6TH INTERNATIONAL EULAG USERS WORKSHOP

MAY 29, 2018, 14:00-14:45

Exploiting Mixed Precision Arithmetic in MPDATA on GPU

KRZYSZTOF ROJEK, ROMAN WYRZYKOWSKI

CZESTOCHOWA UNIVERSITY OF TECHNOLOGY, POLAND

Focus areas and goals

- **Area**: Adaptation of a real-life scientific codes to the most advanced computing architectures.
- **Challenge**: Device architectures are constantly changing. Current architectures are very various. Our codes need to be very portable and flexible.
- **Goal**: Take HPC to the "Industrie 4.0" by implementing smart techniques to optimize the codes in terms of performance and energy consumption.

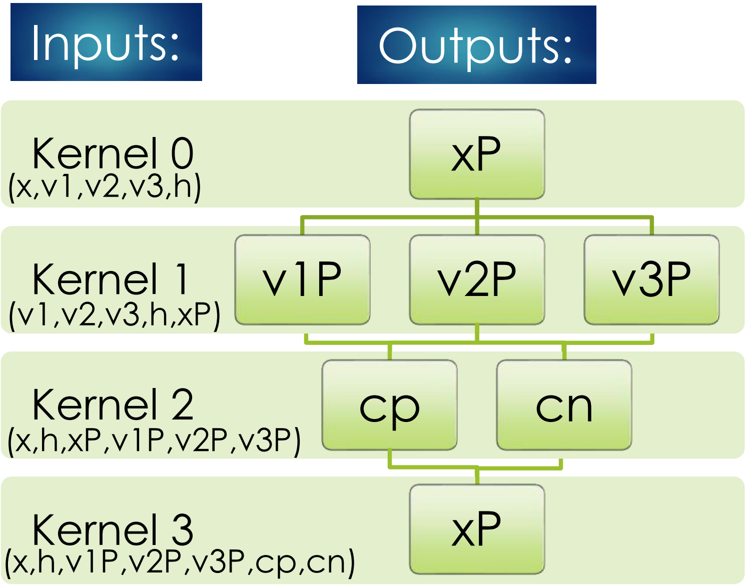
Clusters specification

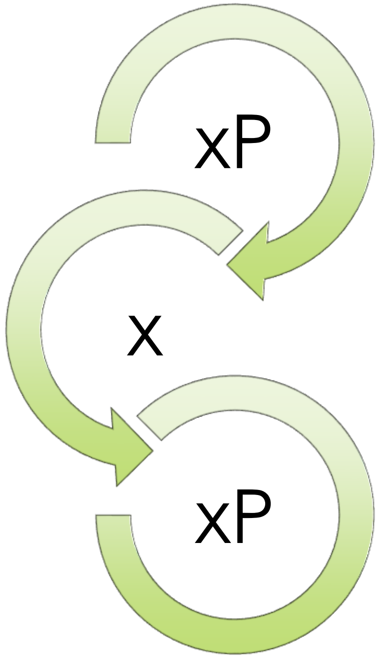
- **Piz Daint** (ranked **3-rd** at top 500):
  - GPU: NVIDIA Tesla **P100 - PASCAL**
  - **1x**GPU per node
  - **Single** GPU design
  - 5320 nodes (up to **36** used in this work)
  - Calculation speed: float is **2x** faster than double
- **MICLAB**:
  - GPU: NVIDIA Tesla **K80 - KEPLER**
  - **2x**GPUs per node
  - **Dual** GPU design
  - **2** nodes (remaining nodes with Intel Xeon Phi)
  - Calculation speed: float is **3x** faster than double
  - Size of data transfer between nodes: **2x** less using float than double
- No access to **sudo** user – it makes a problem when your code is based on DVFS

Expectation: Mixed precision arithmetic allows us to reduce the energy consumption and execution time; It can be used in the real HPC platforms (without special access)

3D MPDATA - Multidimensional Positive Definite Advection Transport Algorithm

- Stencil-based algorithm for numerical simulation of geophysical fluids flows on micro-to-planetary scales:
  - **7** stencils (compressed into **4** kernels) – each depends on one or more others (**343** flops-per-el.)
  - Iterative algorithm – a single iteration represents one time step
  - **11** matrices:
    - **x, xP** – scalar quantity (i.e. temperature); input/output matrices between time steps
    - **v1, v2, v3, v1P, v2P, v3P** – velocity vectors in **i**, **j**, and **k** directions
    - **h** – density matrix
    - **cp, cn** – temporary, intermediate matrices



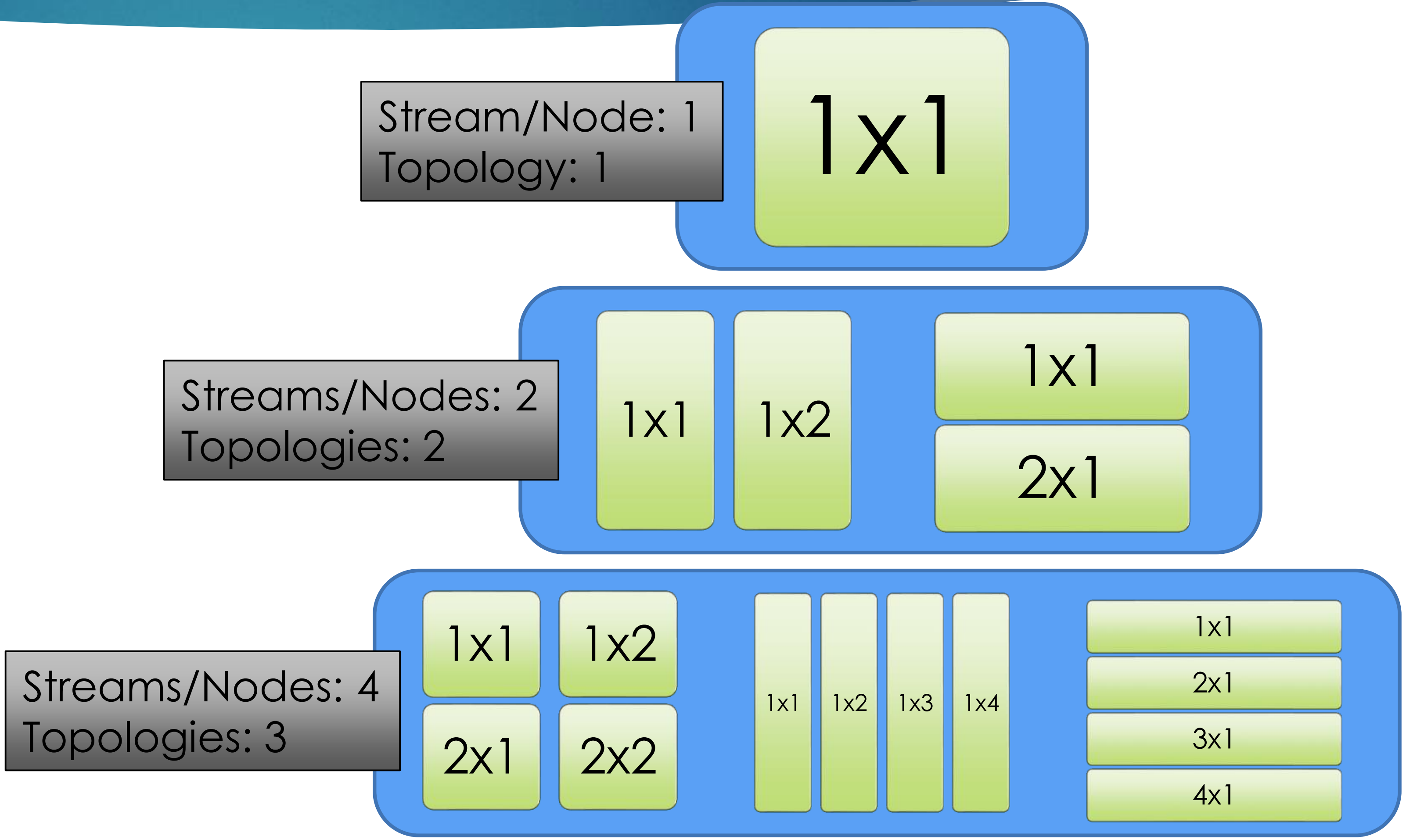


3D MPDATA - Implementation

- **Idea**: Provide a highly parametrized code in order to easily map the algorithm onto GPU
- **Mapping**: Select the right values of given code parameters (configuration) in terms of desired criterion (Energy consumption)
- **How to**: We build the search space of possible configurations and prune it using our machine learning module (MLM)
- **MLM**: It is still the ongoing task. Here we propose to apply the modified random forest algorithm.

