



# Magritte

A **modern** software library for **3D radiative transfer**

Frederik De Ceuster

in collaboration with

Ward Homan, Jeremy Yates, Leen Decin,  
James Hetherington, Guido Cossu, Peter Boyle, ...



# **Solving Radiative Transfer problems using Math and Technology**



Solving **Radiative Transfer** problems  
using Math and Technology









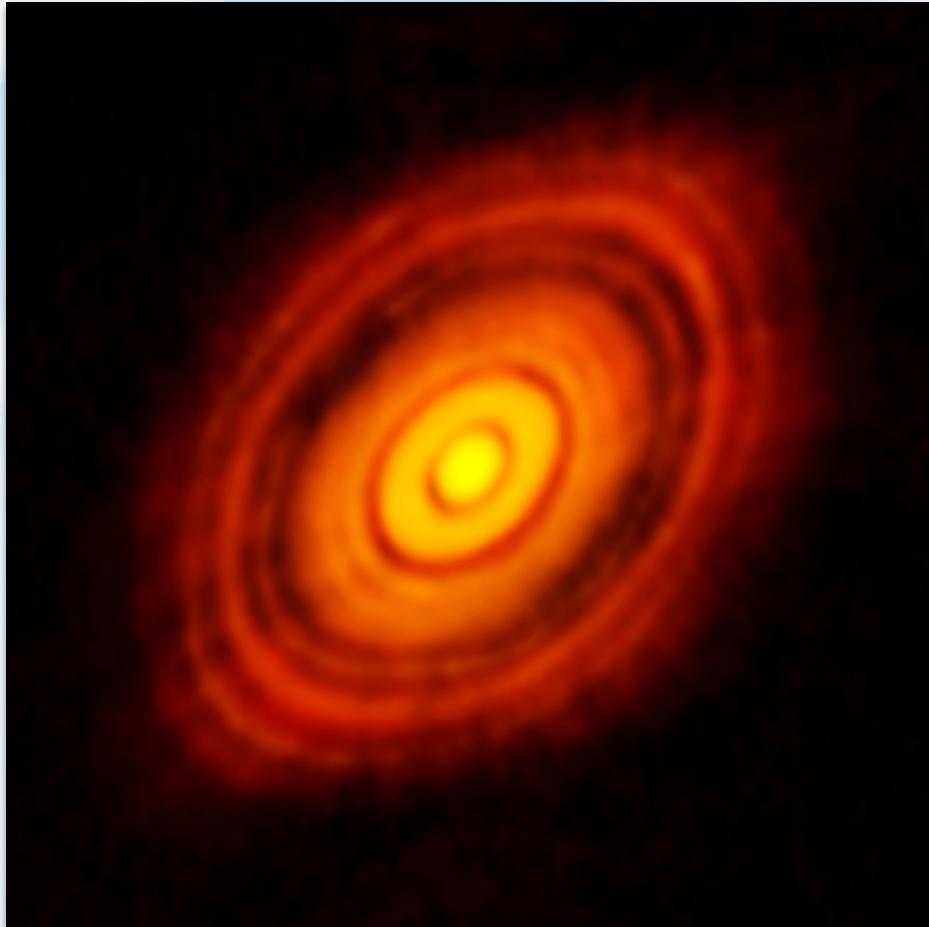
# Atacama Large Millimetre Array





# Disk HL Tauri

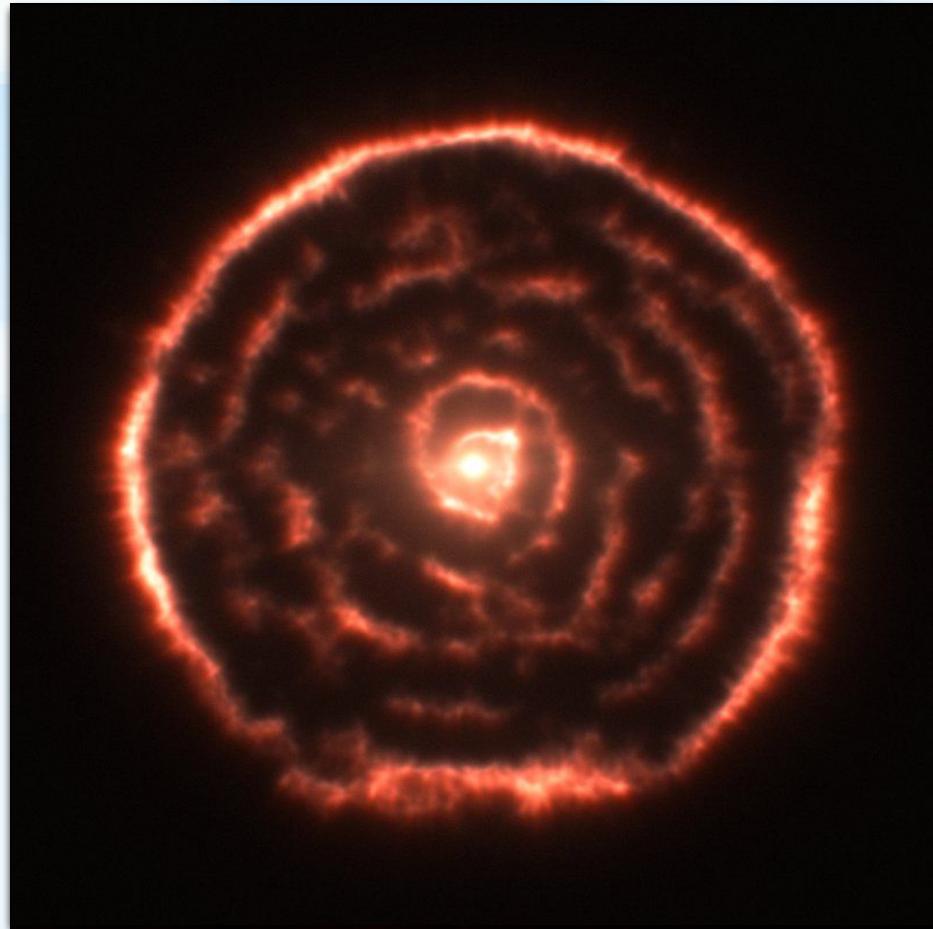
*(ALMA et al. 2015)*

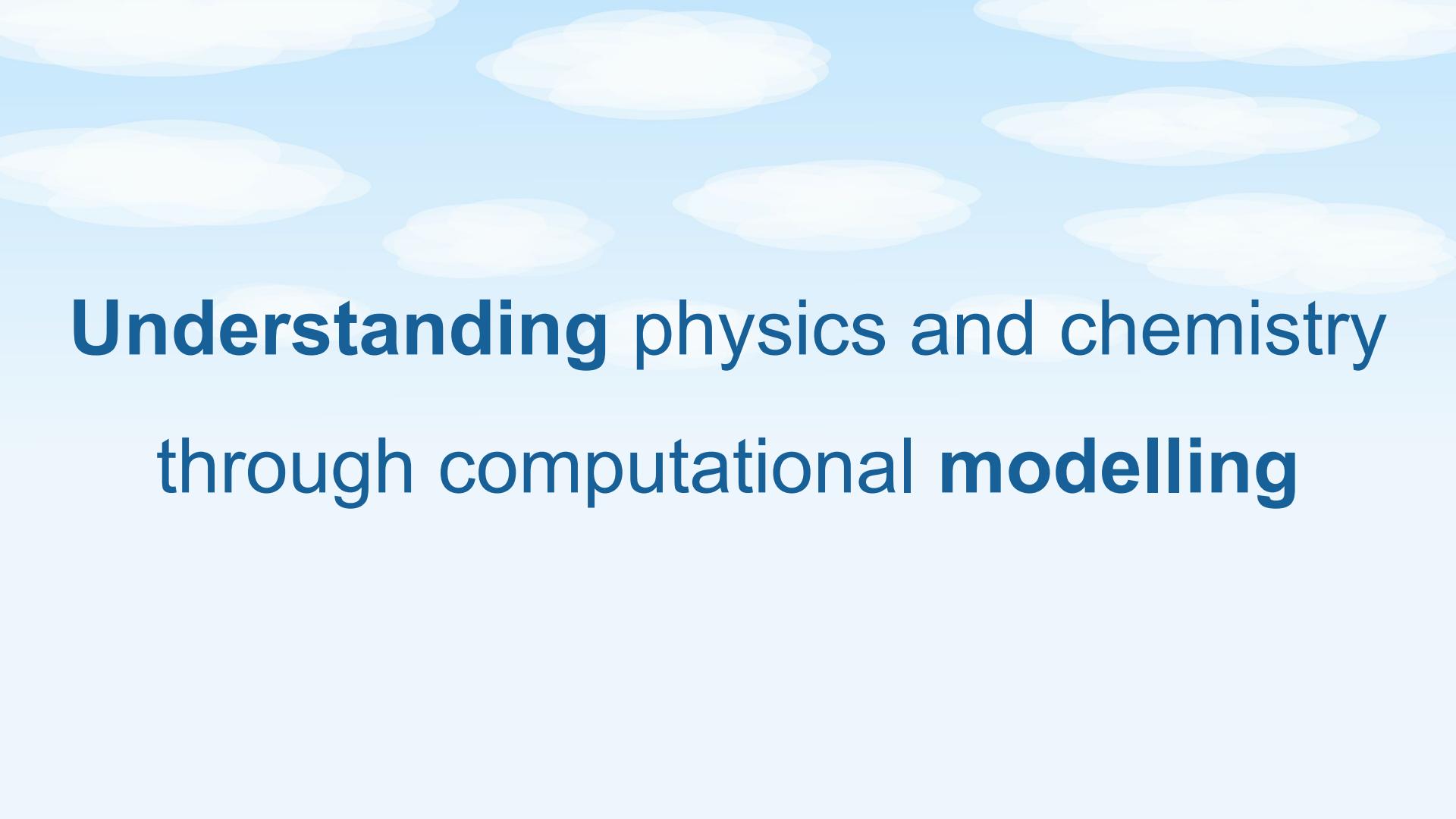




# Spiral R Sculptoris

*(Maercker et al. 2012)*





**Understanding physics and chemistry  
through computational modelling**



# Magritte's Radiative Transfer solver

# Most Radiative Transfer solvers

# Most Radiative Transfer solvers: Monte Carlo

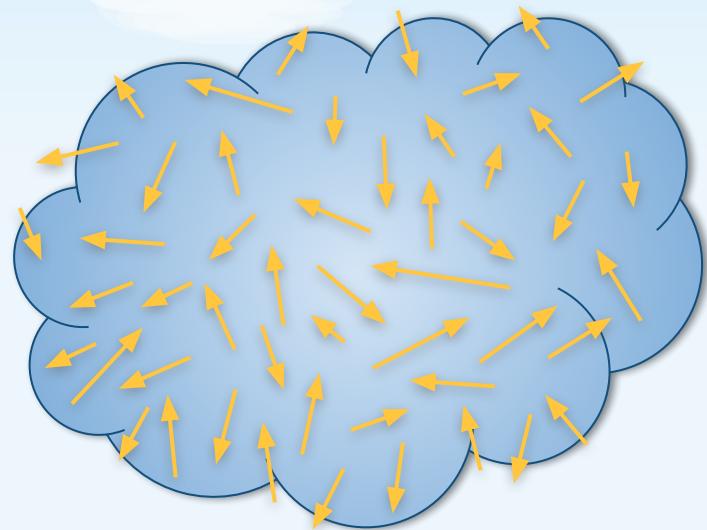
*“Propagate photon packets through the medium”*

Noebauer & Sim (2019)

(+) Fast & Intuitive

(-) Photon trapping

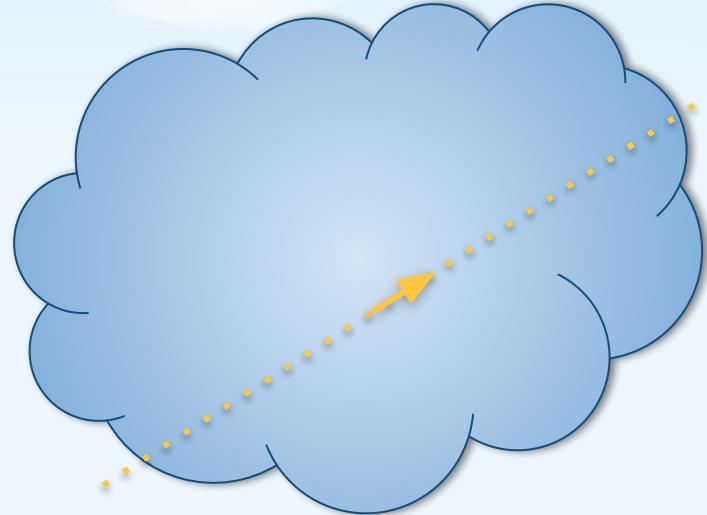
Camps & Baes (2018)





# Magritte's Radiative Transfer solver

***“Solve the radiative transfer equation along a ray”***





# Magritte's Radiative Transfer solver

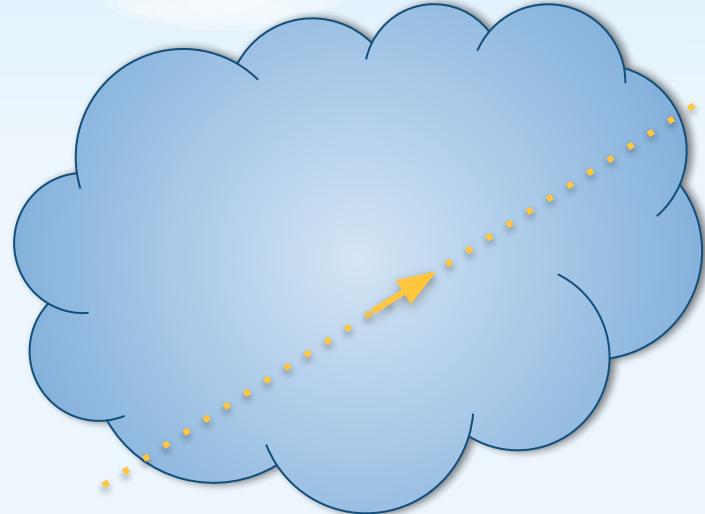
***“Solve the radiative transfer equation along a ray”***

What is a ray?

*Straight line through the model*

How to trace a ray?

...





