Julian Merten

Joint cluster reconstructions on GPUs

Institut für Theoretische Astrophysik
Zentrum für Astronomie
Universität Heidelberg
INAF - Osservatorio Astronomico di Bologna

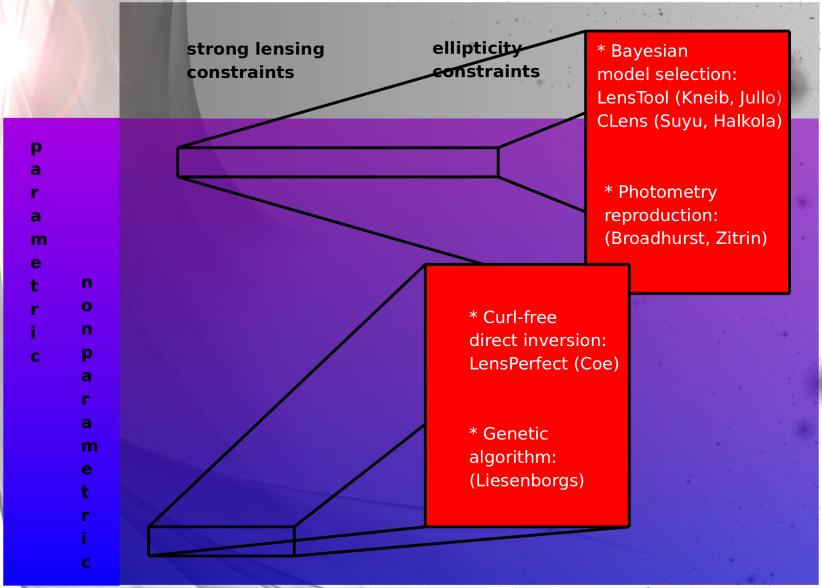
September 20th, 2010

with: Massimo Meneghetti (OABO)
& Matthias Bartelmann (ITA)

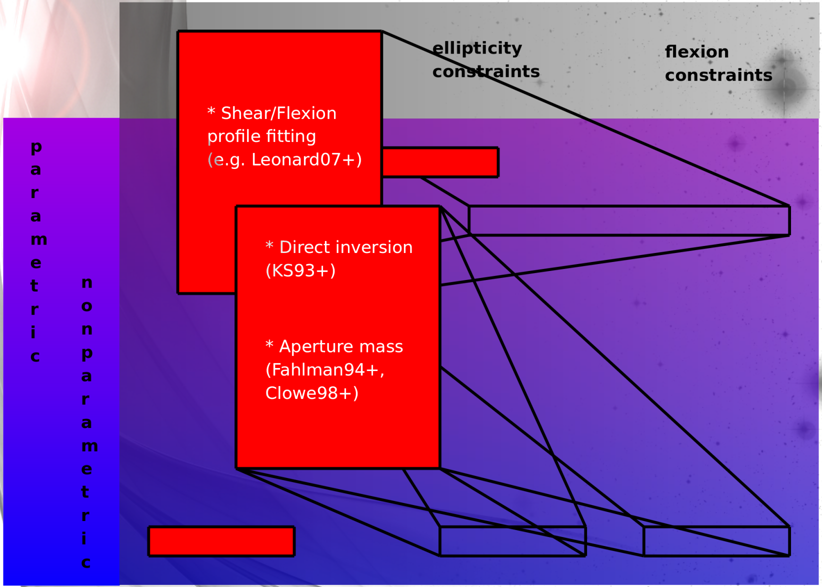
Cluster mass reconstruction zoology

	strong lensing constraints	ellipticity constraints	flexion constraints
parametric			
nonparametric			