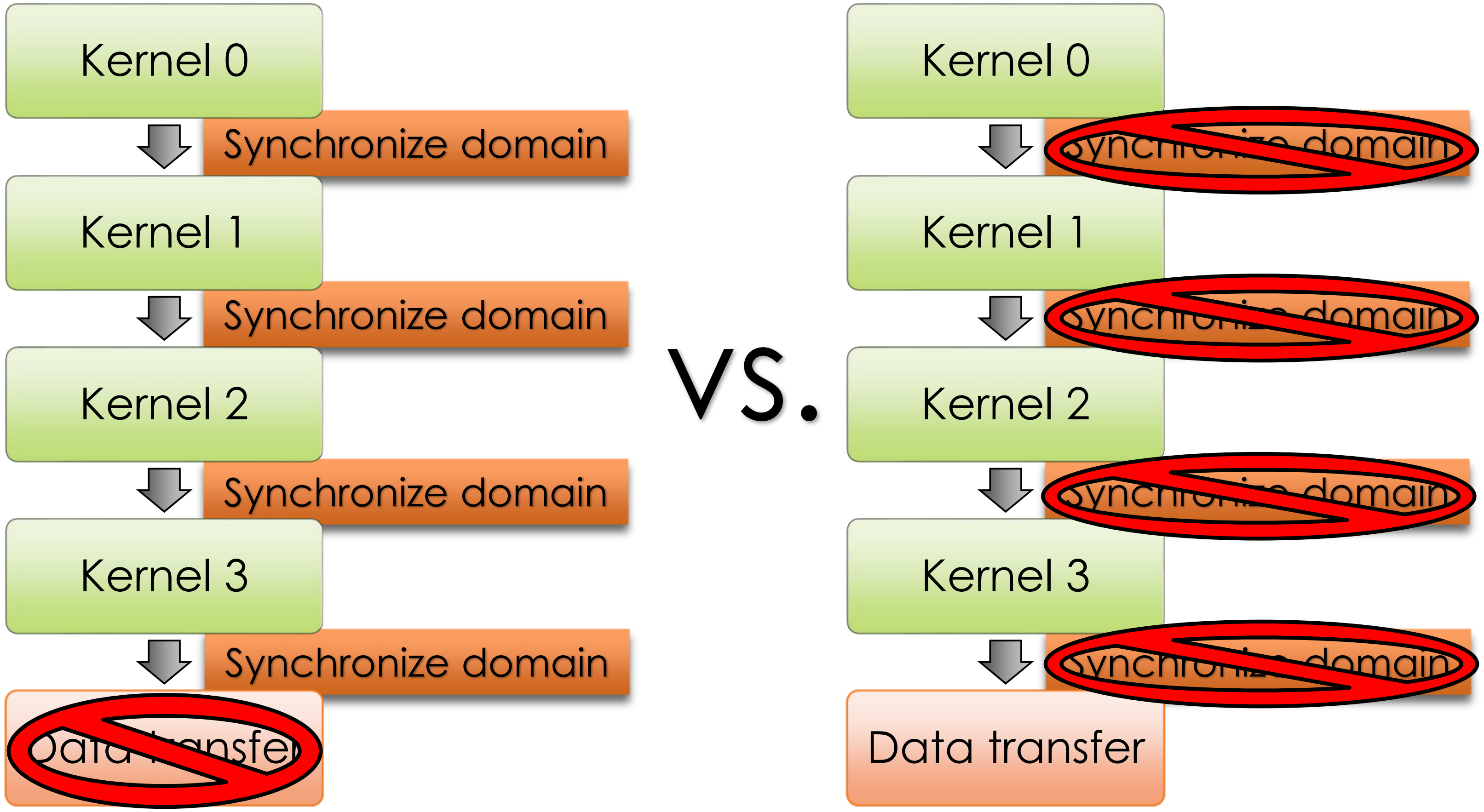
Domain decomposition across GPUs

- We can use different number of:
  - Streams count (SPG)
  - Nodes count (NDS)
- With different topologies:
  - Topology of streams (TGP)
  - Topology of nodes (TDS)



Synchronization vs. data transfer

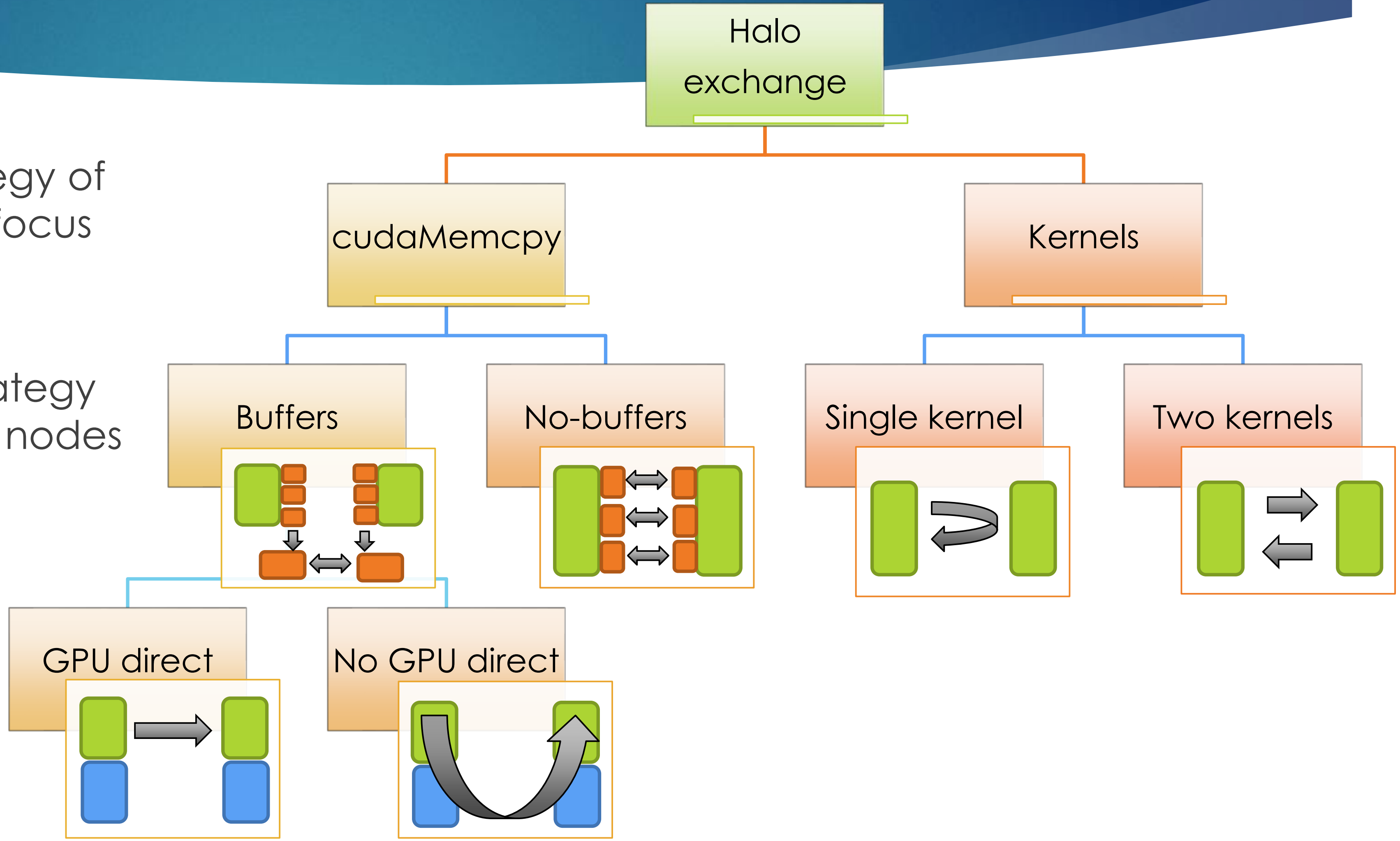
- Each stream share the same subdomain
- Computations depends on the neighboring streams
- Halo exchange is not required within a single node