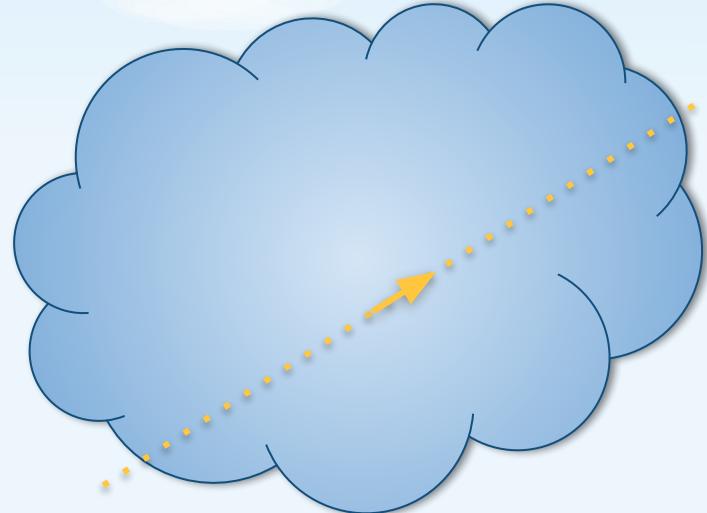
# Magritte's Radiative Transfer solver

***“Solve the radiative transfer equation along a ray”***

$$\frac{dI}{dz} = \eta - \chi I$$

$\eta$  = emissivity

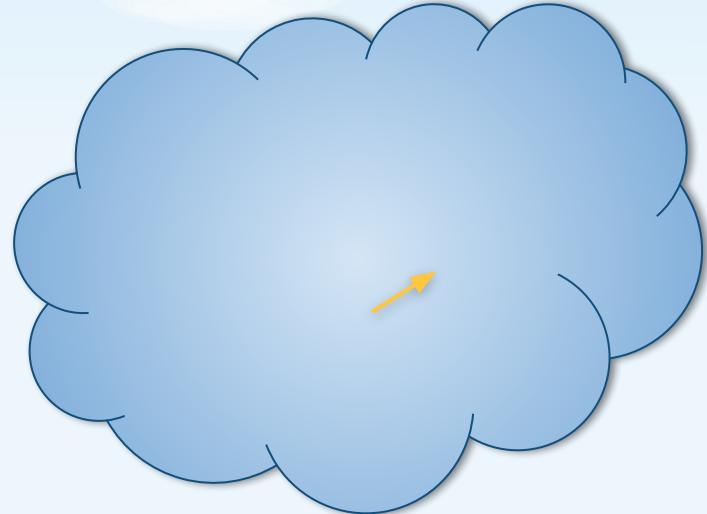
$\chi$  = opacity





# Magritte's Radiative Transfer solver

***“Solve the radiative transfer equation along a ray”***

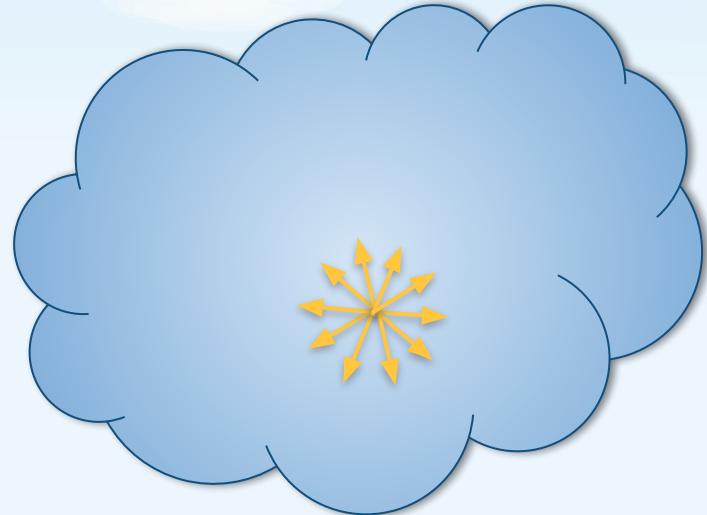




# Magritte's Radiative Transfer solver

***“Solve the radiative transfer equation **along a ray**”***

- *for each direction*

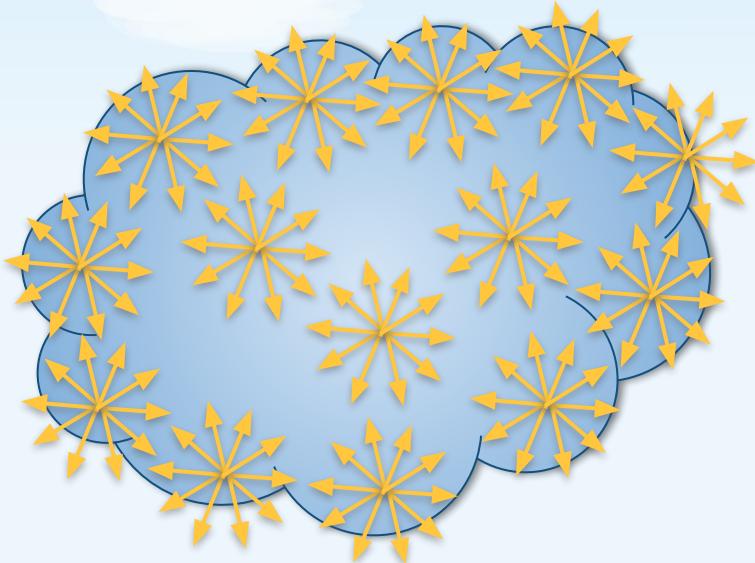




# Magritte's Radiative Transfer solver

***“Solve the radiative transfer equation along a ray”***

- for each ***direction***
- for each ***point***

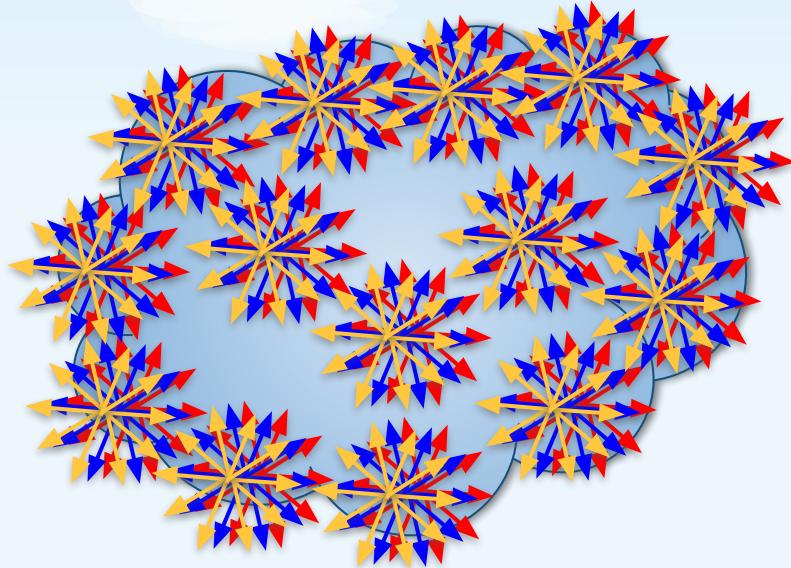




# Magritte's Radiative Transfer solver

***“Solve the radiative transfer equation along a ray”***

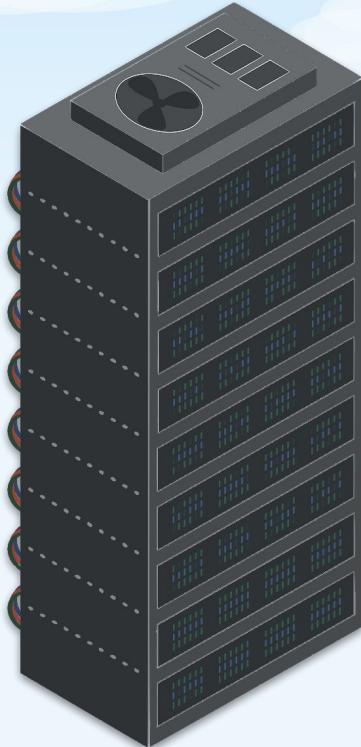
- for each ***direction***
- for each ***point***
- for each ***frequency bin***





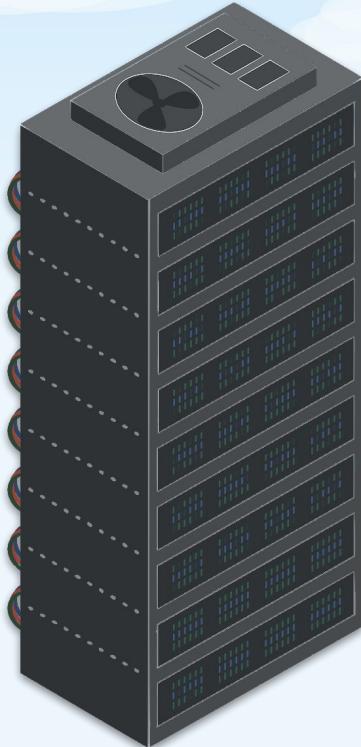
# Technology to the rescue!

# Modern high-performance architecture



- Layers of **parallelism**
- Hardware **accelerators**

# Modern high-performance architecture



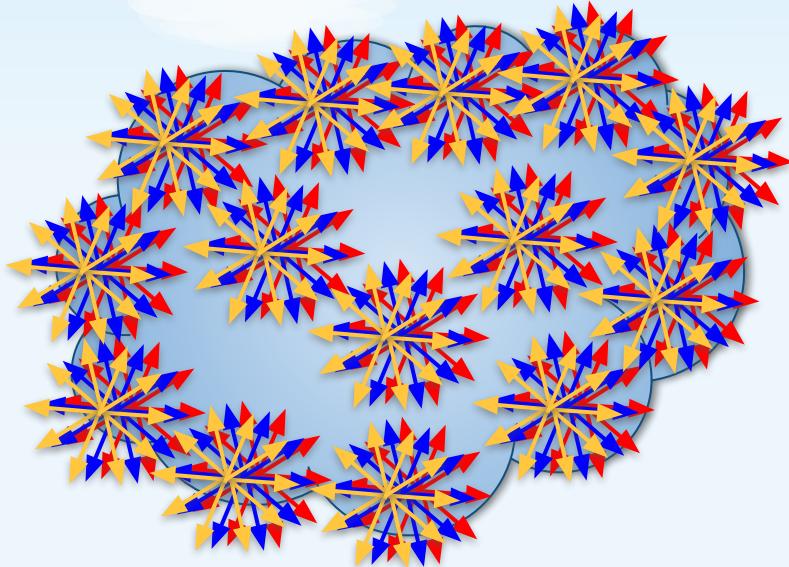
- Layers of **parallelism**
- Hardware **accelerators**



# Magritte's Radiative Transfer solver

***“Solve the radiative transfer equation along a ray”***

- for each ***direction***
- for each ***point***
- for each ***frequency bin***



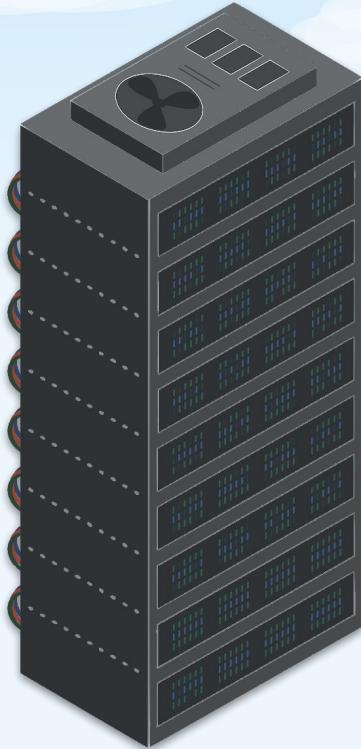


# Magritte's Radiative Transfer solver

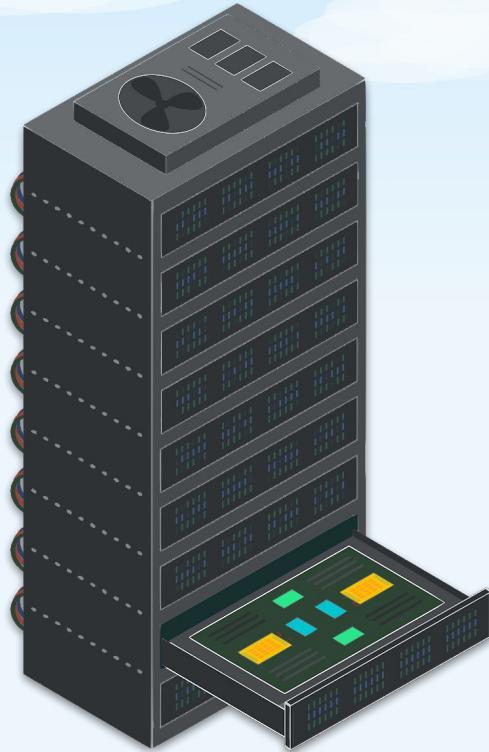
***“Solve the radiative transfer equation **along a ray**”***

- *for each direction*                        nodes                        (MPI)
- *for each point*                              cores / threads    (OpenMP)
- *for each frequency bin*    vectorisation                        (Grid-SIMD)

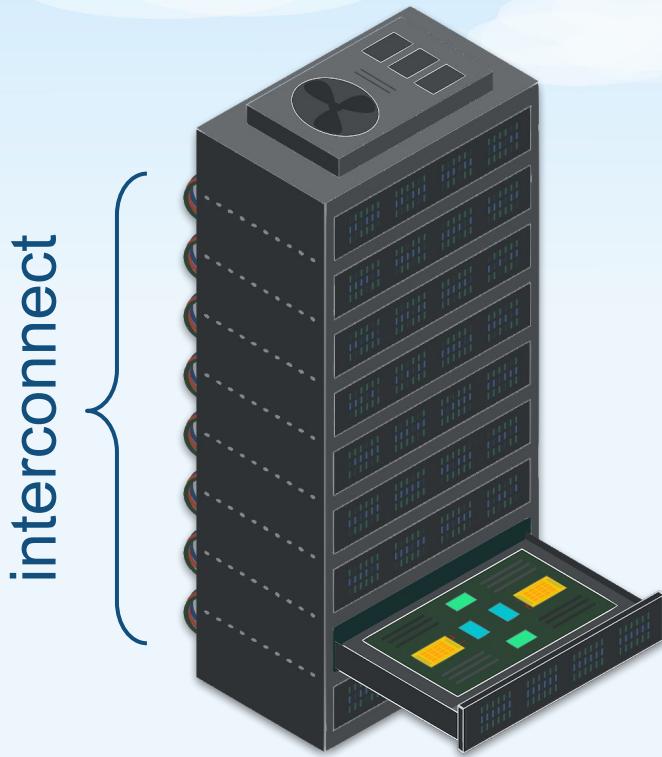
# Modern high-performance architecture



# Modern high-performance architecture



# Modern high-performance architecture

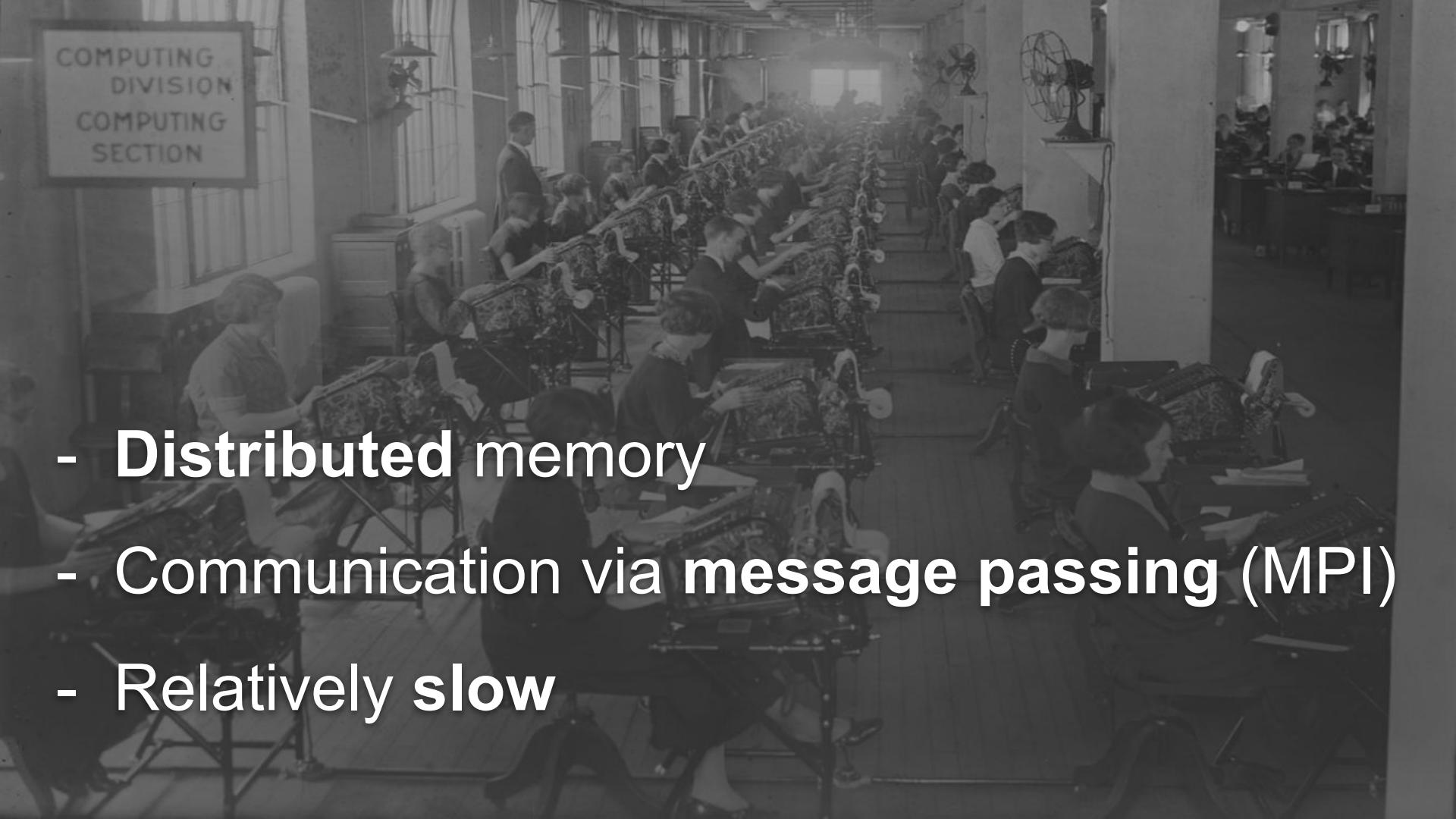


COMPUTING  
DIVISION  
COMPUTING SECTION

# Node-level parallelism

COMPUTING  
DIVISION  
COMPUTING SECTION





COMPUTING  
DIVISION  
COMPUTING  
SECTION

- **Distributed memory**
- Communication via **message passing (MPI)**
- Relatively slow