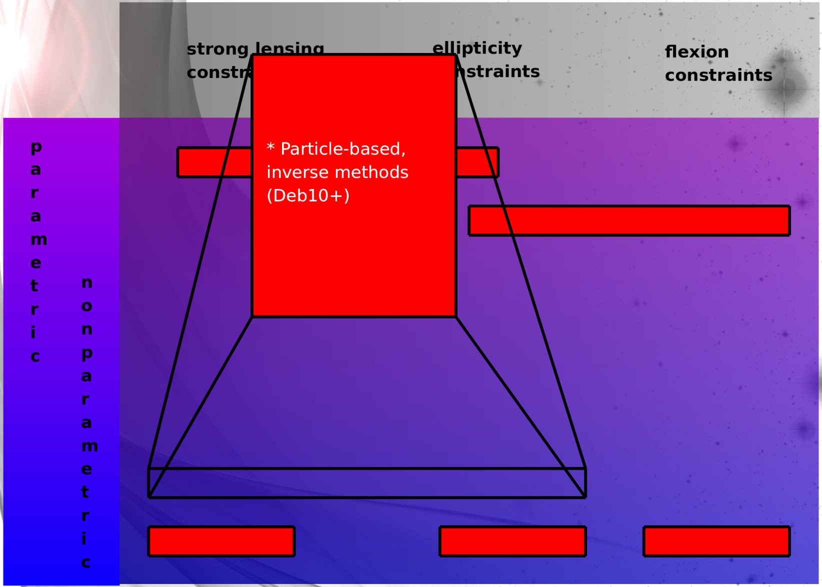
Cluster mass reconstruction zoology



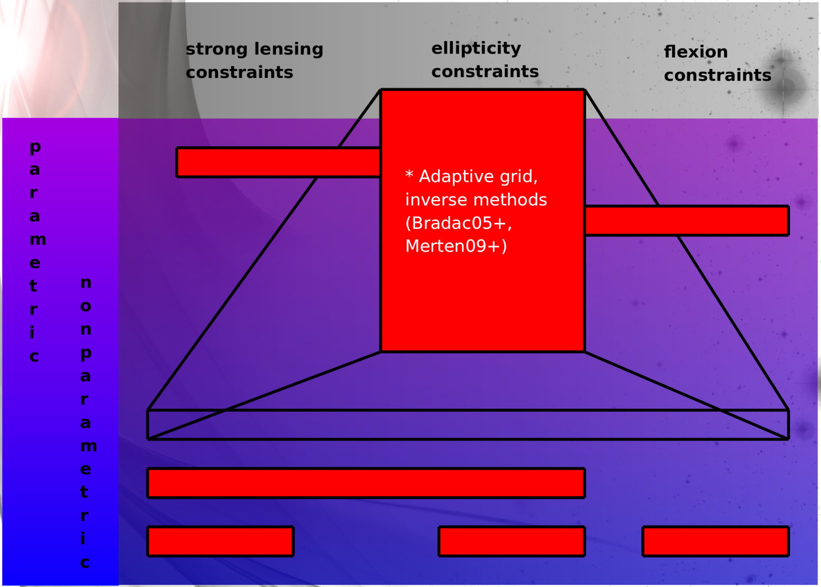
Cluster mass reconstruction zoology



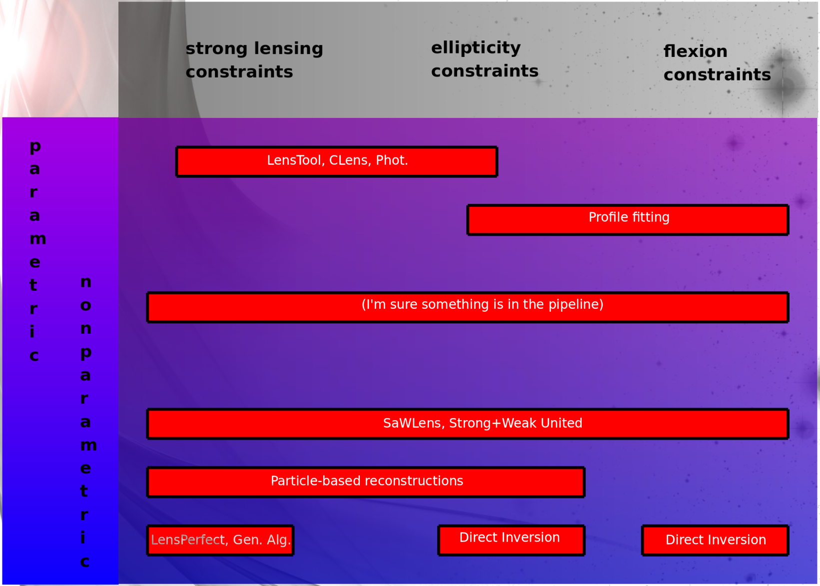
Cluster mass reconstruction zoology



Cluster mass reconstruction zoology



Cluster mass reconstruction zoology



The basic idea of an "inverse" method (Bartelmann96)

Cluster lensing in a box

$$\beta = \theta - \alpha(\theta)$$

$$\partial = \partial_1 + i\partial_2 \qquad \partial^* = \partial_1 - i\partial_2$$

$$\psi \qquad \alpha = \partial\psi$$

$$2\gamma = \partial\partial\psi \qquad 2\kappa = \partial^*\partial\psi$$