- Each stream works on distributed subdomains
- Computations are independent within a single time step
- Halo exchange is required after each time step

Technique of halo exchanging

- By selecting the right strategy of halo exchanging we can focus on more parallelism or less operations
- We can use a different strategy within node and between nodes



Other parameters

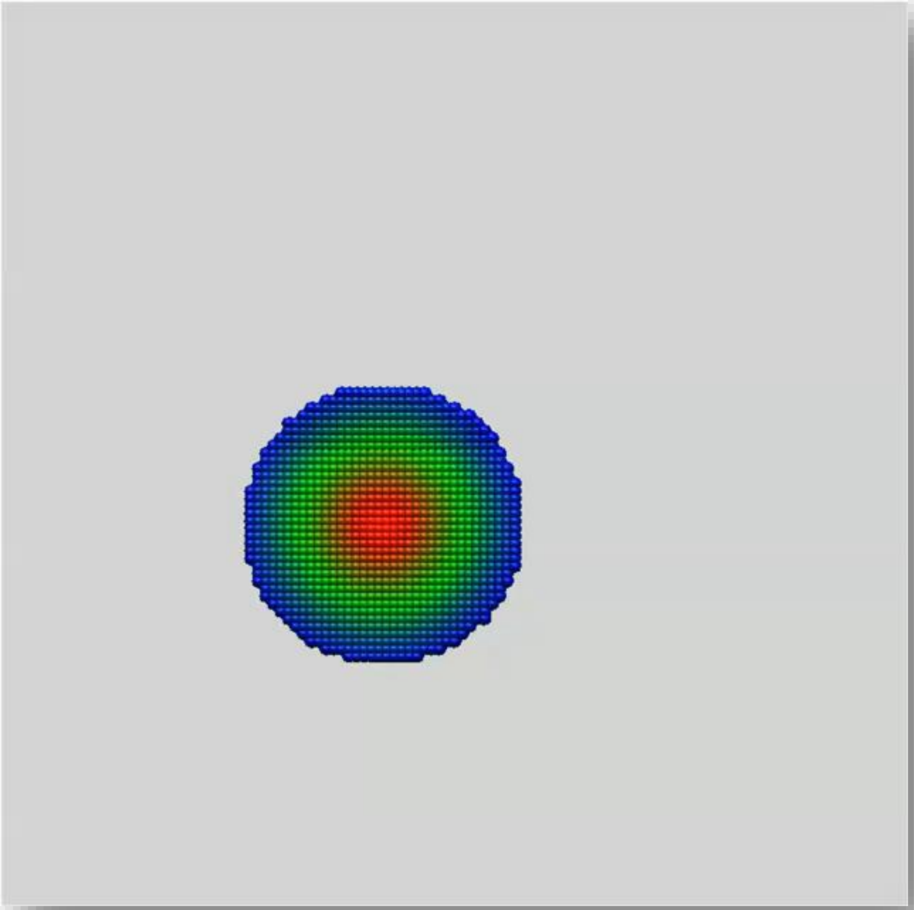
- We takes into consideration also some basic parameters:
  - CUDA block sizes for each of 4 kernels
    - CUDA blocks are of size X times Y, where
      - X*Y mod 32 = 0
      - X>=Y
      - X mod 16 =0
      - X*Y <= 1024
      - X<=M and Y<=N, where NxMxL is a size of the grid
  - Data alignment and padding within a range from 1 to 4096 B
    - Align in: {1, 2, 4, 8, …, 4096}

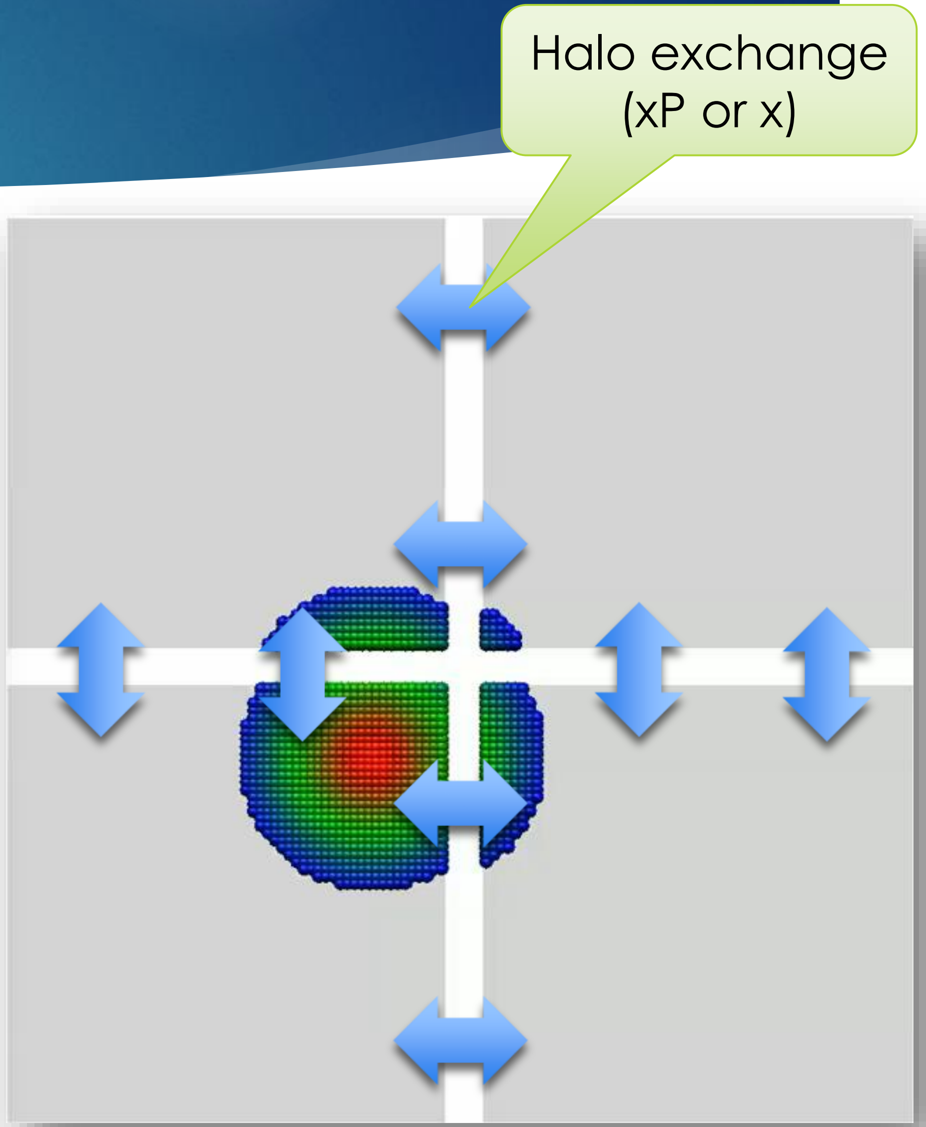
At this moment we are negative ☹

- **Assumption**: We believe that we can find a really good configuration by testing about 5000 configurations from the search space (more that this is too expensive)
- We consider two possible approaches:
  - **Positive**: Find good solutions and eliminate groups that seem to be worse than ours
    - **Risk**: When we find a branch with a good solution we can eliminate other branches (also quite good) that should be worse. In fact we can eliminate a branch containing the best solution.
  - **Negative**: Find bad solutions and eliminate them
    - **Risk**: When we find branches with bad solutions we can eliminate them although the worst one can be still in (the best one also is there).
- **Fact**: We test random branches (we may not select the best or the worst one); we are searching for the suboptimal solution.