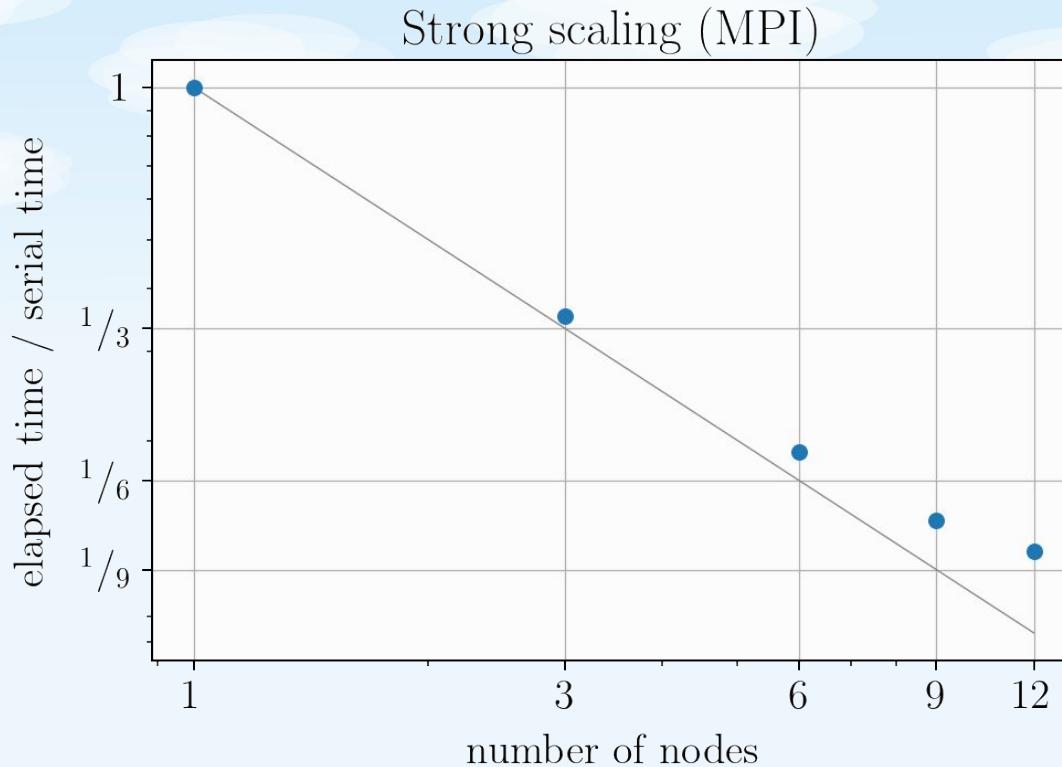
# Magritte's Radiative Transfer solver

***“Solve the radiative transfer equation **along a ray**”***

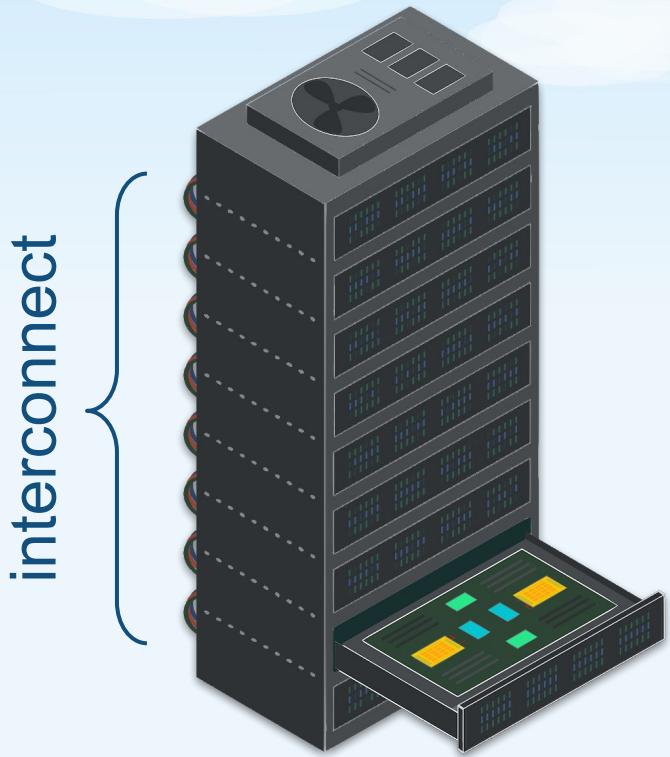
- *for each direction*                    nodes                    (MPI)
- *for each point*                        cores / threads    (OpenMP)
- *for each frequency bin*    vectorisation            (Grid-SIMD)



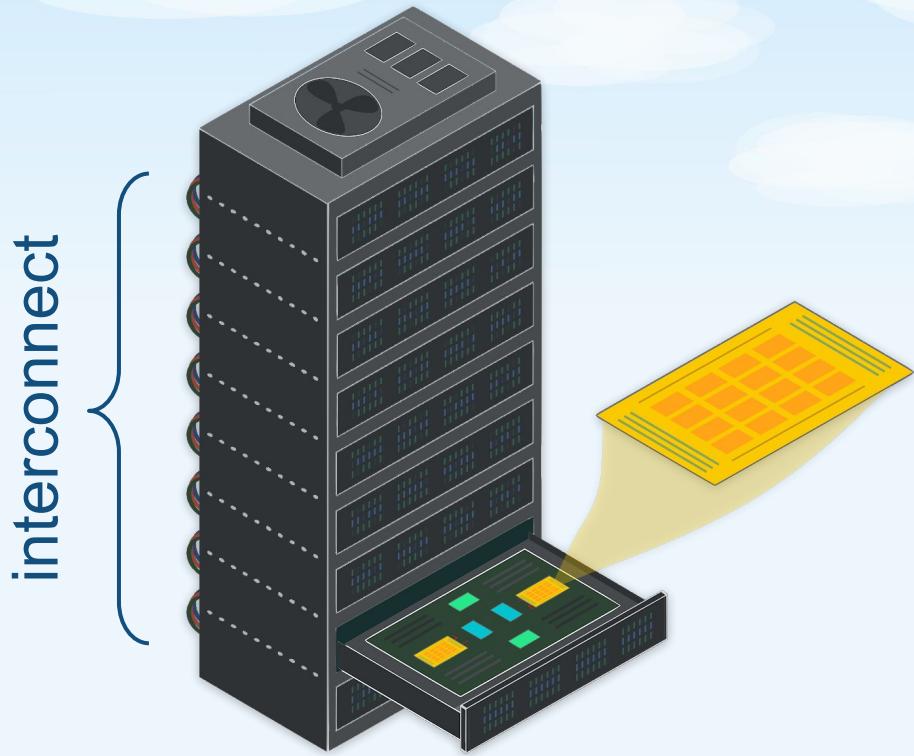
# Magritte's MPI parallelisation



# Modern high-performance architecture

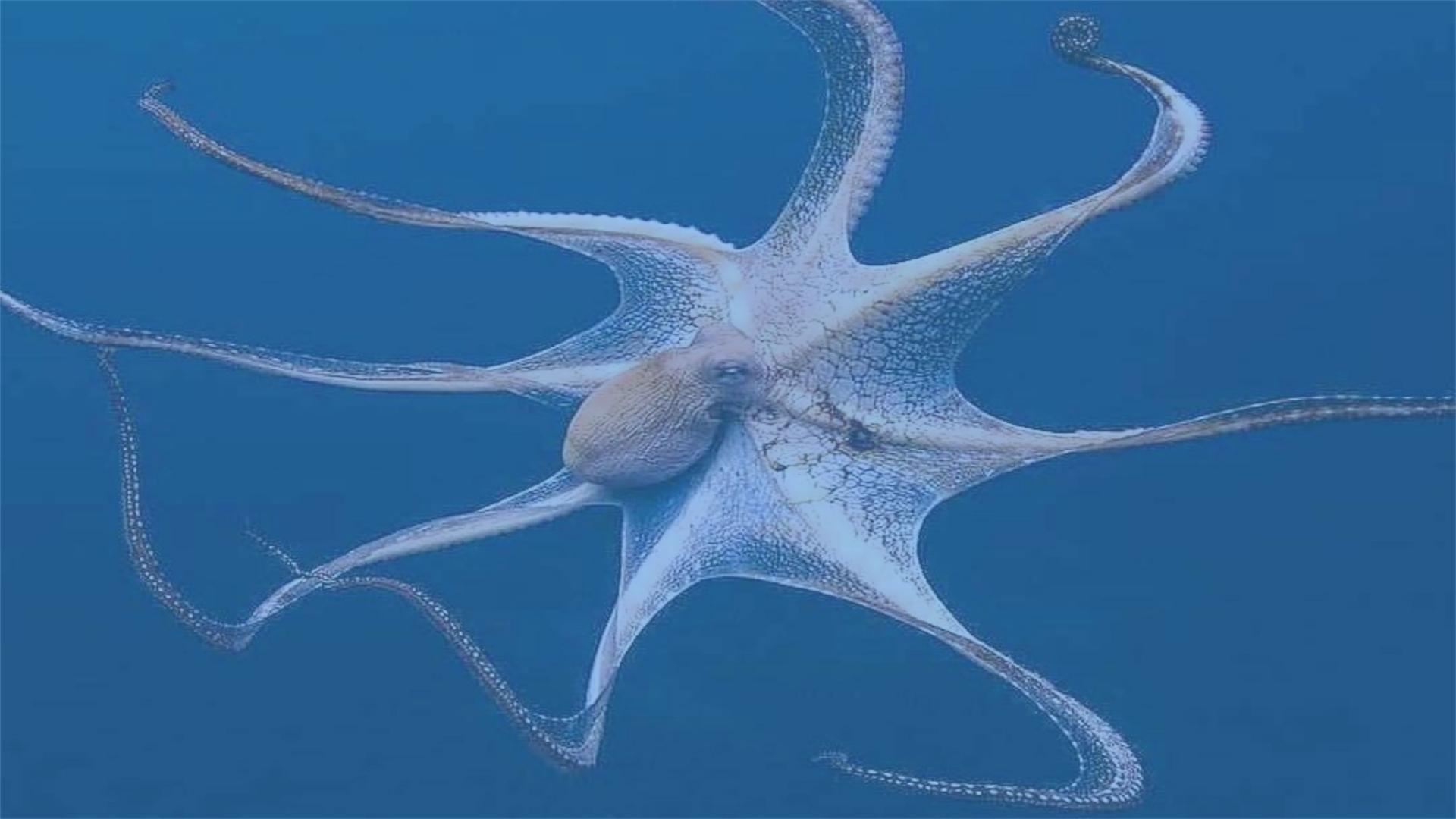


# Modern high-performance architecture



A close-up photograph of an octopus's tentacles against a blue background. The tentacles are textured and have suckers at their tips. They are spread out, creating a sense of movement. The lighting highlights the texture of the tentacles.

# Thread-level parallelism



- Shared memory
- Communication via **memory** (OpenMP)
- Relatively **fast**



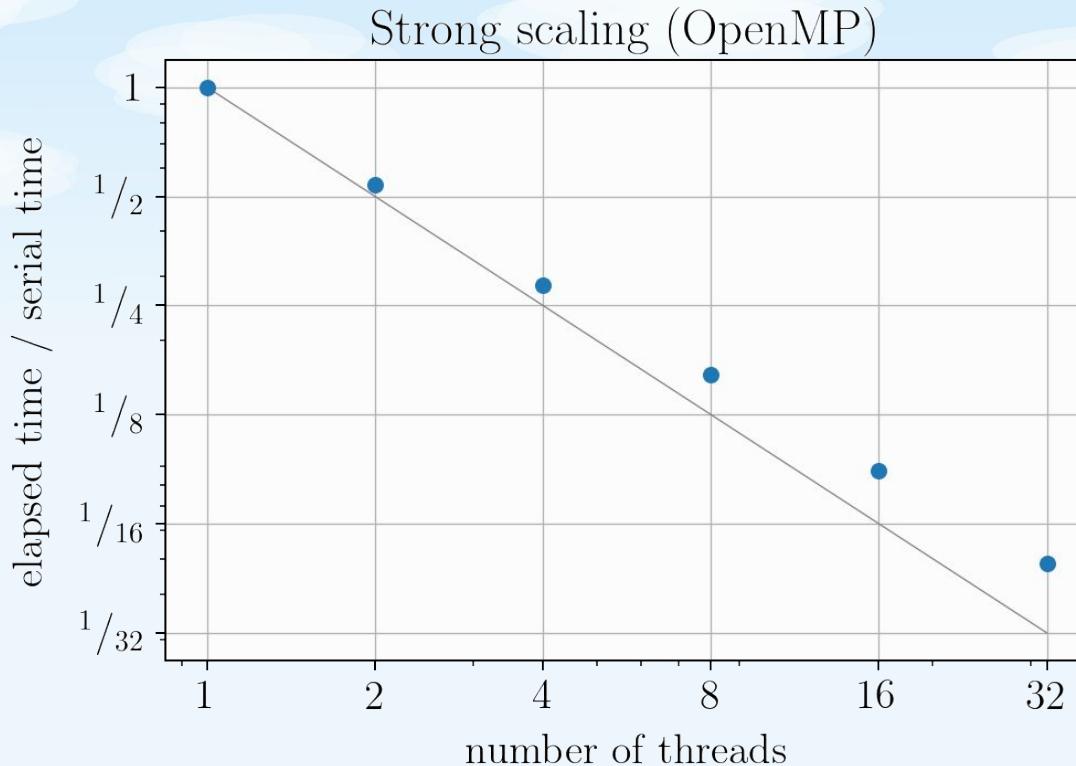
# Magritte's Radiative Transfer solver

***“Solve the radiative transfer equation **along a ray**”***

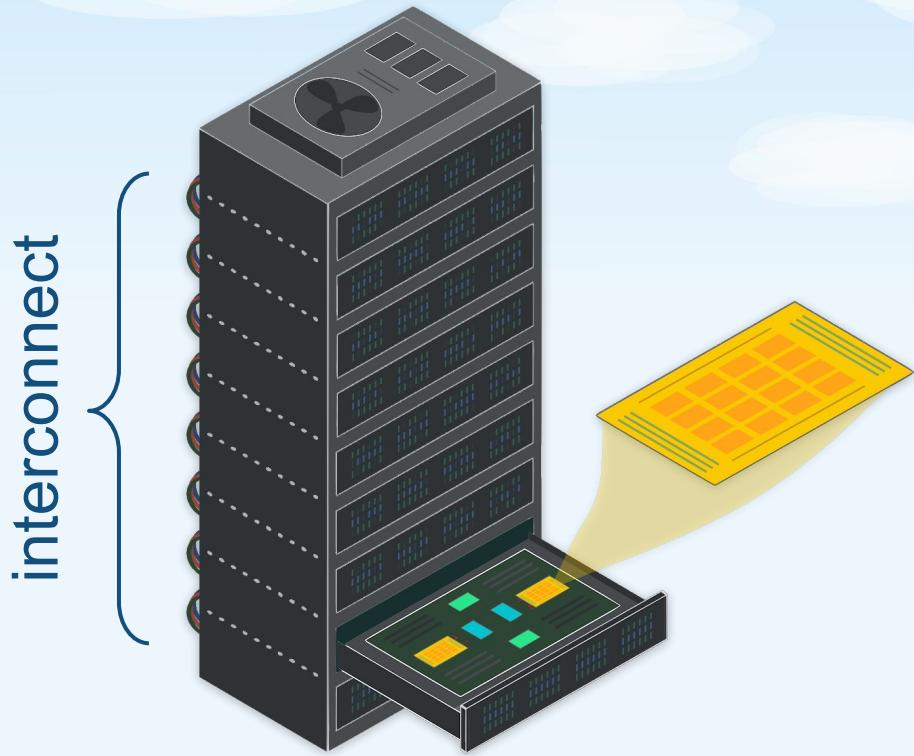
- *for each direction*                    nodes                    (MPI)
- *for each point*                        cores / threads    (OpenMP)
- *for each frequency bin*    vectorisation            (Grid-SIMD)