$$2F = \partial^*\partial\partial\psi \qquad 2G = \partial\partial\partial\psi$$

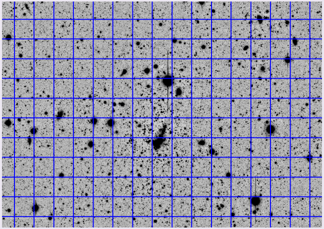
Statistical approach

$$\chi^2(\psi) = \chi_1^2 + \chi_2^2 + \chi_3^2 + \ldots$$

Possible constraints:

- Ellipticities of background sources
- Flexion (JM10 in prep.)
- Multiple image systems (Bradač05+)
- Critical curve estimates (JM09+)

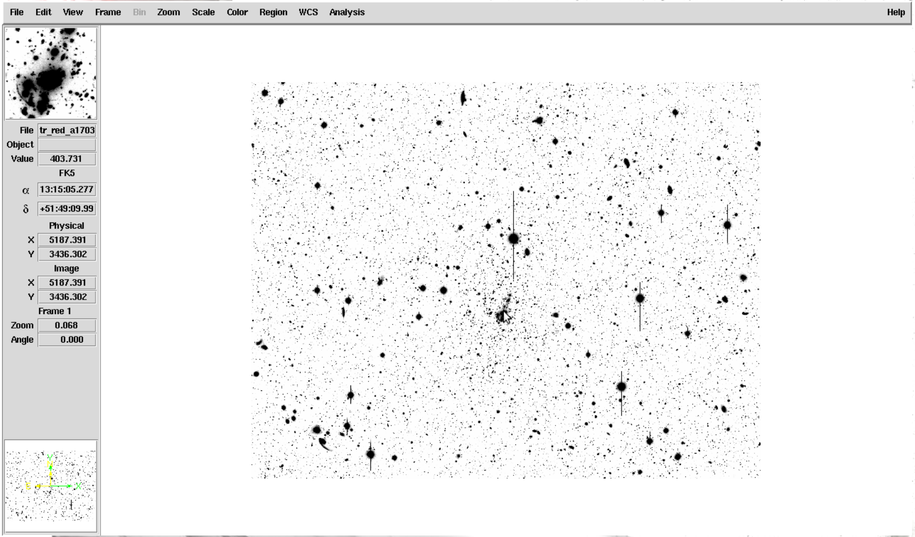
The trick



$$\frac{\partial\chi^2(\psi_k)}{\partial\psi_l} \stackrel{!}{=} 0$$

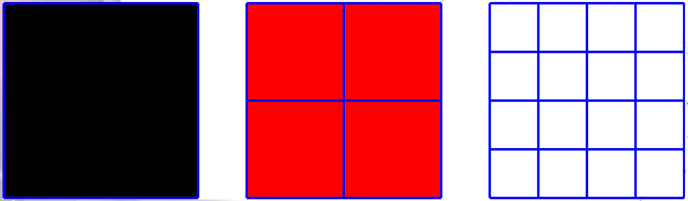
$$\Rightarrow \mathcal{B}_{lk}\psi_k = \mathcal{V}_l$$