Solid body rotation test

- **Precision:**
  - DOUBLE
- **Diameter:**
  - 28.0
- **L2 norm:**
  - 0.0746
- **Diffusion error:**
  - 1.7503
- **Phase error:**
  - 0.7576

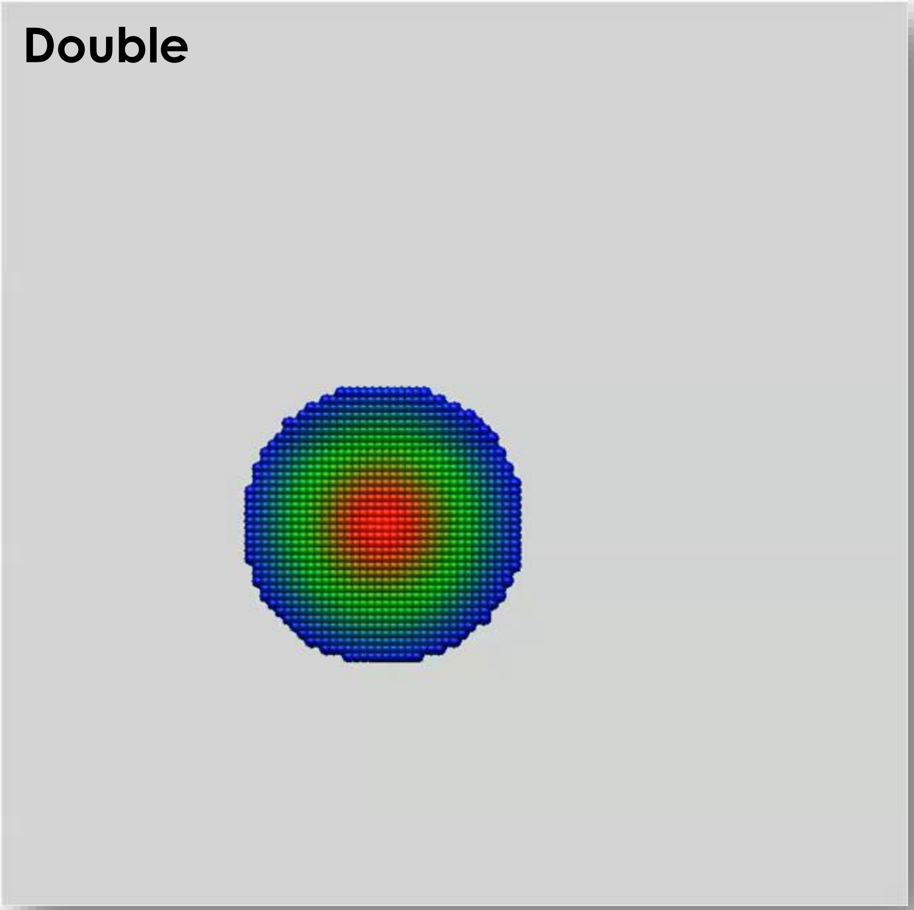


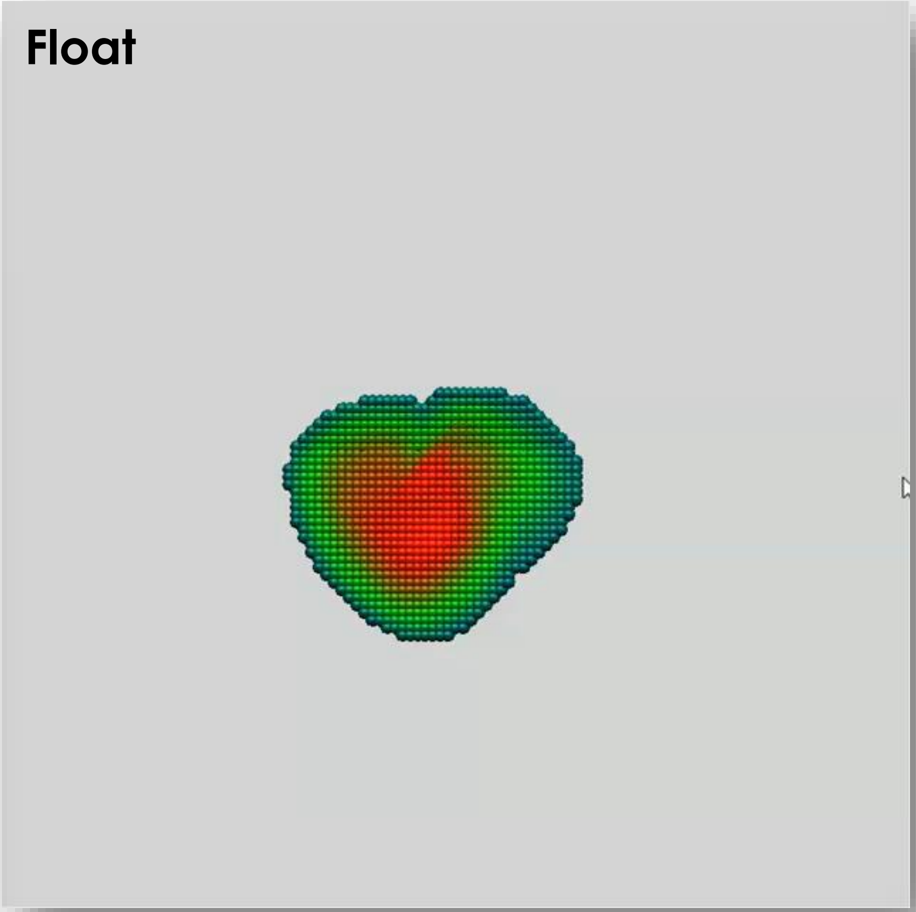


Double vs. float precision

- **Precision:** DOUBLE
- **Diameter**: 28.0
- **L2 norm**: 0.0746
- **Diff. err.**: 1.7503
- **Phase err.**: 0.7576

- **Precision:** FLOAT
- **Diameter**: 28.0
- **L2 norm**: 0.1301
- **Diff. err.**: 2.2439
- **Phase err.**: 7.5919