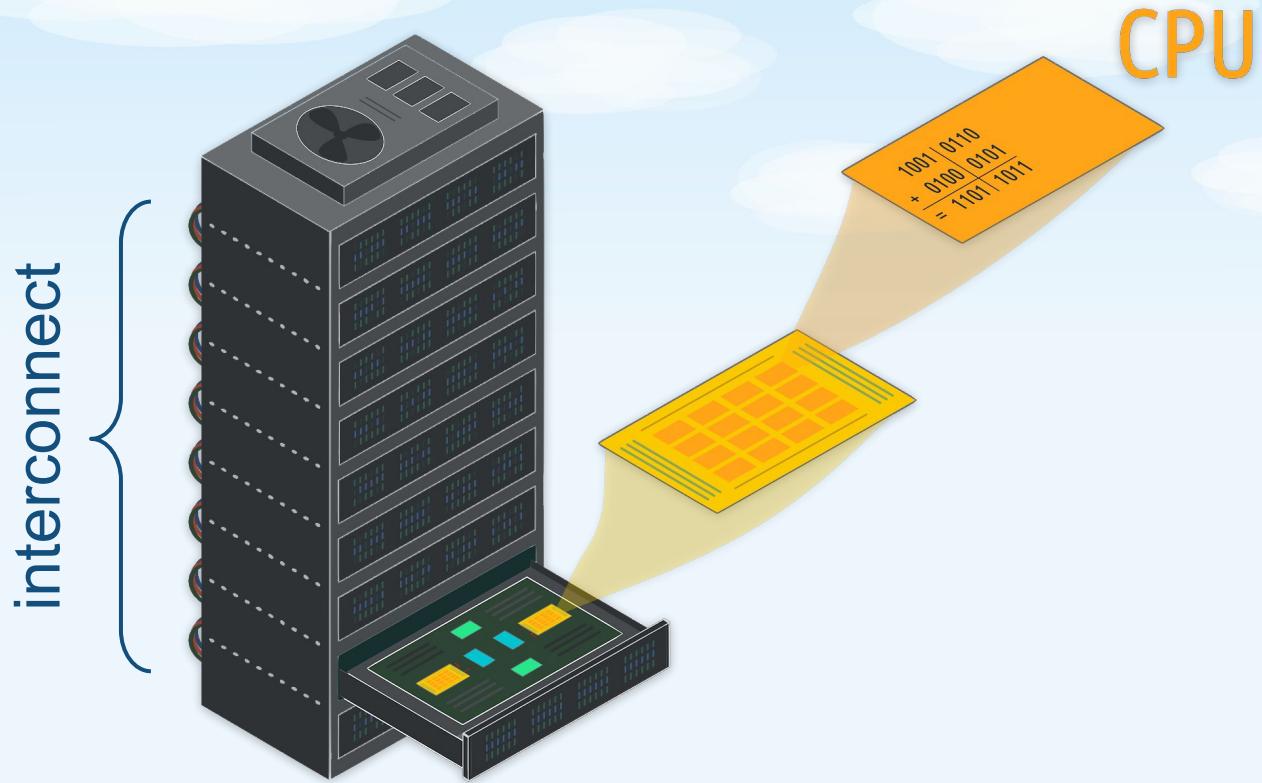
# Magritte's OpenMP parallelisation



# Modern high-performance architecture



# Modern high-performance architecture



# Vectorisation



# SIMD-parallelism



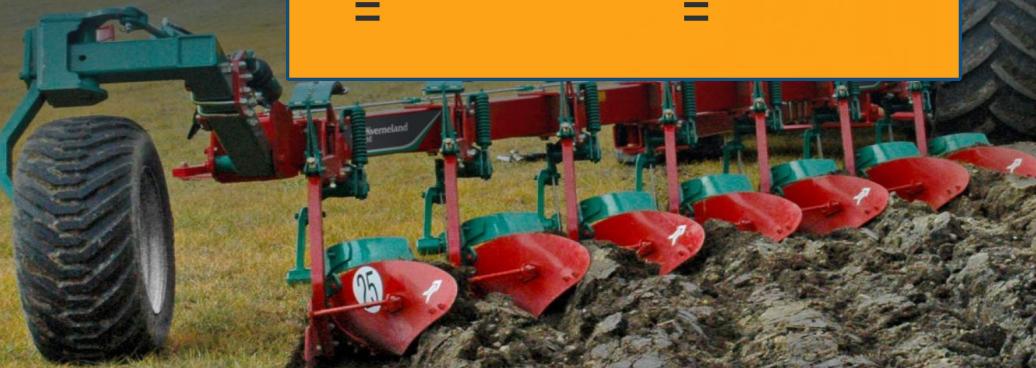
# Single Instruction Multiple Data



# Single Instruction Multiple Data

$$\begin{array}{r} 1001 \\ + 0100 \\ \hline = \end{array} \qquad \begin{array}{r} 0110 \\ + 0101 \\ \hline = \end{array}$$

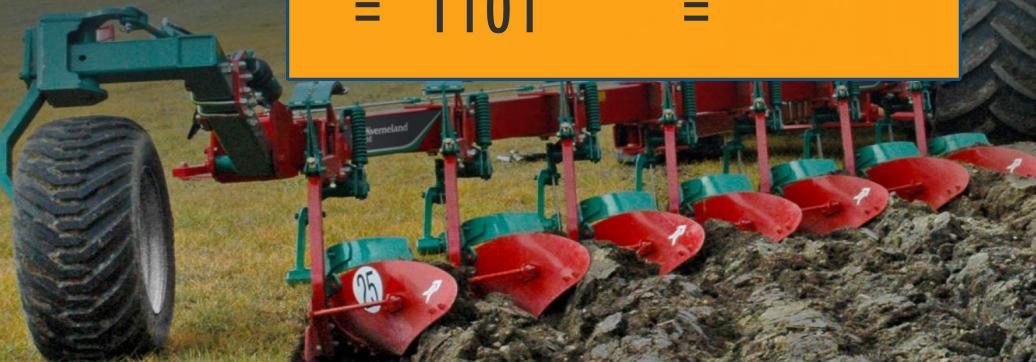
2 regular instr.



# Single Instruction Multiple Data

$$\begin{array}{r} 1001 \\ + 0100 \\ \hline = 1101 \end{array} \qquad \begin{array}{r} 0110 \\ + 0101 \\ \hline = \end{array}$$

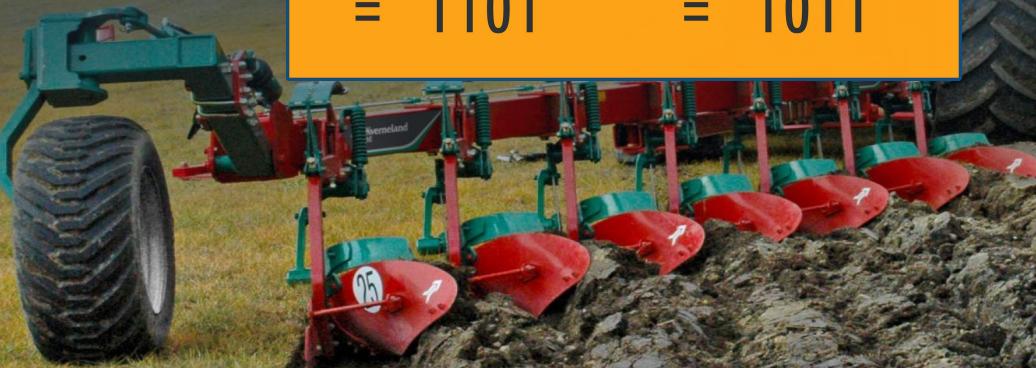
2 regular instr.



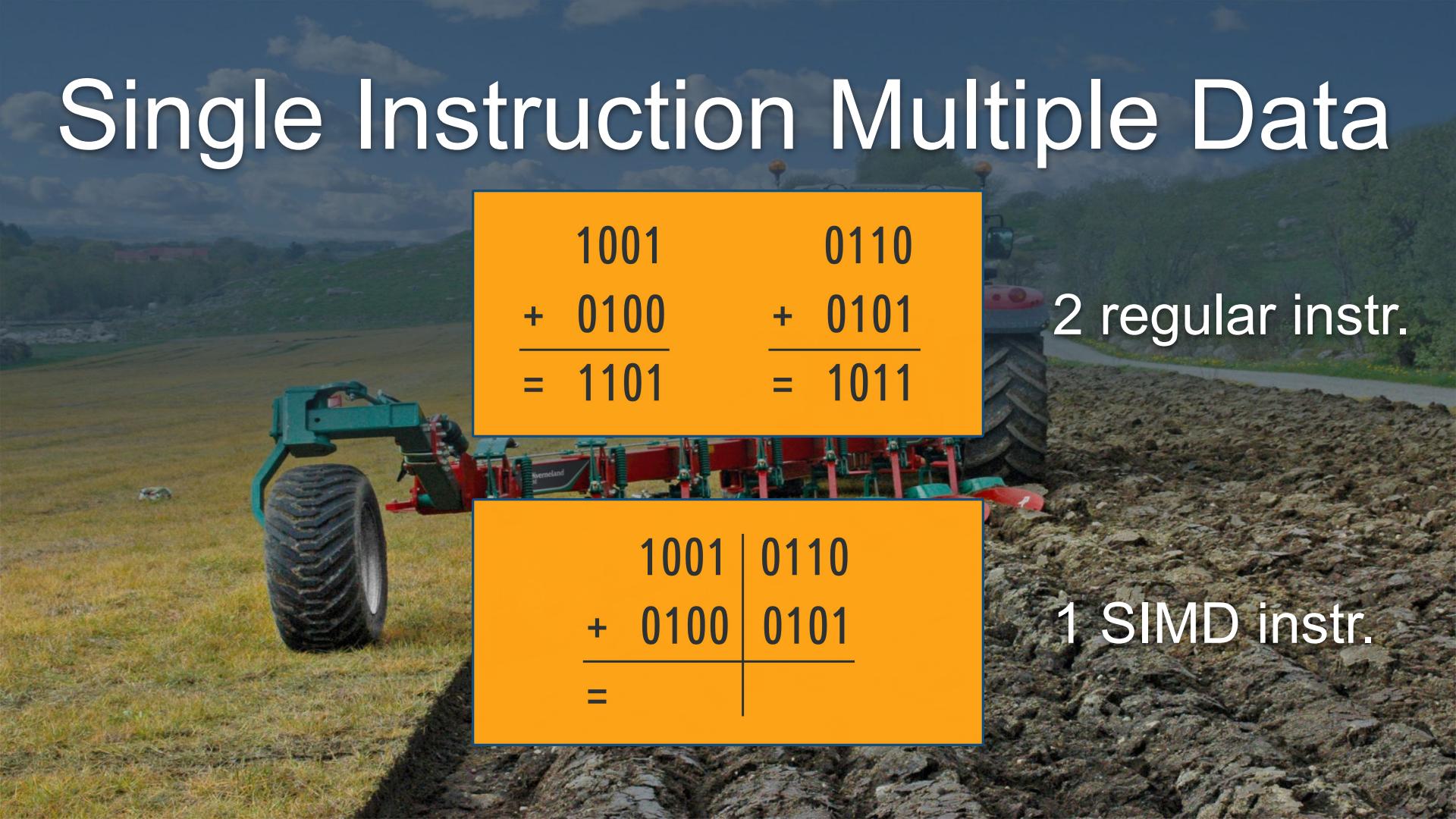
# Single Instruction Multiple Data

$$\begin{array}{r} 1001 \\ + 0100 \\ \hline = 1101 \end{array} \qquad \begin{array}{r} 0110 \\ + 0101 \\ \hline = 1011 \end{array}$$

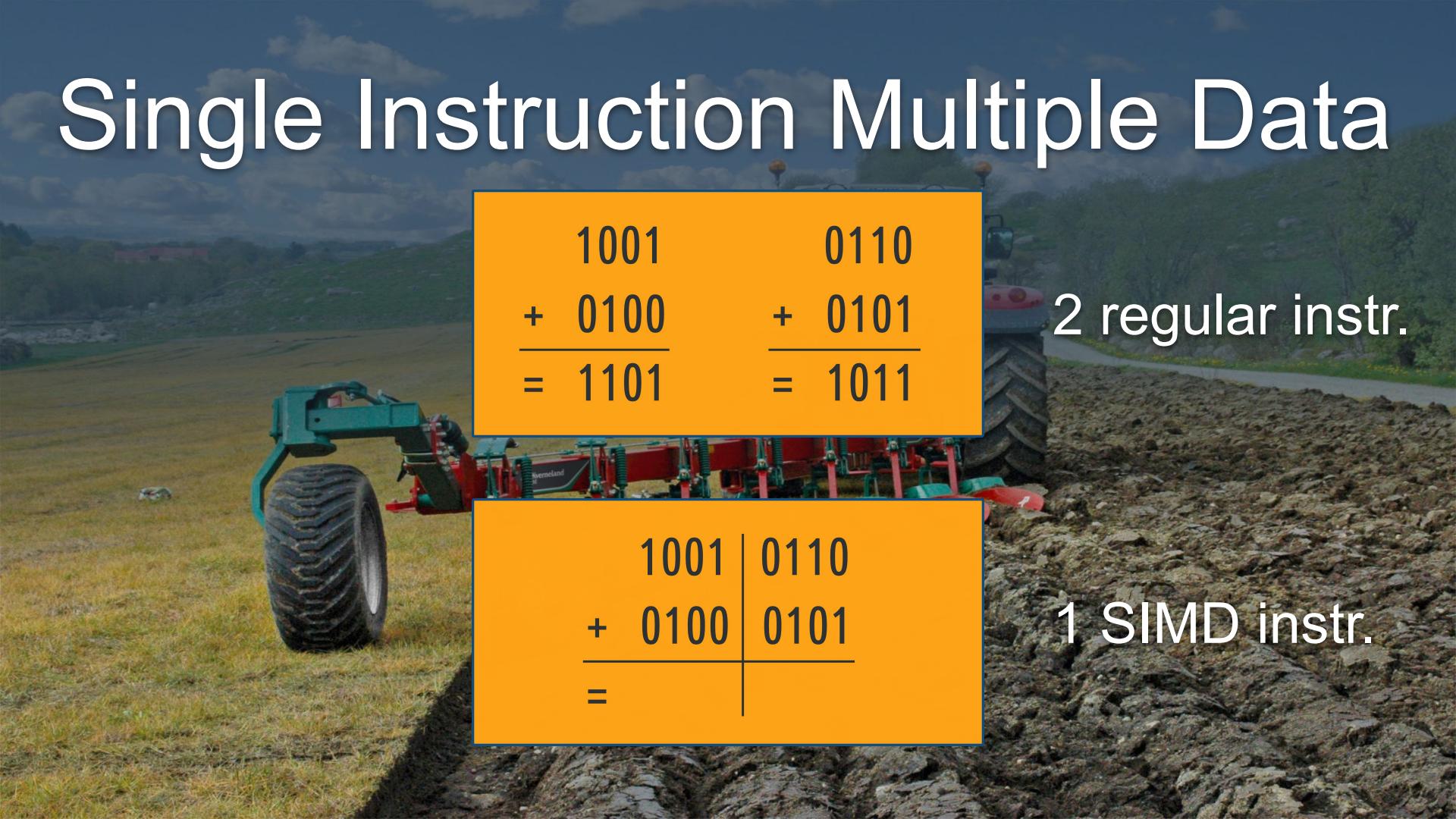
2 regular instr.



# Single Instruction Multiple Data

A photograph of a tractor in a field, shown from a side-front angle. The tractor is plowing a dark, rocky soil. The background shows rolling hills under a blue sky with white clouds.
$$\begin{array}{r} 1001 \\ + 0100 \\ \hline = 1101 \end{array} \qquad \begin{array}{r} 0110 \\ + 0101 \\ \hline = 1011 \end{array}$$

2 regular instr.