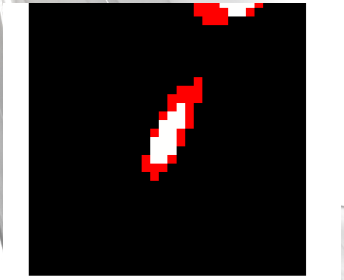
A problem of different scales *(JM10, Bradač09)*

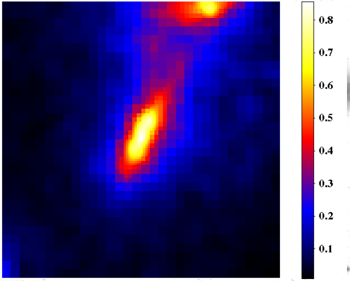


(Abell 1703 in SUBARU r-band)

A problem of different scales *(JM10, Bradač09)*







To go or not to go nonparametric

PROs

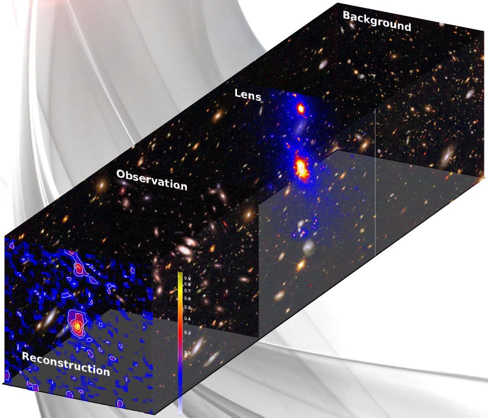
- Easily extendible
  - Lensing constraints (e.g. magnification)
  - Other observables (e.g. kinematics, X-ray)
  - More exotic (Parity, spin-properties of the output field)
- No model assumptions
  - No wrong choices (esp. baryonic component)
  - By construction: sensitive to substructure
  - Accurate characterisation of lens properties

CONs

- Need for rather complex algorithms
  - "Numerical" stability
  - Problem of overfitting
  - Runtime
- Error estimation
  - No analytical approach found yet
  - MCMC clearly out of range (, so far)
  - Resampling via bootstrapping $\Rightarrow$ Runtime

*Realistic lensing simulations: **SkyLens*** *(Meneghetti, JM 08/10)*

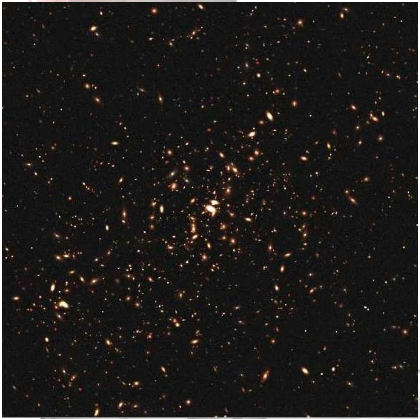
Developers: Massimo Meneghetti, Peter Melchior, Fabio Bellagamba, JM

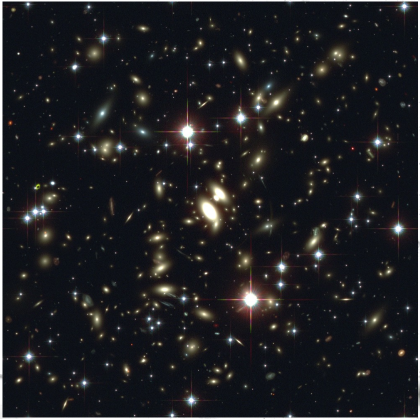


Name	Description
D	aperture diameter
g	detector gain
A_{pix}	pixel area
$F(\lambda)$	used filter
$M(\lambda)$	mirror filter curve
$O(\lambda)$	optics filter curve
$C(\lambda)$	CCD filter curve
FoV	total field-of-view
RON	detector readout-noise
f	flat-field accuracy
a	residual flat-field error
PSF	PSF model
t_{exp}	exposure time
$A(\lambda)$	atmospheric extinction
m_{a}	airmass
SED_{sky}	sky-background emission
SED_{gal}	background population
α	deflection angle map