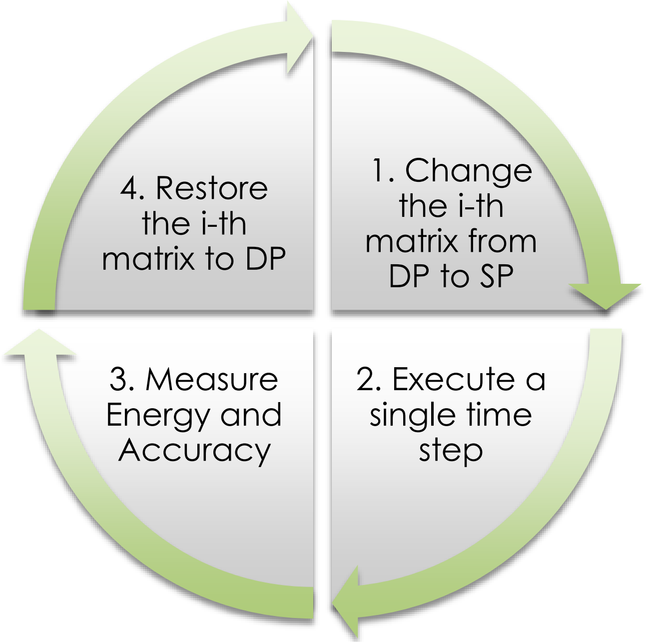




The concept of mixed precision usage

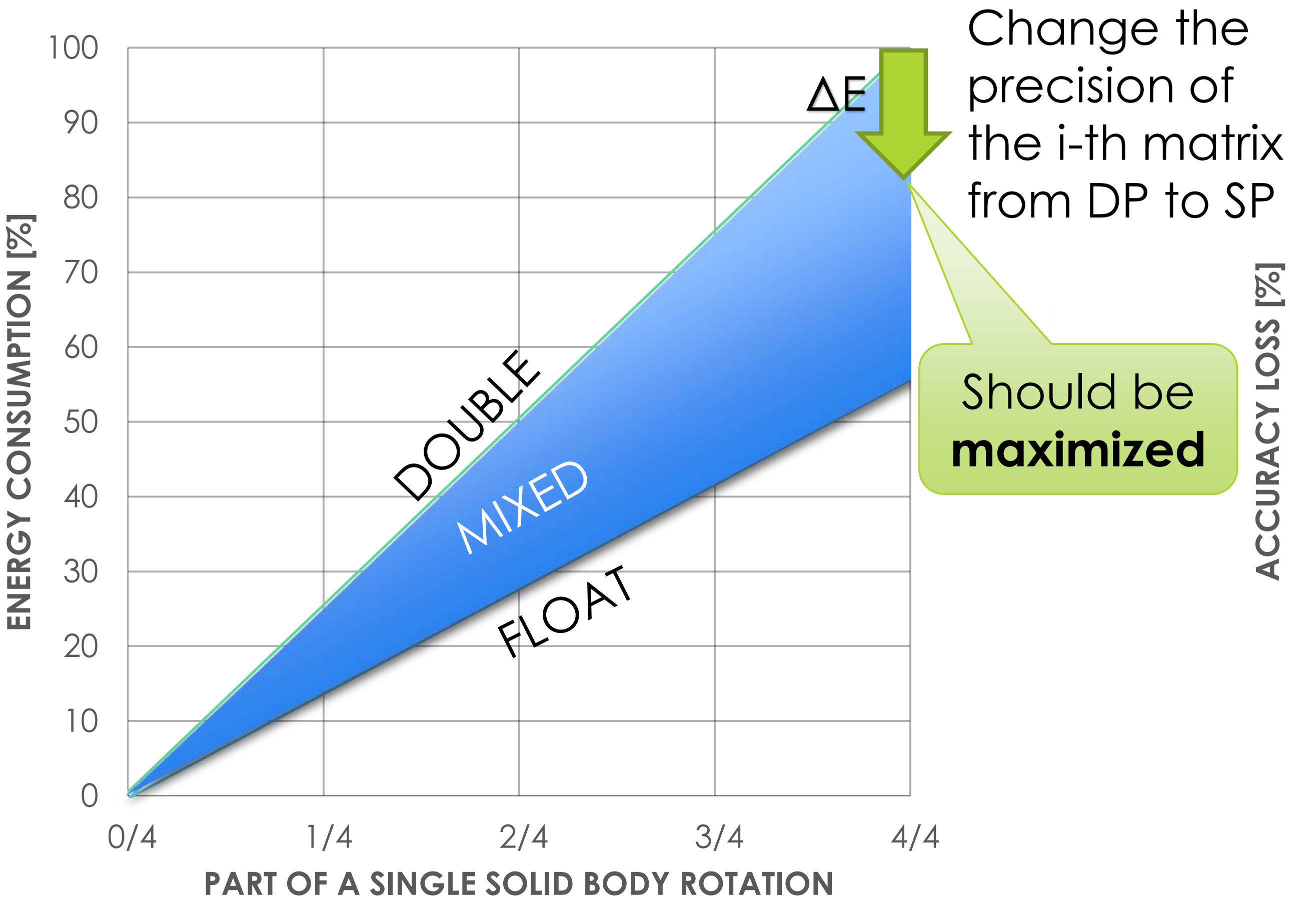
- **Goal**: Reduce the energy consumption
- **Condition**: Keep the accuracy at a high level (**1%** loss is acceptable)
- **Assumptions**:
  - The proposed method is intended to iterative algorithms
  - Dynamic approach, self adaptable to a particular simulation
  - Self adaptation is done based on the short **training stage** (the first **11** time steps)

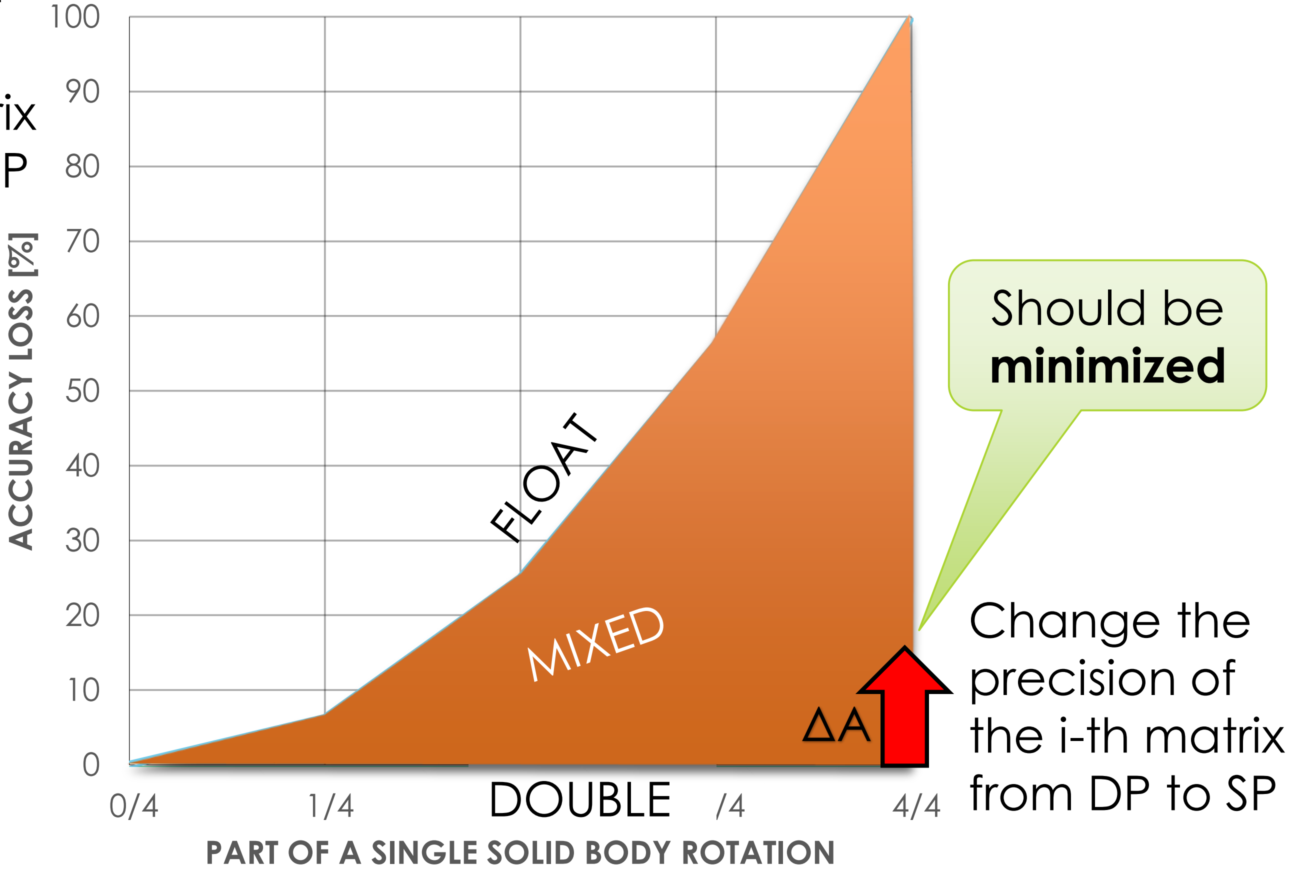
Training stage:



Traditional approach based on static selection of precision arithmetic is less flexible and may be too restrictive for some simulations

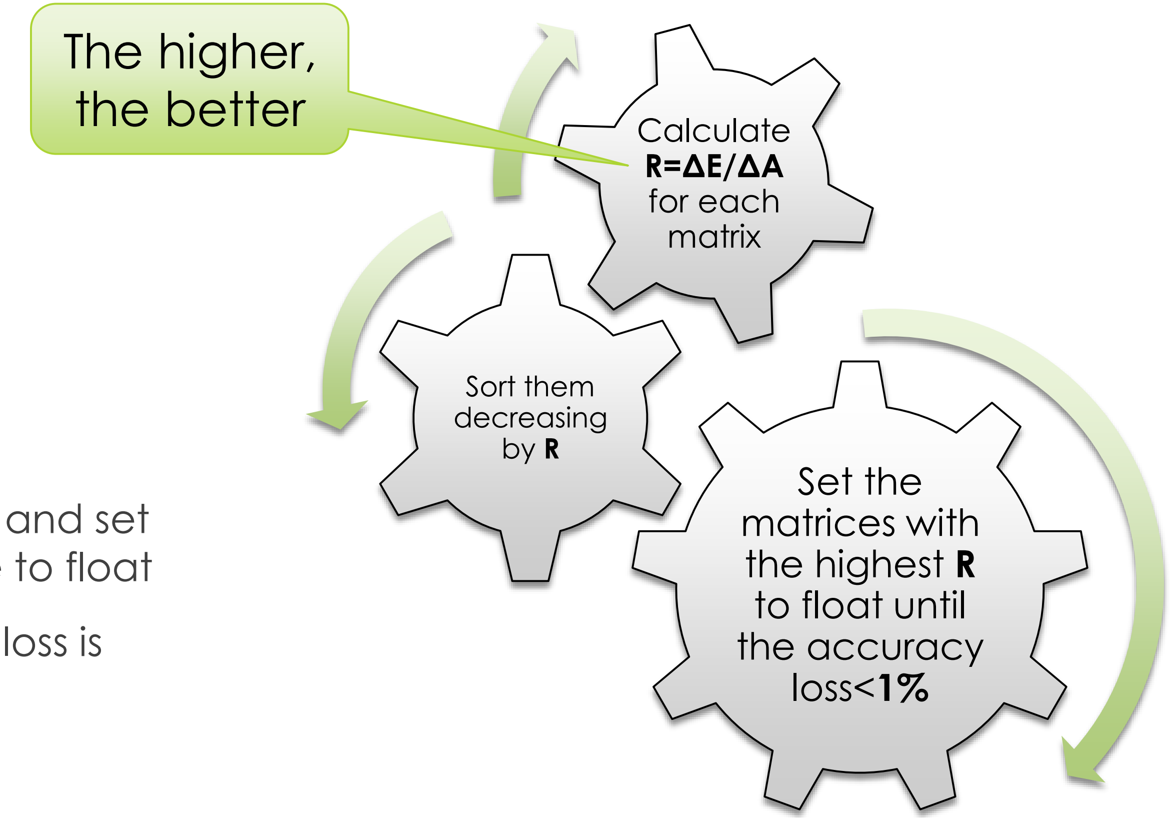
Data analysis from the training stage





Selection of matrices to the float group

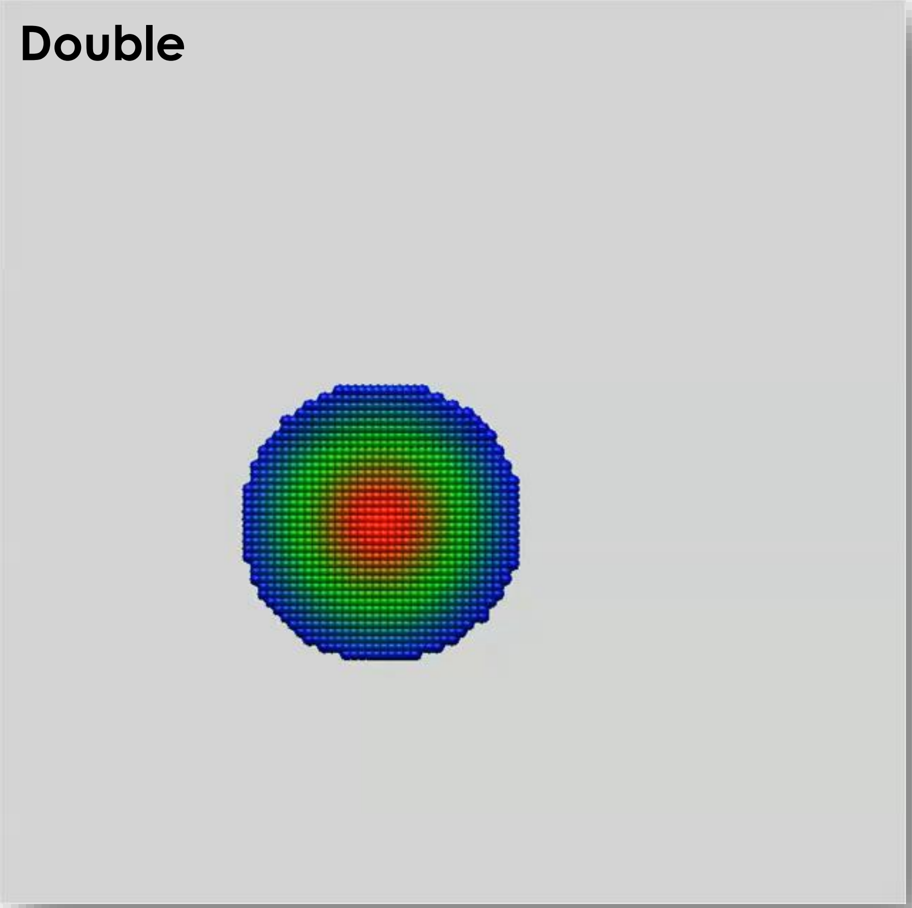
- **Assumptions**:
  - ΔE – should be maximized
  - ΔA – should be minimized
- **Conclusion**:
  - R=ΔE/ΔA – the higher, the better
- **Method**:
  - We estimate the R ratio for each matrix and set matrices with the highest R from double to float
  - This step is repeated until the accuracy loss is lower than 1%