A photograph of a tractor in a field, shown from a side-front angle. The tractor is plowing a dark, rocky soil. The background shows rolling hills under a blue sky with white clouds.
$$\begin{array}{r} 1001 | 0110 \\ + 0100 | 0101 \\ \hline = \end{array}$$

1 SIMD instr.

# Single Instruction Multiple Data

$$\begin{array}{r} 1001 \\ + 0100 \\ \hline = 1101 \end{array} \qquad \begin{array}{r} 0110 \\ + 0101 \\ \hline = 1011 \end{array}$$

2 regular instr.

$$\begin{array}{r} 1001 | 0110 \\ + 0100 | 0101 \\ \hline = 1101 | 1011 \end{array}$$

1 SIMD instr.





# Magritte's Radiative Transfer solver

***“Solve the radiative transfer equation **along a ray**”***

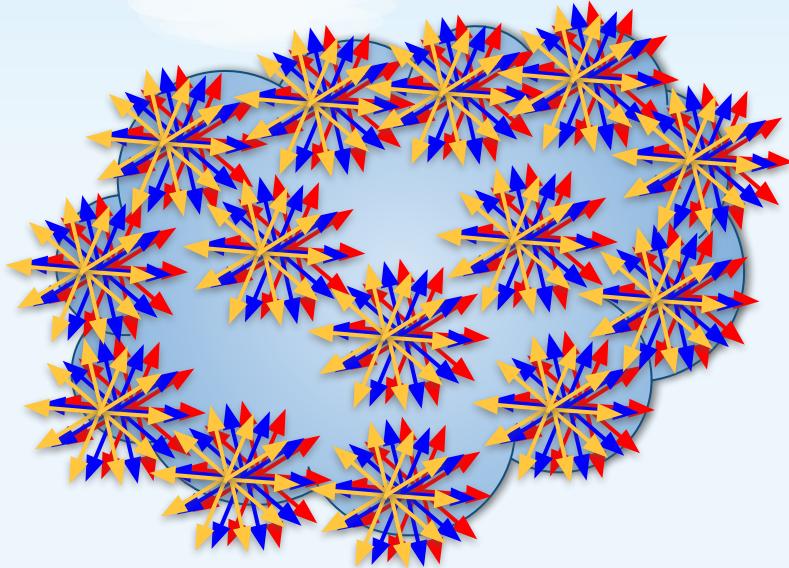
- *for each direction*                    nodes                    (MPI)
- *for each point*                        cores / threads    (OpenMP)
- *for each frequency bin*    vectorisation            (Grid-SIMD)



# Magritte's Radiative Transfer solver

***“Solve the radiative transfer equation along a ray”***

- for each ***direction***
- for each ***point***
- for each ***frequency bin***



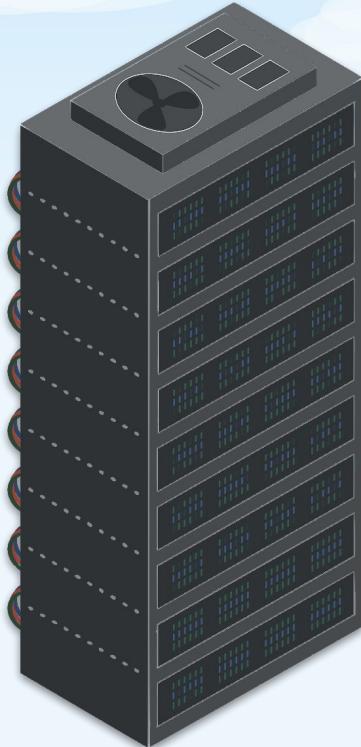


# Magritte's Radiative Transfer solver

***“Solve the radiative transfer equation **along a ray**”***

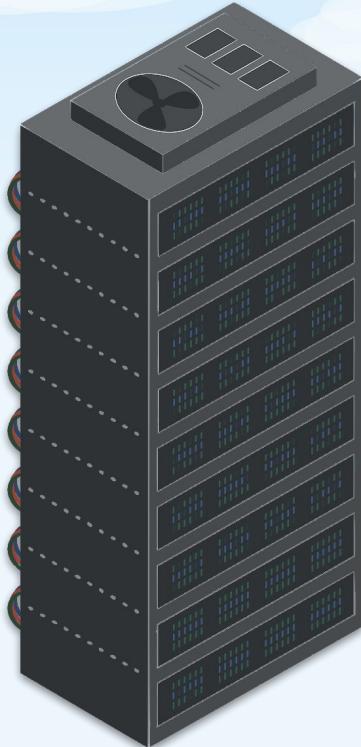
- *for each direction*                    nodes                    (MPI)
- *for each point*                        cores / threads    (OpenMP)
- *for each frequency bin*    vectorisation            (Grid-SIMD)

# Modern high-performance architecture



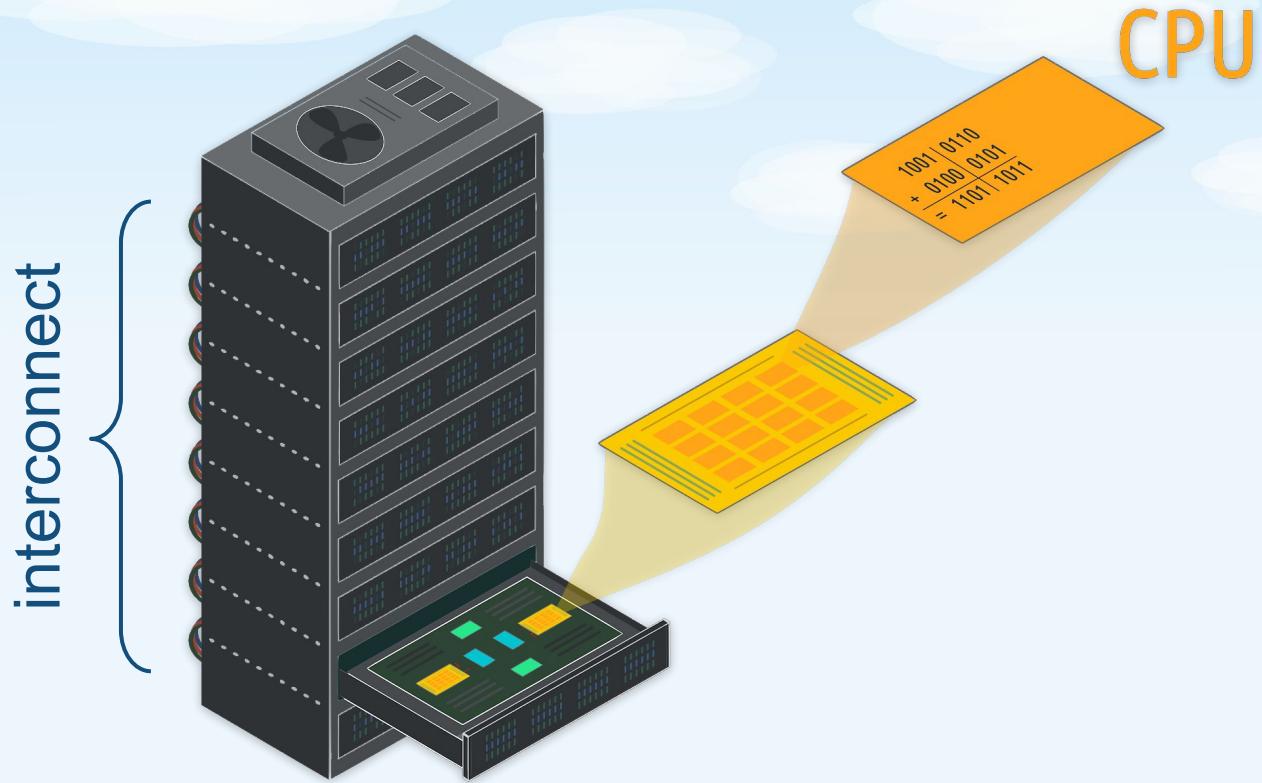
- Layers of **parallelism**
- Hardware **accelerators**

# Modern high-performance architecture

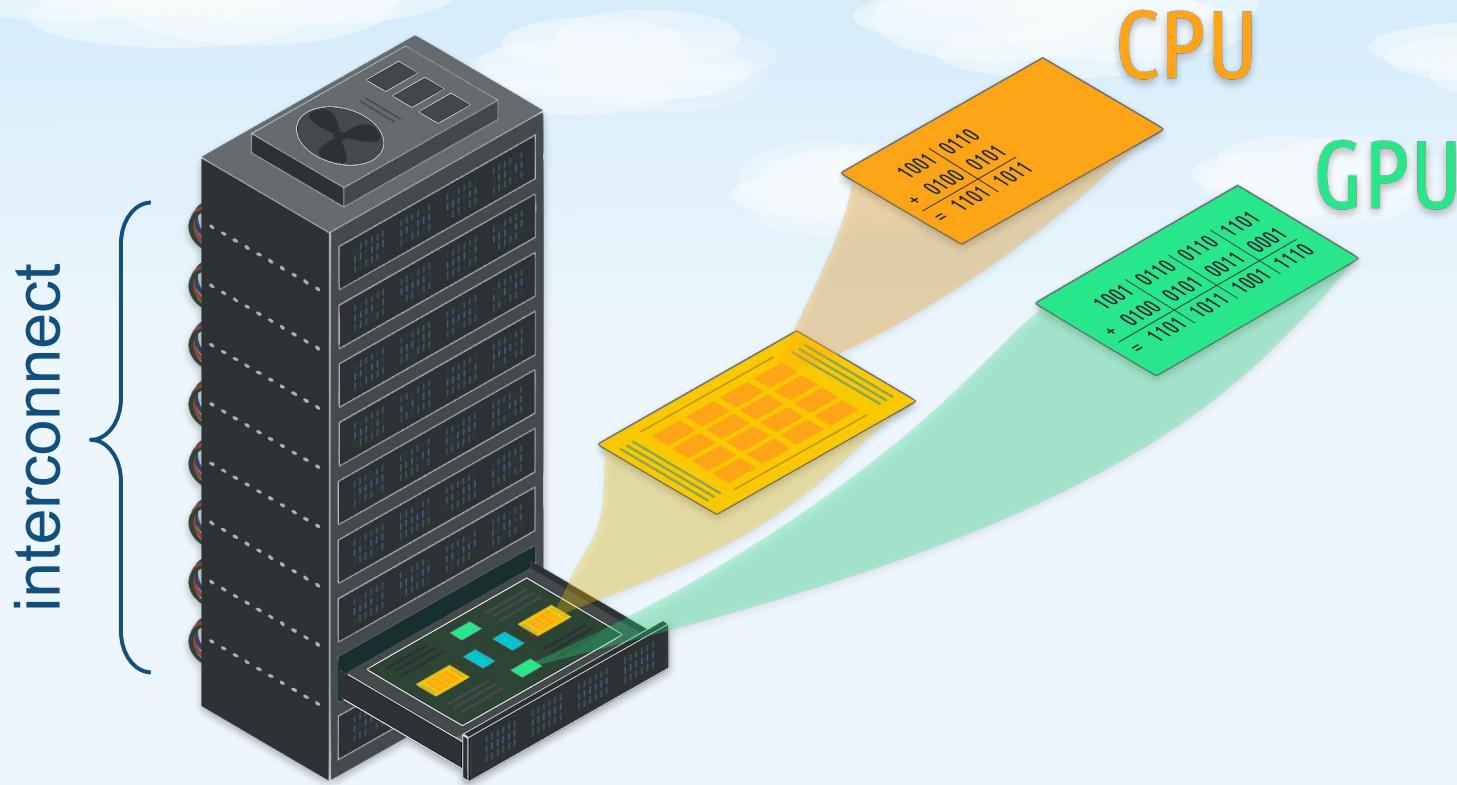


- Layers of **parallelism**
- Hardware **accelerators**

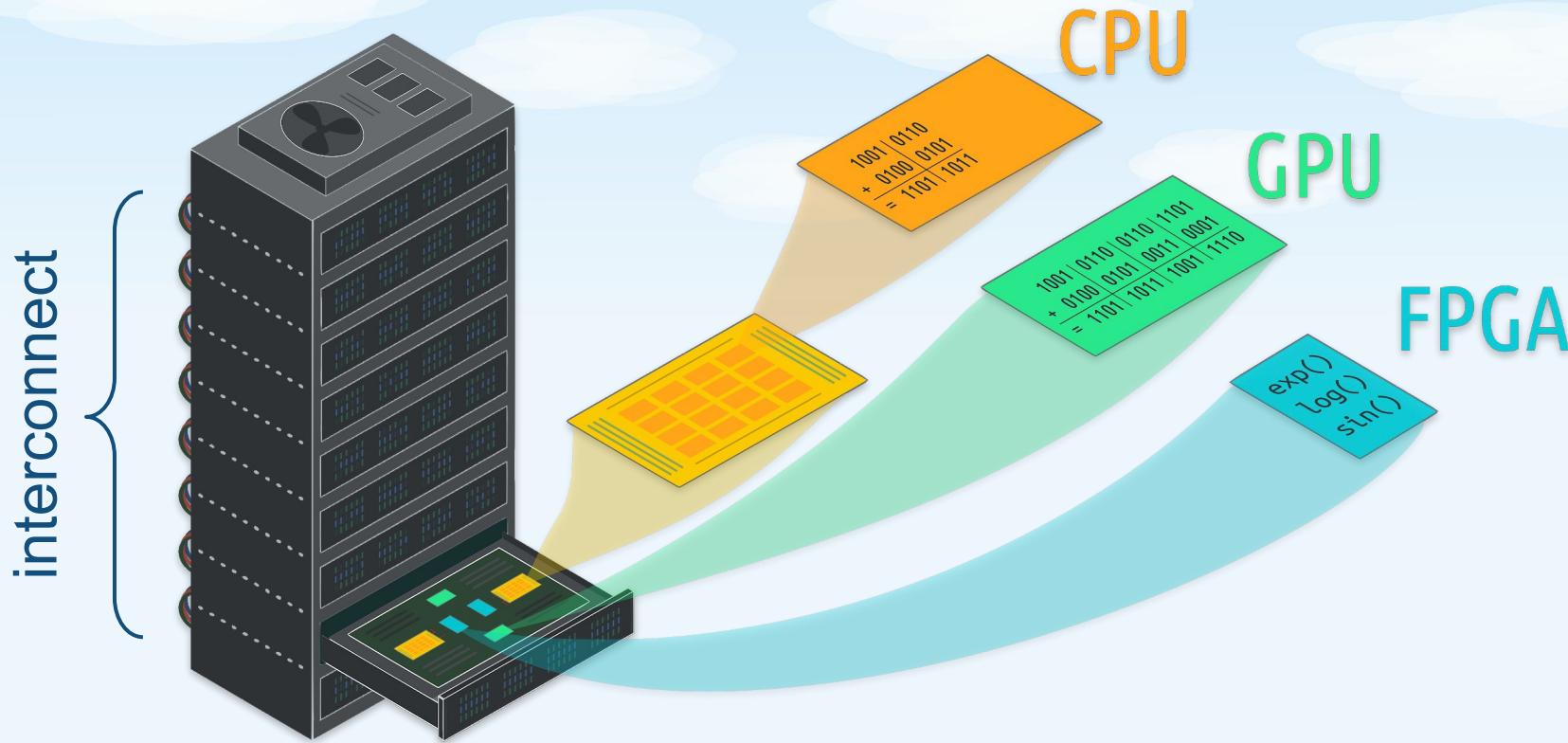
# Modern high-performance architecture



# Modern high-performance architecture



# Modern high-performance architecture





# Magritte's accelerator support

(At least) GPU acceleration for the main solver

	Host (CPU)	Accelerator (GPU)
<b>Setup 1</b>	AMD EPYC Rome	AMD Radeon Instinct MI50
<b>Setup 2</b>	Intel Skylake	Nvidia Tesla V100
<b>Setup 3</b>	Intel Skylake	Nvidia Tesla P100



# Magritte

A **modern** software library for **3D radiative transfer**

RT solver



# Magritte

A **modern** software library for **3D radiative transfer**





# Magritte's mesher

Questions:

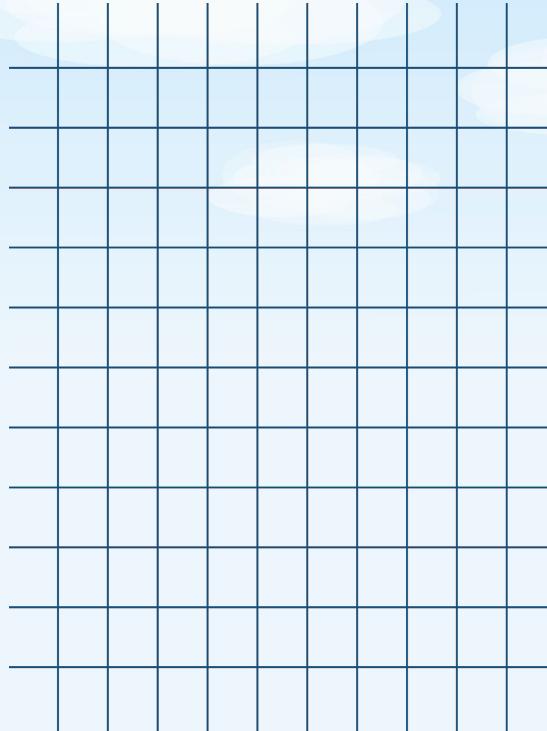
*“How to represent the object on a computer?”*

Note:

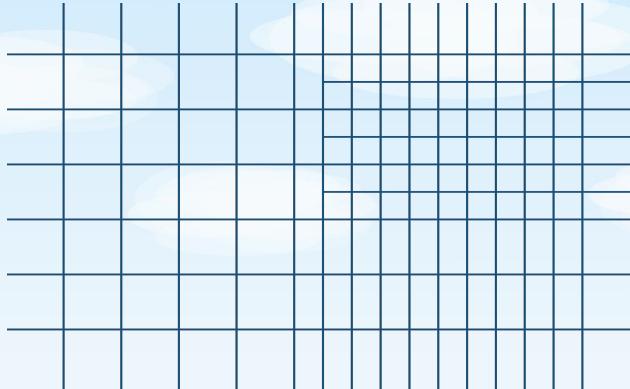
*“The representation is the source of numerical errors!”*



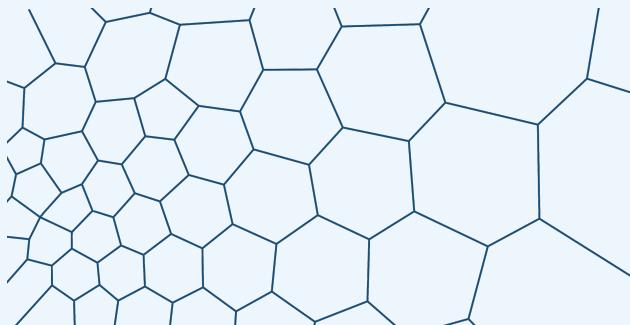
# Magritte's mesher



structured



unstructured



SPH





# Magritte's mesher

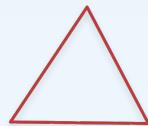
Magritte's **geometric data structure** only contains

- cell / point locations
- nearest neighbour lists



# Magritte's mesher

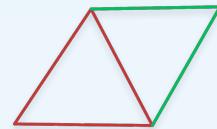
*“Grow the mesh from a **seed** by advancing the **front**”*





# Magritte's mesher

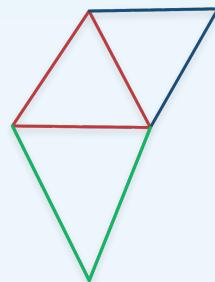
*“Grow the mesh from a **seed** by advancing the **front**”*





# Magritte's mesher

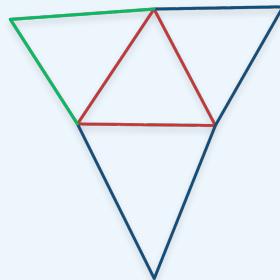
*“Grow the mesh from a **seed** by advancing the **front**”*





# Magritte's mesher

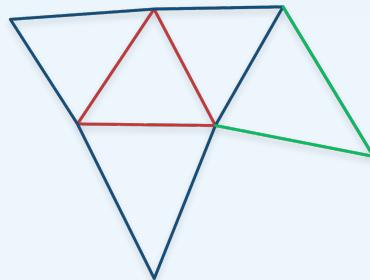
*“Grow the mesh from a **seed** by advancing the **front**”*





# Magritte's mesher

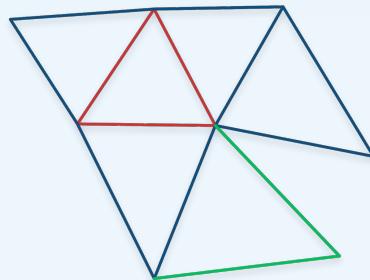
*“Grow the mesh from a **seed** by advancing the **front**”*





# Magritte's mesher

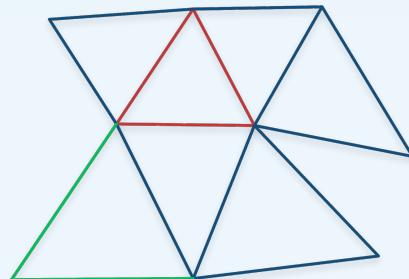
*“Grow the mesh from a **seed** by advancing the **front**”*





# Magritte's mesher

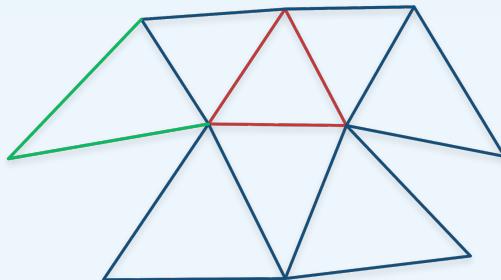
*“Grow the mesh from a **seed** by advancing the **front**”*





# Magritte's mesher

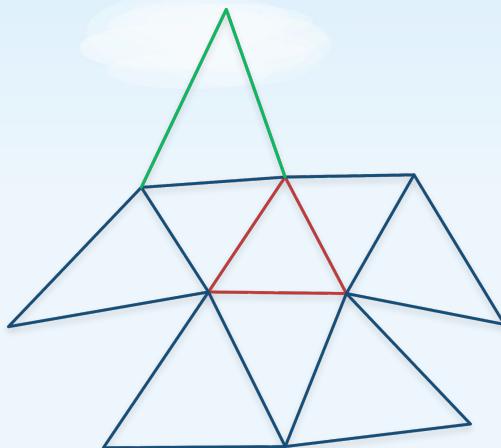
*“Grow the mesh from a **seed** by advancing the **front**”*





# Magritte's mesher

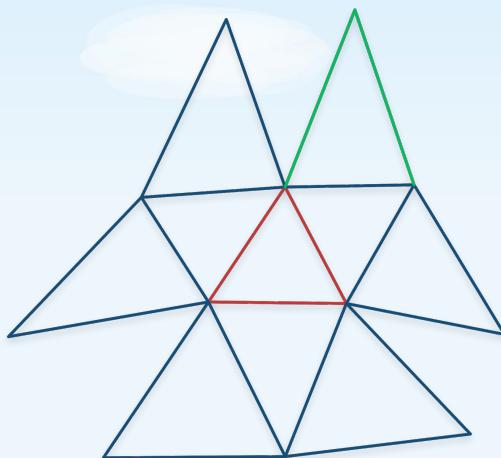
*“Grow the mesh from a **seed** by advancing the **front**”*





# Magritte's mesher

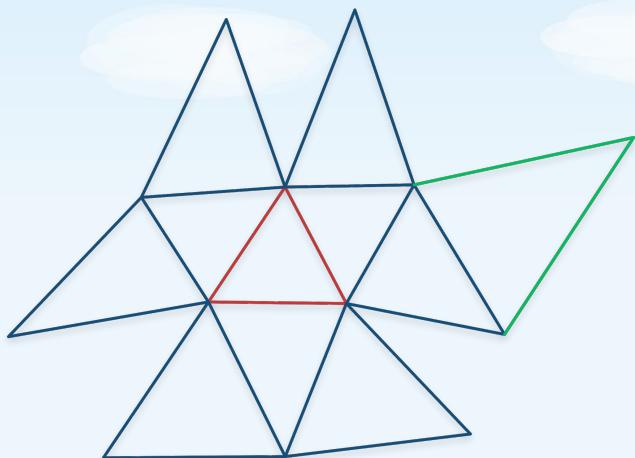
*“Grow the mesh from a **seed** by advancing the **front**”*





# Magritte's mesher

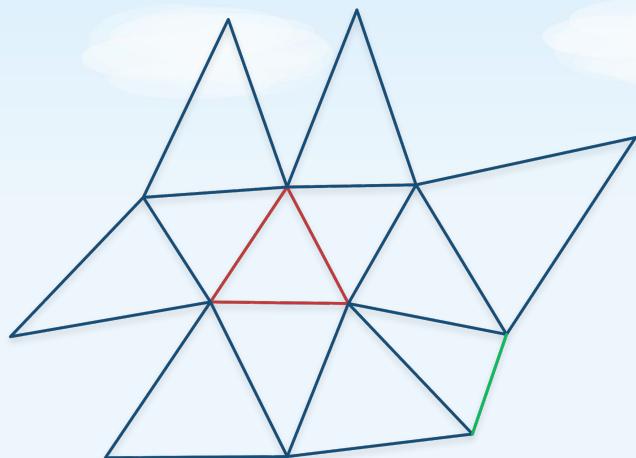
*“Grow the mesh from a **seed** by advancing the **front**”*





# Magritte's mesher

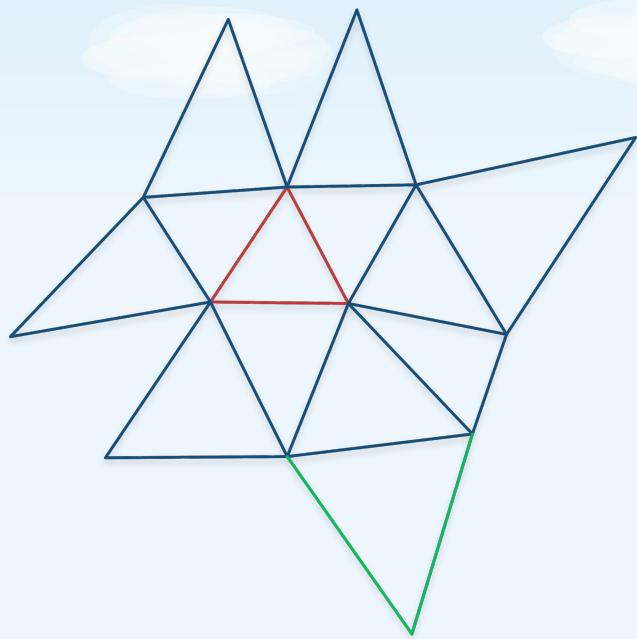
*“Grow the mesh from a **seed** by advancing the **front**”*





# Magritte's mesher

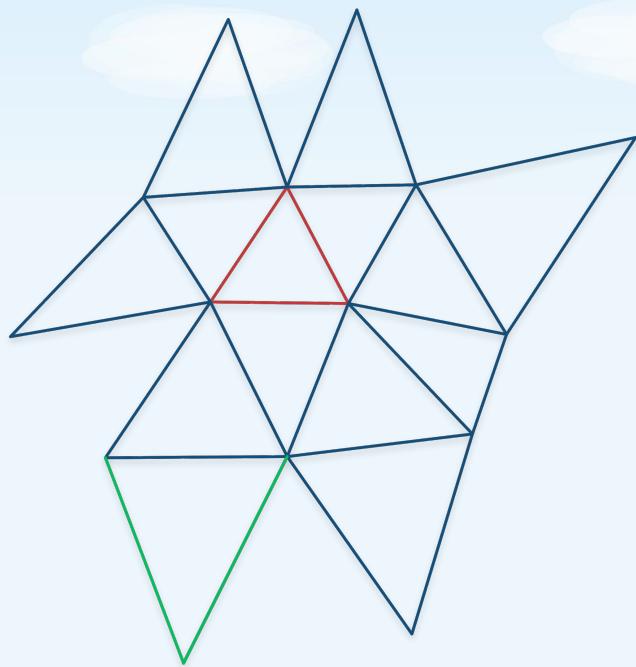
*“Grow the mesh from a **seed** by advancing the **front**”*





# Magritte's mesher

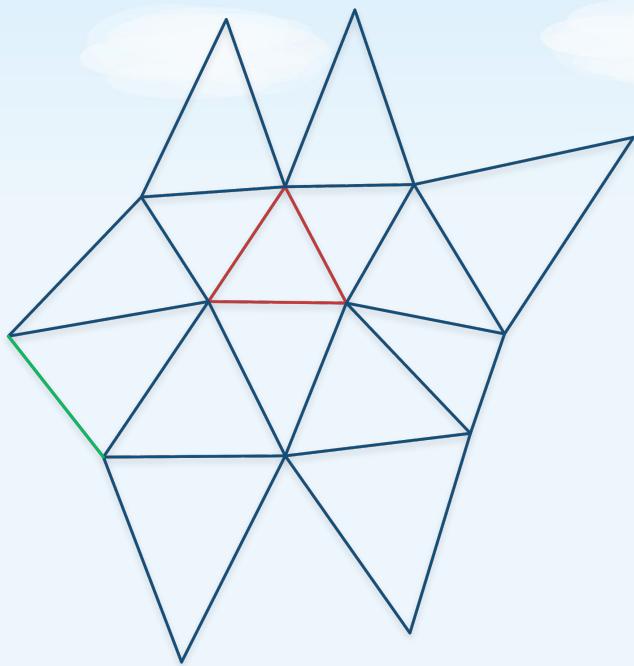
*“Grow the mesh from a **seed** by advancing the **front**”*





# Magritte's mesher

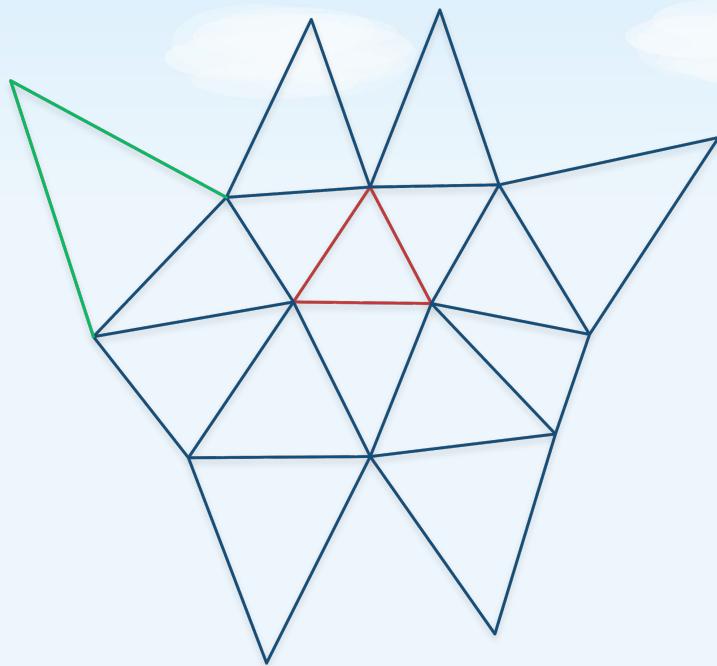
*“Grow the mesh from a **seed** by advancing the **front**”*





# Magritte's mesher

*“Grow the mesh from a **seed** by advancing the **front**”*





# Magritte's mesher

How to **define the shape** of a new triangle?

