, 33)
```

3. Construct a test to show the difference between rounding modes on a multiplication operation of two `dfeFixOffset(4, 4, SignMode.TWOSCOMPLEMENT)` numbers. Vary the number of fraction bits – what is the impact on the bias difference between `TONEAR` and `TONEAREVEN` and why?