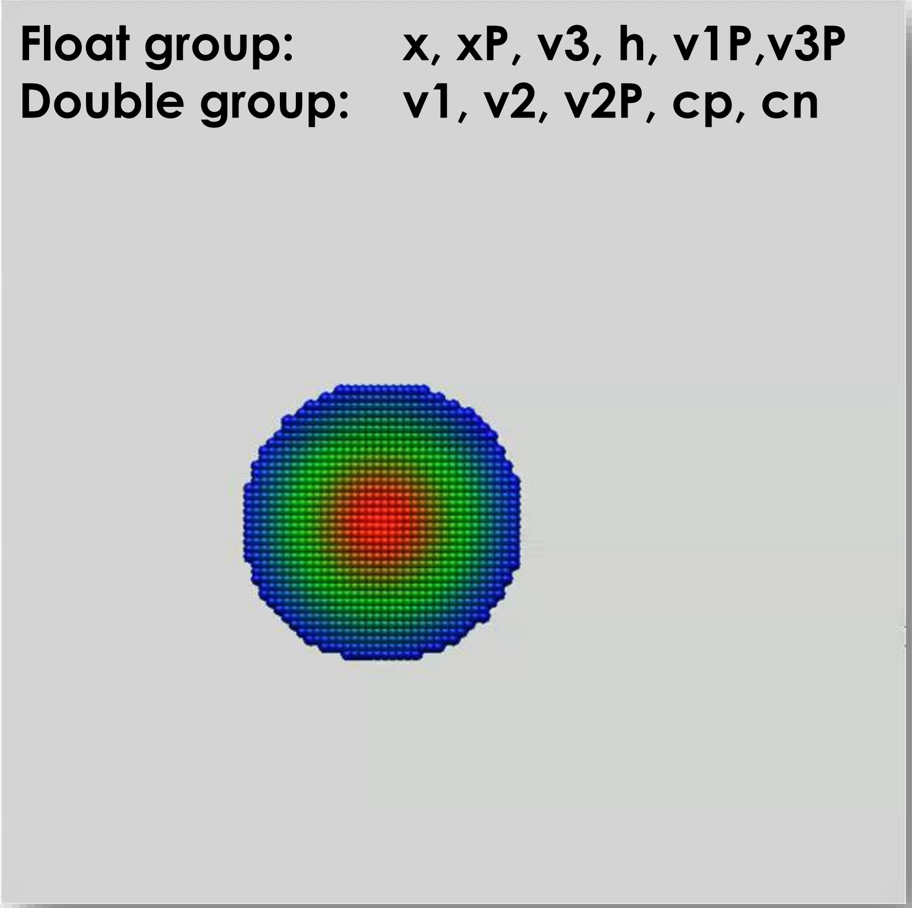


Accuracy: double vs. mixed precision

- **Precision:** DOUBLE
- **Diameter**: 28.0
- **L2 norm**: 0.0746
- **Diff. err.**: 1.7503
- **Phase err.**: 0.7576

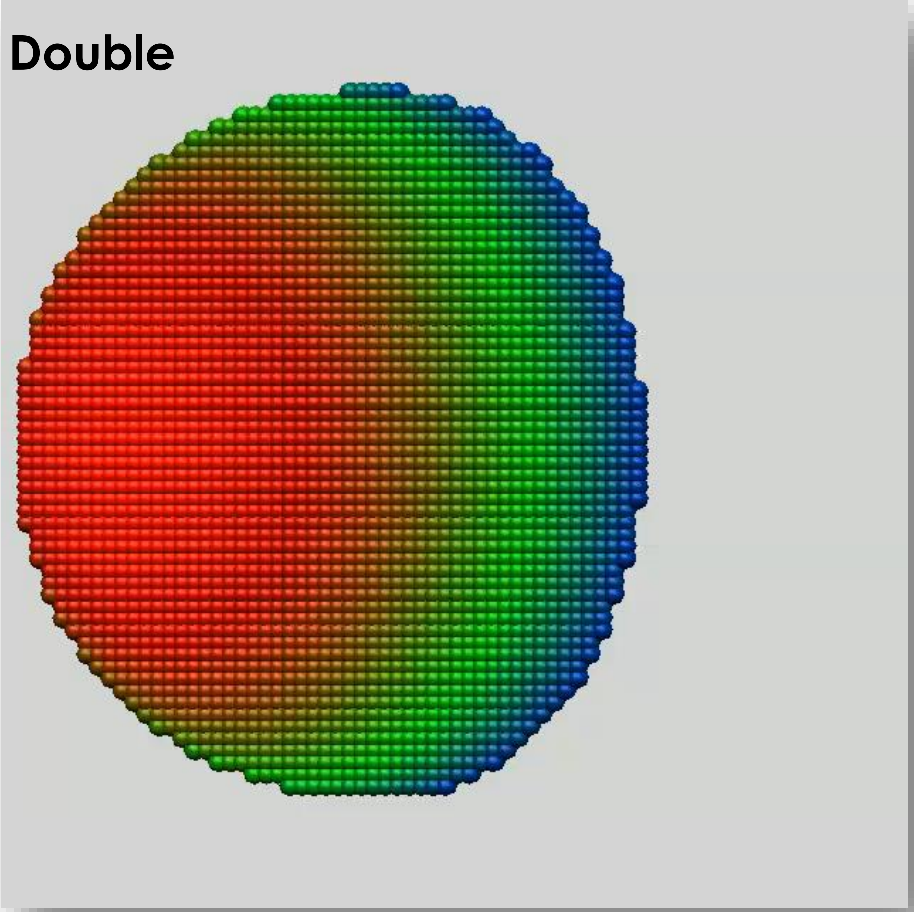
- **Precision:** MIXED
- **Diameter**: 28.0
- **L2 norm**: 0.0749
- **Diff. err.**: 1.7504
- **Phase err.**: 0.7576

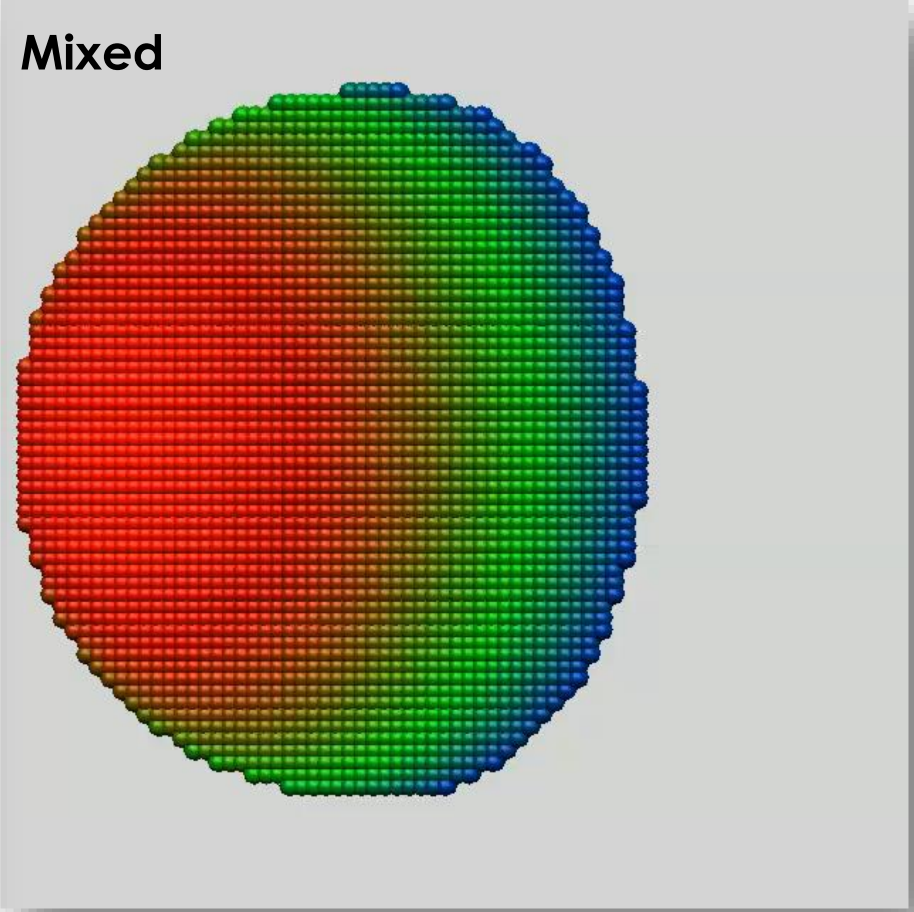




Accuracy: validation for other tests

- The proposed method was also validated for the other tests
- The difference between L2 norms for double and mixed precision is 0.00001
- The phase is 44.2135 for both cases