- “*Define the desired properties as a **heuristic***”
- “*Determine the triangle by **optimising** the heuristic*”



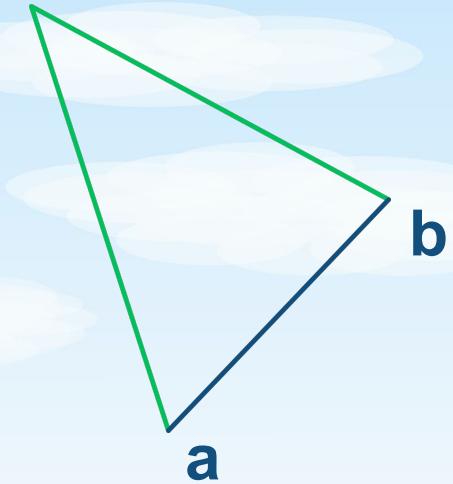
# Magritte's mesher

Example



# Magritte's mesher

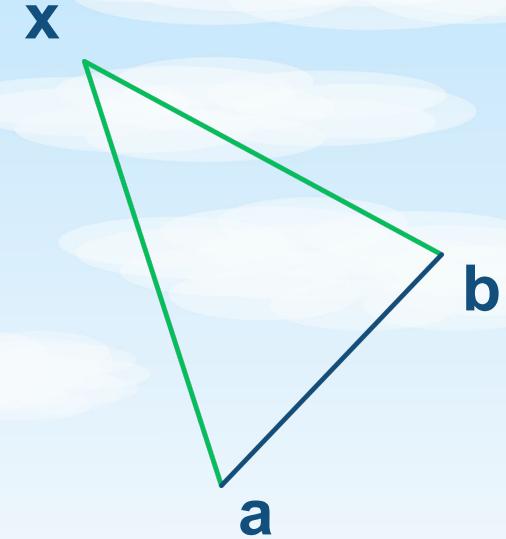
Example





# Magritte's mesher

Example





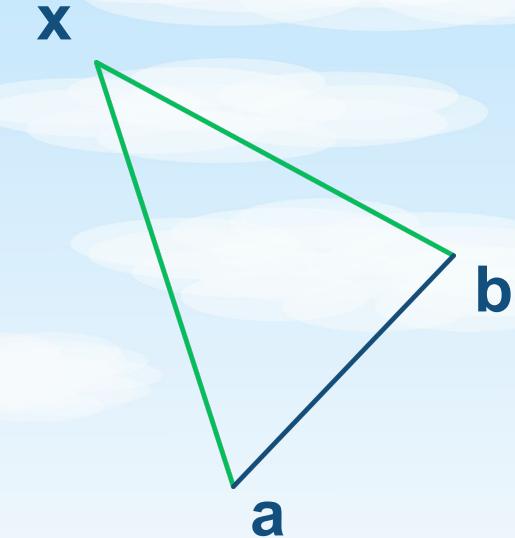
# Magritte's mesher

Example

**Desired property**    edge length  $\approx L$

**Heuristic**                  $h(\mathbf{a}, \mathbf{b}, \mathbf{x}) = (|\mathbf{a}-\mathbf{x}| - L)^2 + (|\mathbf{b}-\mathbf{x}| - L)^2$

Find  $\mathbf{x}$  by minimising  $h(\mathbf{a}, \mathbf{b}, \mathbf{x})$  as function of  $\mathbf{x}$





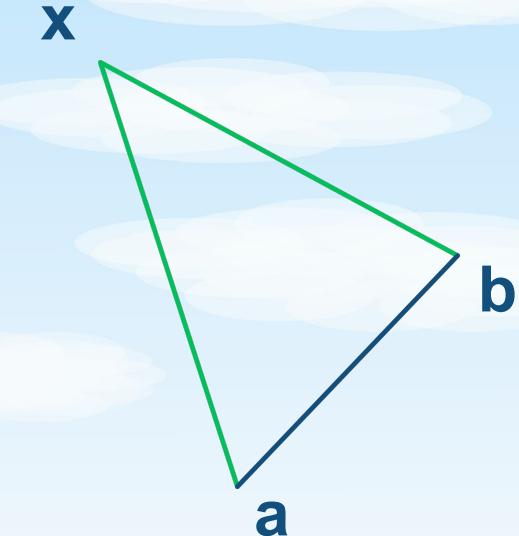
# Magritte's mesher

Example

Desired property    edge length  $\approx L(\mathbf{x})$

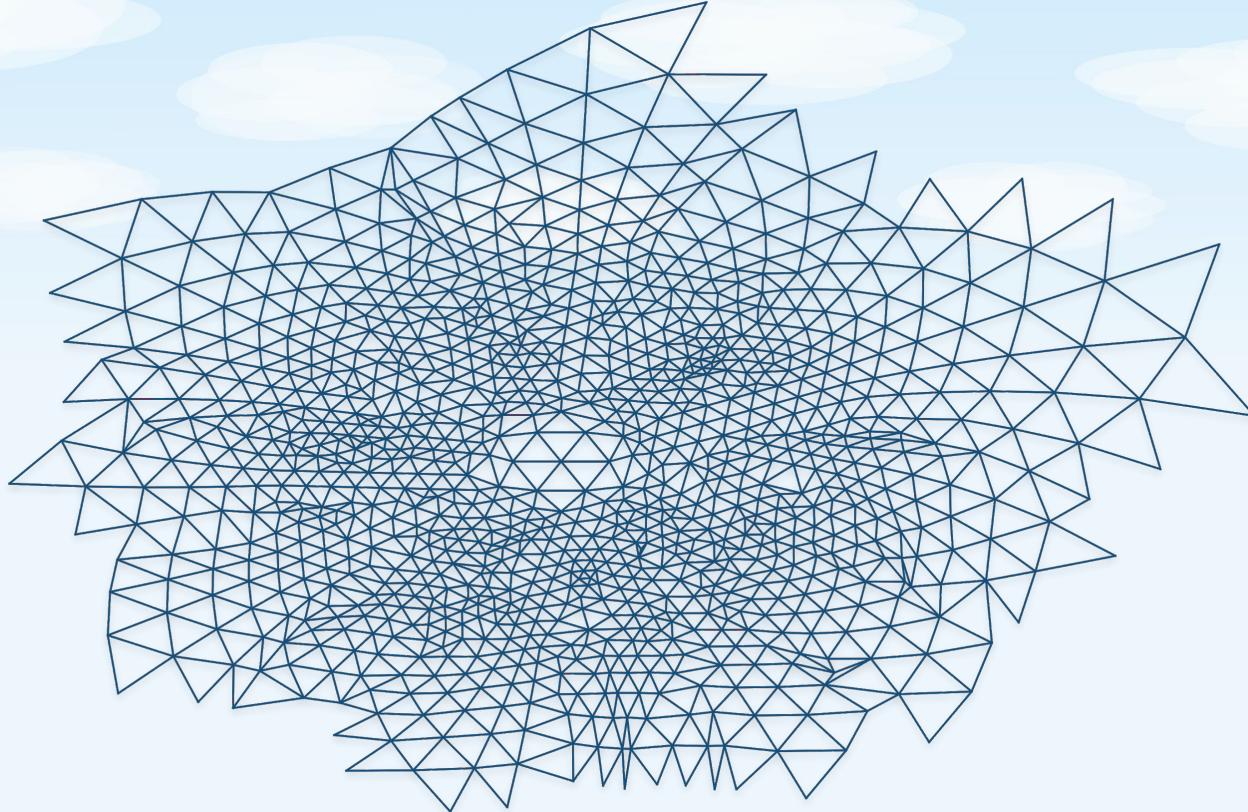
Heuristic                 $h(\mathbf{a}, \mathbf{b}, \mathbf{x}) = (|\mathbf{a}-\mathbf{x}| - L(\mathbf{x}))^2 + (|\mathbf{b}-\mathbf{x}| - L(\mathbf{x}))^2$

Find  $\mathbf{x}$  by minimising  $h(\mathbf{a}, \mathbf{b}, \mathbf{x})$  as function of  $\mathbf{x}$





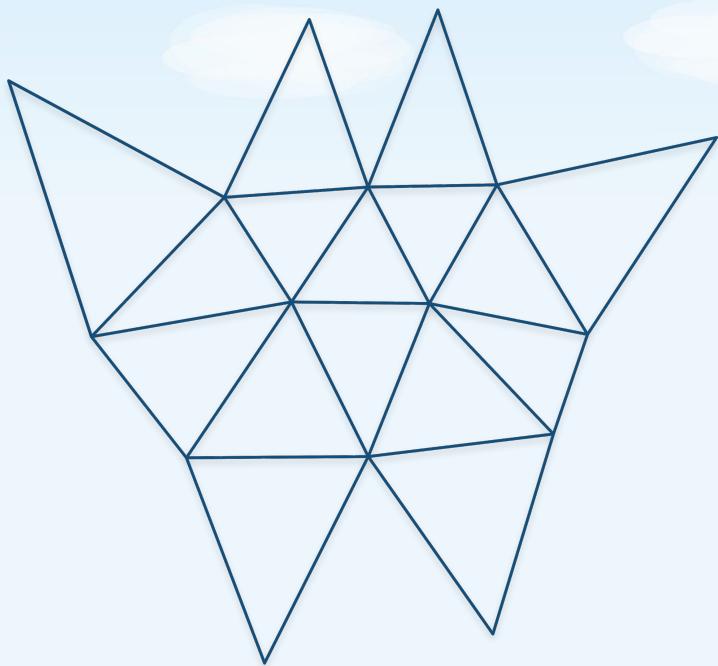
# Magritte's mesher





# Magritte's mesher

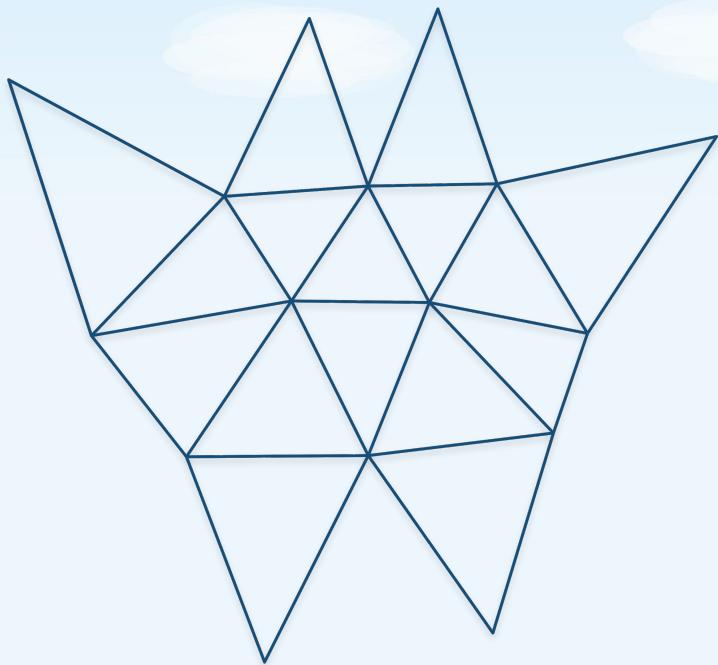
*“How does this help in **constraining numerical errors**? ”*





# Magritte's mesher

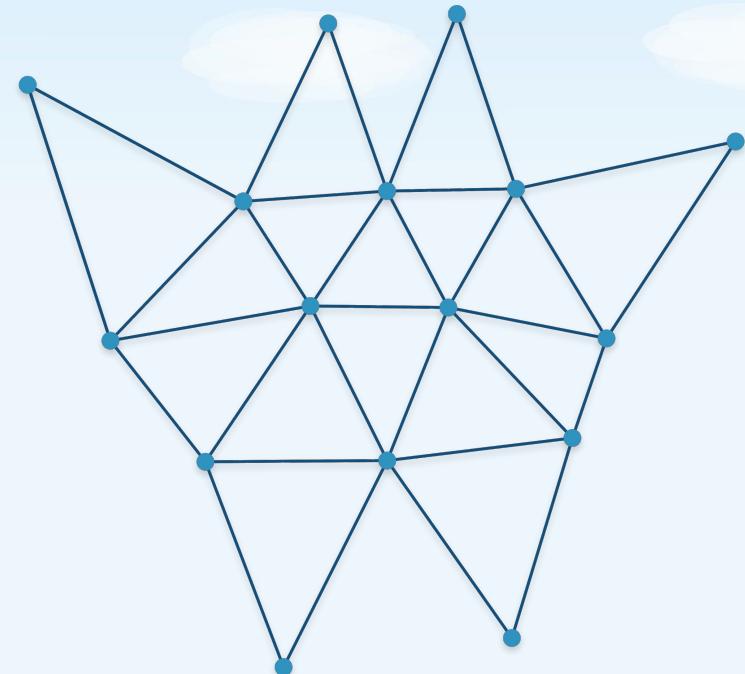
*“Triangulation determines point locations and neighbours”*





# Magritte's mesher

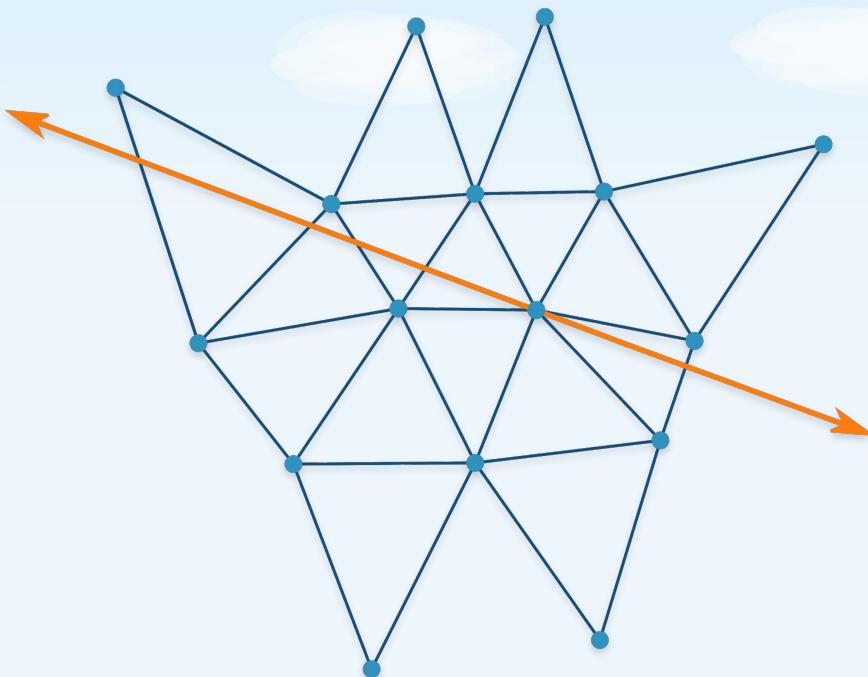
*“Triangulation determines point locations and neighbours”*





# Magritte's mesher

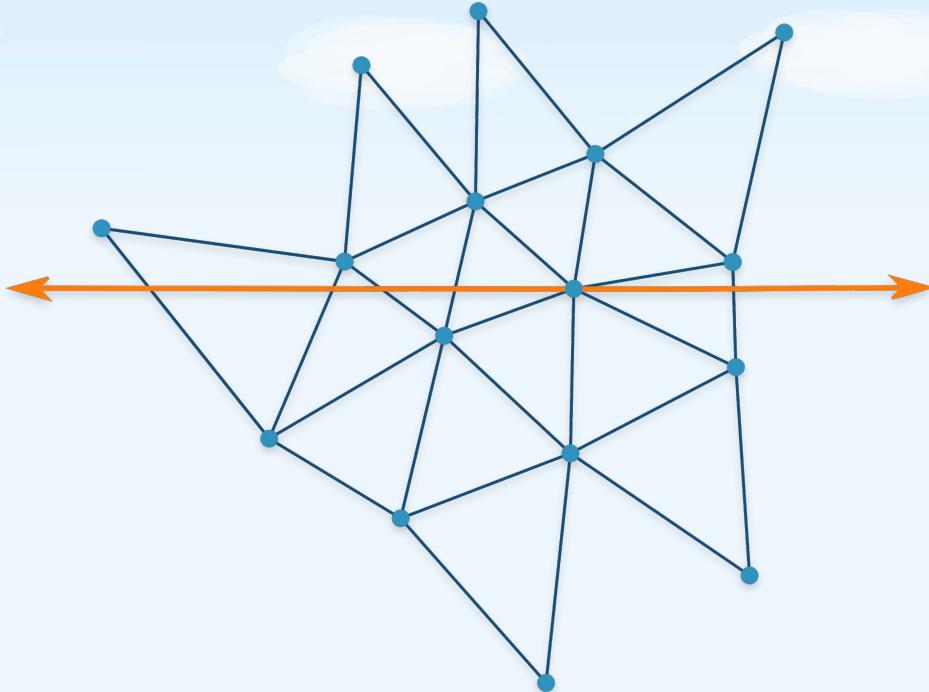
*"Triangulation determines ray **step size**  $\leq$  **edge length**"*





# Magritte's mesher

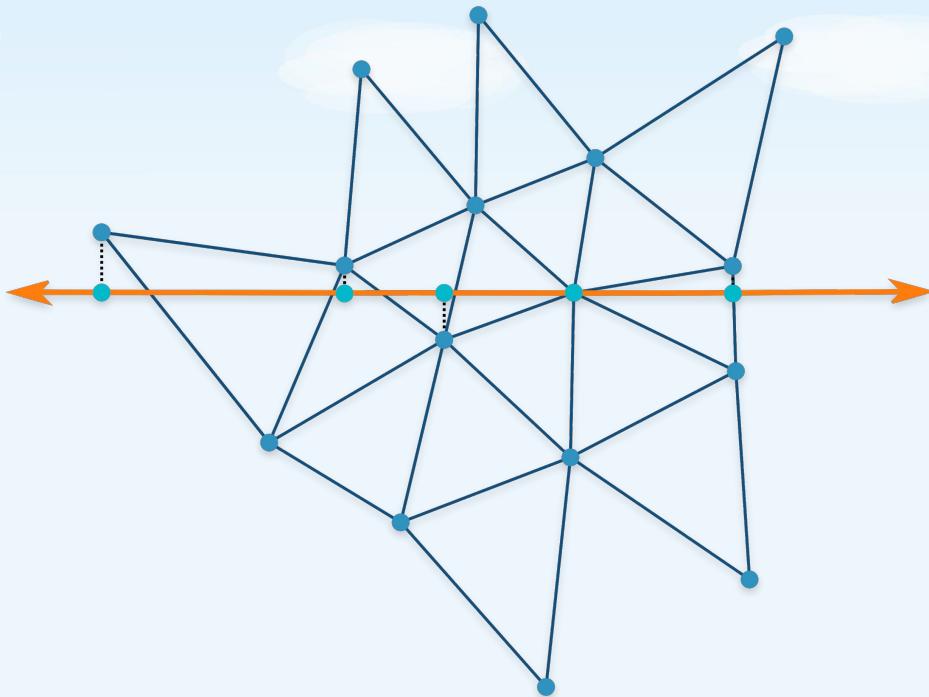
*"Triangulation determines ray **step size**  $\leq$  **edge length**"*





# Magritte's mesher

*"Triangulation determines ray **step size**  $\leq$  **edge length**"*





# Magritte

A **modern** software library for **3D radiative transfer**





# Magritte

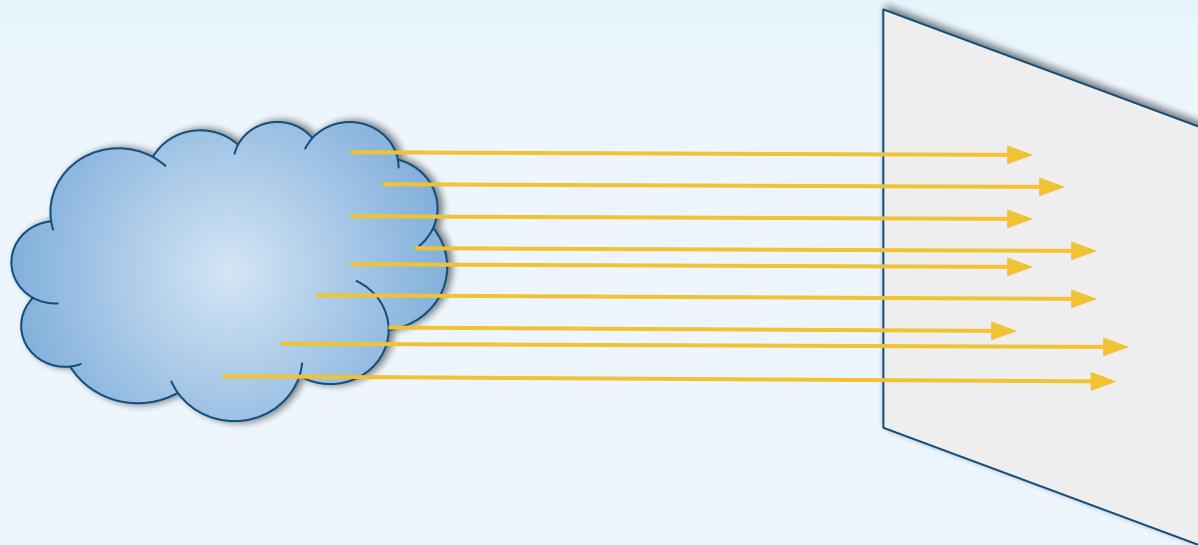
A **modern** software library for **3D radiative transfer**





# Magritte's imager

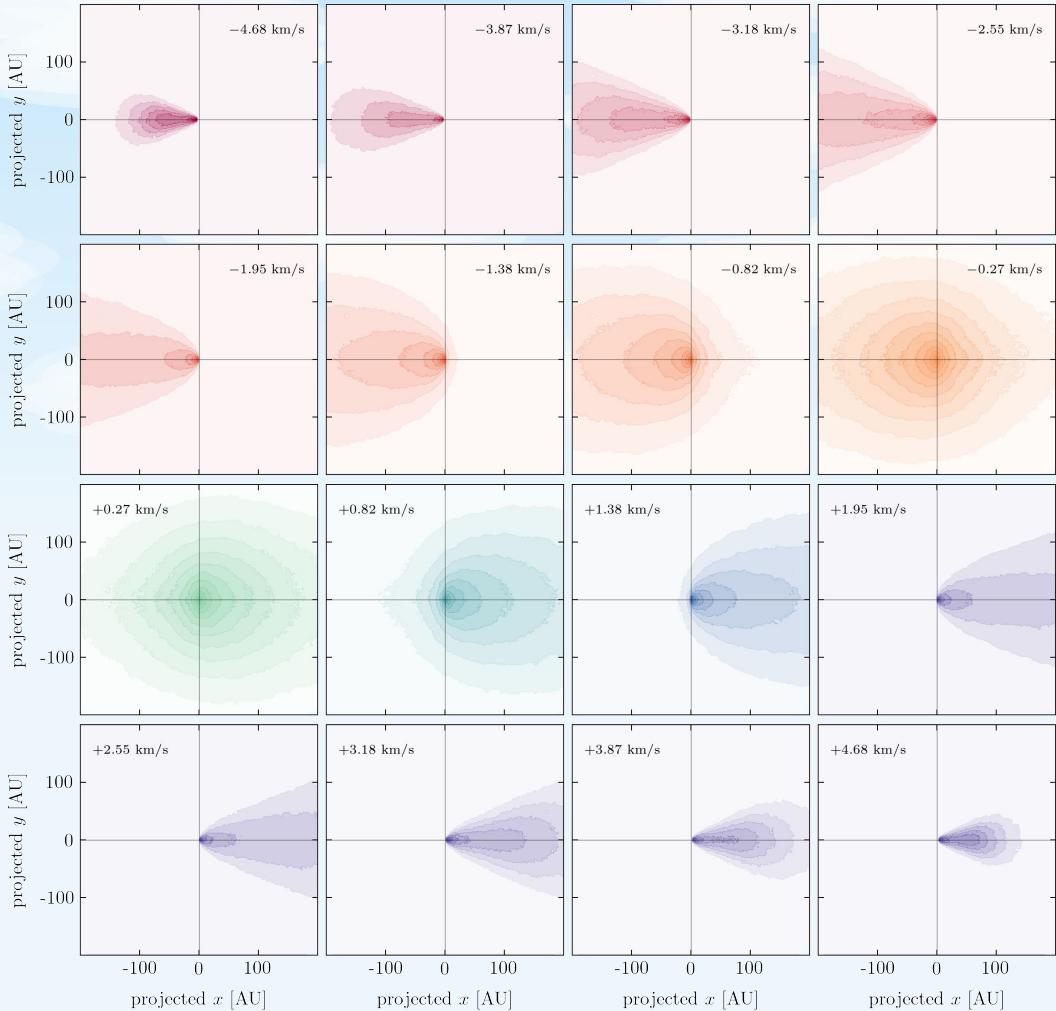
***“Collect all radiation that leaves the model in a certain direction and project it on a screen”***





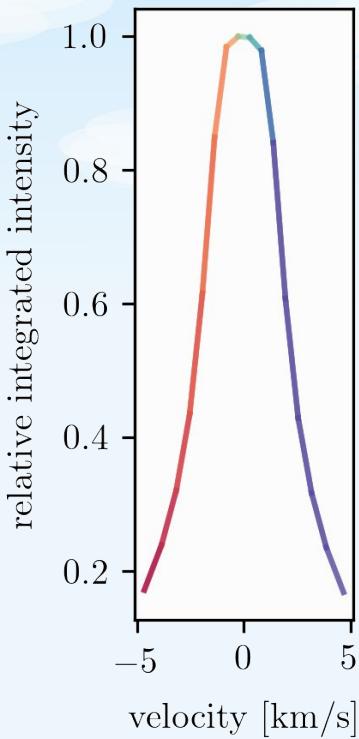
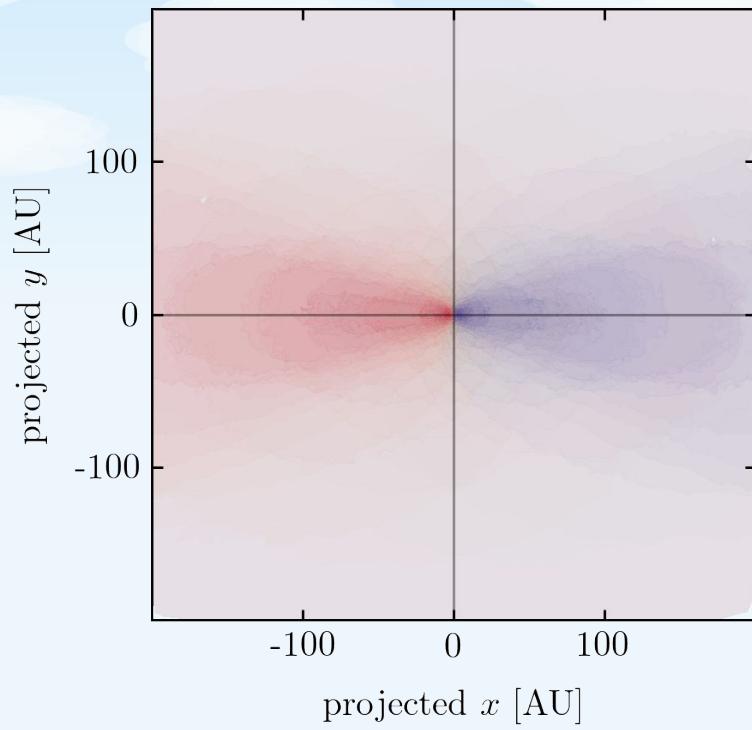
# Magritte

## Channel maps of a Keplerian Disk (CO $J=1-0$ line)





# Magritte's imager





# Magritte

A **modern** software library for **3D radiative transfer**





# Magritte

A modern software library for 3D radiative transfer

observation

model + error estimate





# Magritte

A **modern** software library for 3D radiative transfer



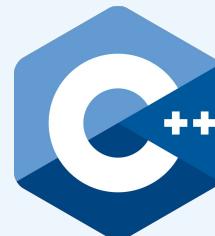


# Magritte

A **modern** software library for 3D radiative transfer



python<sup>TM</sup>





# Magritte's status



# Magritte's status

First paper submitted to MNRAS

## ***Non-LTE atomic and molecular line modelling***

- **Implementation** of line radiative transfer
- Semi-analytic **tests** and cross-code **benchmarks**
- **Application** to a Keplerian disk model



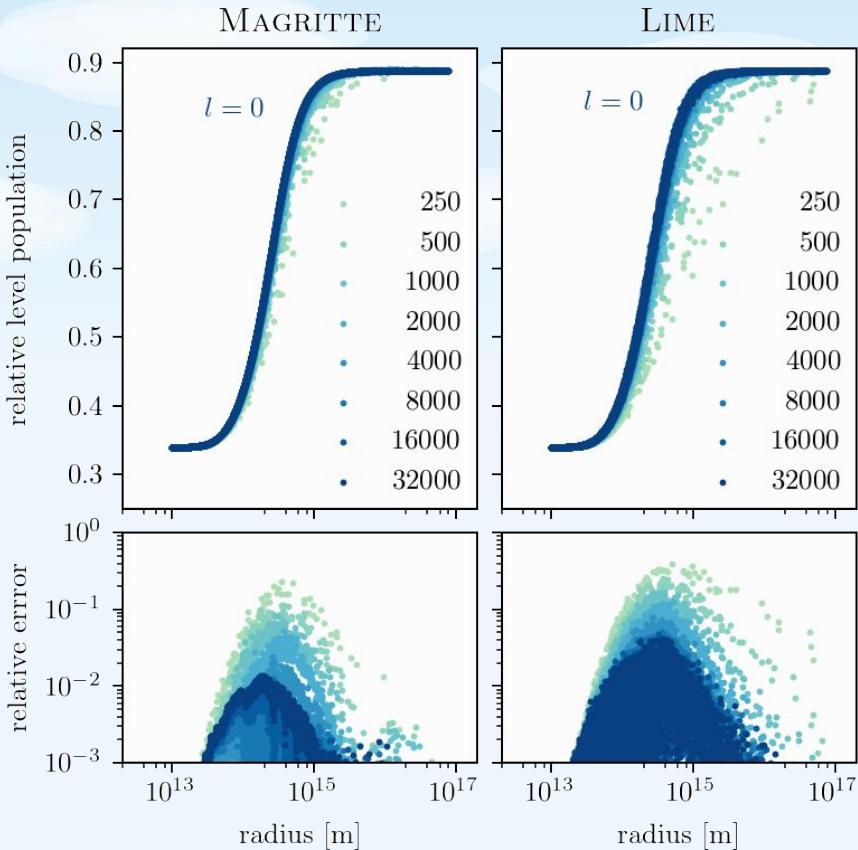
# Magritte vs. Lime (*Brinch & Hogerheijde 2010*)

Benchmark 1a  
(*van Zadelhoff et al. 2002*)

Magritte is

- More accurate
- More precise
- (1.6x faster)

than Lime





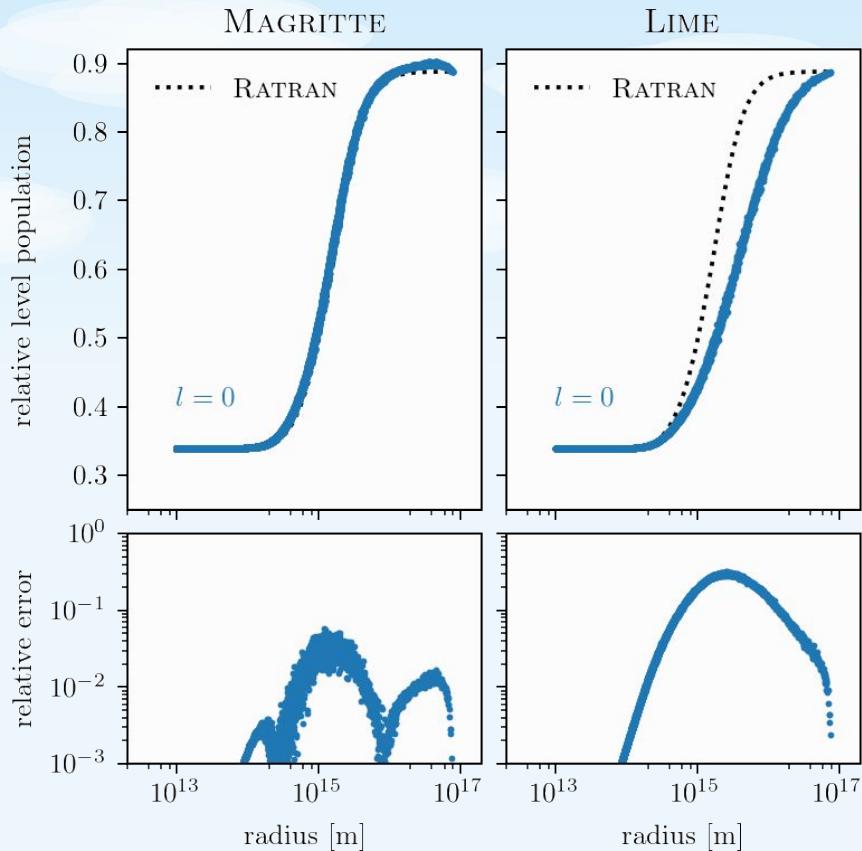
# Magritte vs. Lime (*Brinch & Hogerheijde 2010*)

Benchmark 1b  
(*van Zadelhoff et al. 2002*)

Magritte is

- More accurate
- More precise
- (1.6x faster)

than Lime





# Magritte's status

First paper submitted to MNRAS

**Non-LTE atomic and molecular line modelling**

Second paper in prep.

**Design strategy and performance analysis**



# Magritte



# Magritte





# Magritte



# Thanks!

[github.com/Magritte-code](https://github.com/Magritte-code)



@FredDeCeuster



@FredDeCeuster