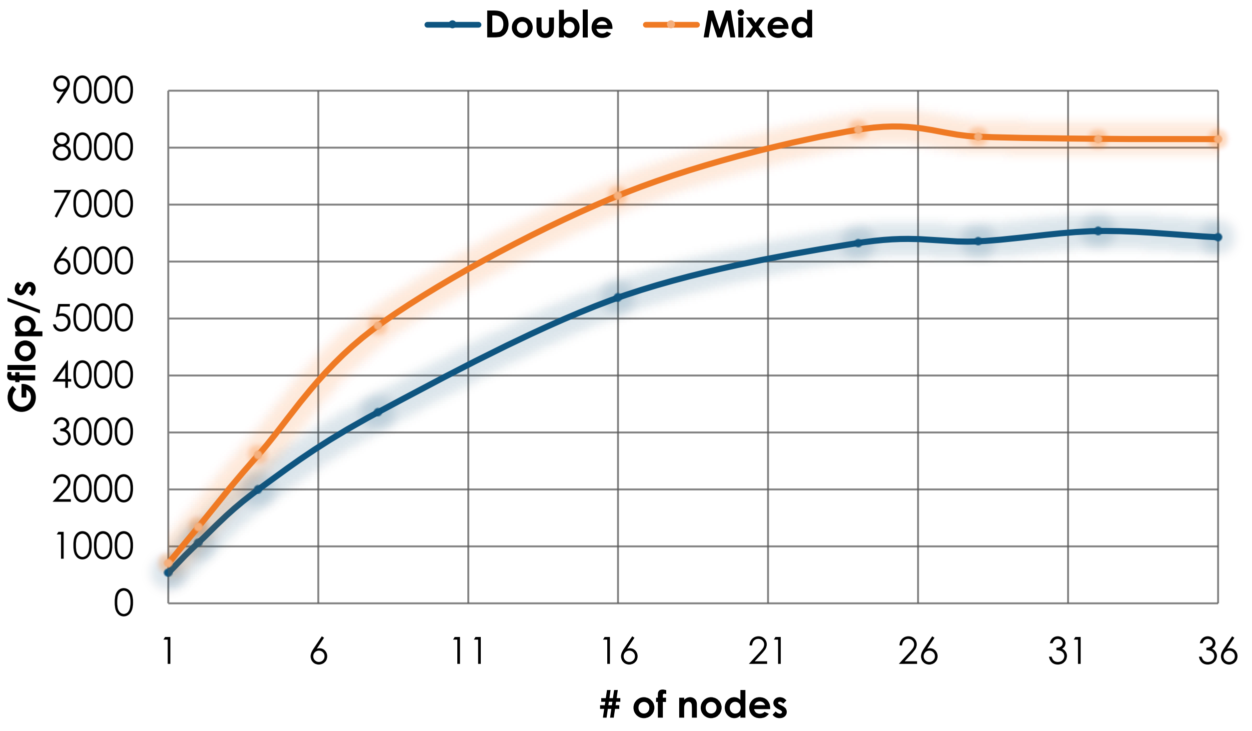


Energy: results from Piz Daint

- Test: 512x512x512 – 3909 time steps

Precision	Nodes	Time [s]	Speed-up	E[kJ]	E red. [%]
Double	1	335.48		44	
Mixed	1	255.02	1.32	35	19.79
Double	32	27.52		71	
Mixed	24	21.65	1.27	48	**32.63**

Best performance



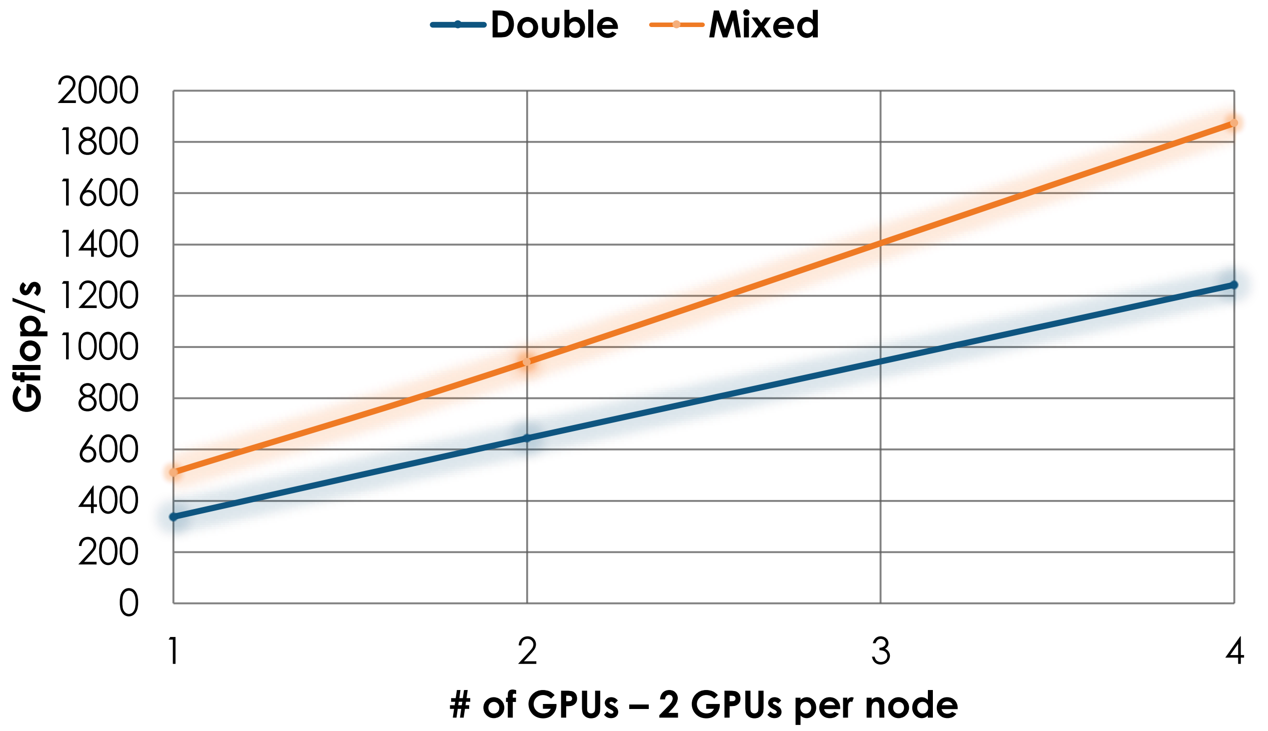
Conclusion: Energy consumption is reduced by **33%**

Energy: results from MICLAB

- Test: 512x512x512 – 3909 time steps

Precision	GPUs/ Chips	Time [s]	Speed -up	E[kJ]	E red. [%]
Double	1/2	533.65		80	
Mixed	1/2	352.18	1.51	53	34.00
Double	4/8	144.83		87	
Mixed	4/8	96.04	1.51	57	**33.66**

Best performance



Conclusion: Energy consumption is reduced by **33%**

Conclusion

- The developed implementation of MPDATA is very flexible and portable
- The proposed method allows us to automate the code adaptation even for a very large number of possible configurations
- Mixed precision arithmetic allows us to reduce the energy consumption and execution time
- It can be used in the real HPC platforms without special access to the machine
- It has an effect on the computation speed, data transfer, and scalability of the application
- The proposed method allows us to reduce the energy consumption by **33%** without loss in accuracy
- It has also improved the performance by the factor of **1.27** for Piz Daint and **1.51** for MICLAB in relation to double precision arithmetic

Thank You!

Contact me: **krojek@icis.pcz.pl**