Realistic lensing simulations: ***SkyLens*** *(Meneghetti, JM 08/10)*

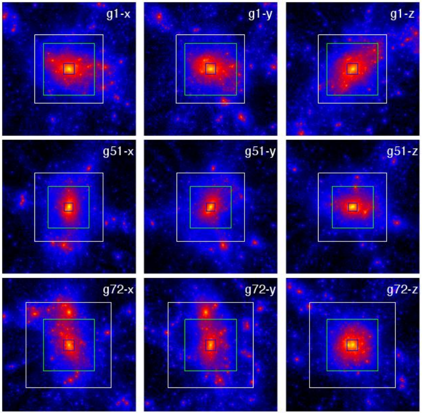
Developers: Massimo Meneghetti, Peter Melchior, Fabio Bellagamba, JM

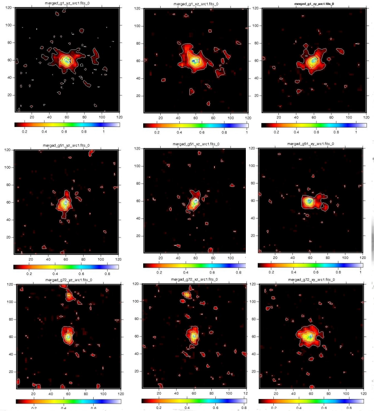




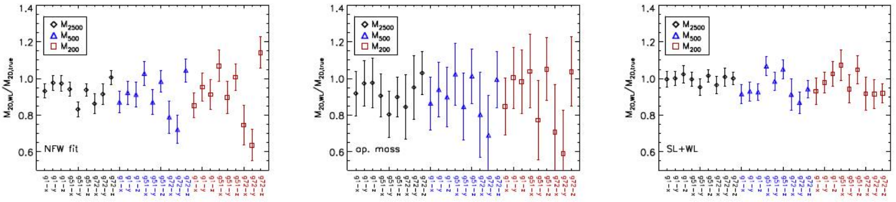
Realistic lensing simulations: **SkyLens** *(Meneghetti, JM 08/10)*

Developers: Massimo Meneghetti, Peter Melchior, Fabio Bellagamba, JM

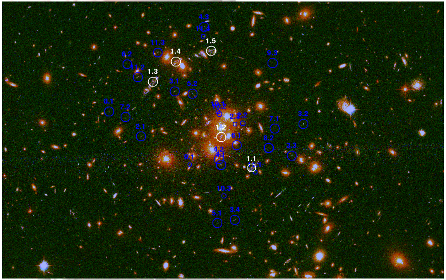


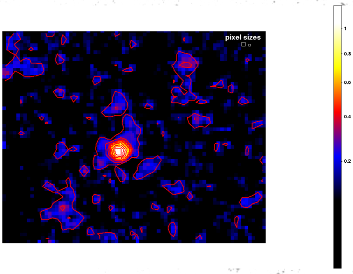


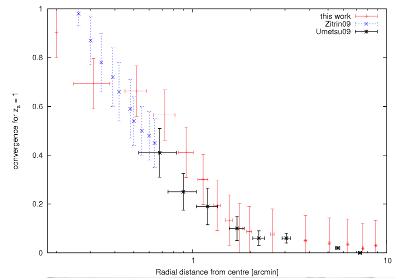
Developers: Massimo Meneghetti, Peter Melchior, Fabio Bellagamba, JM

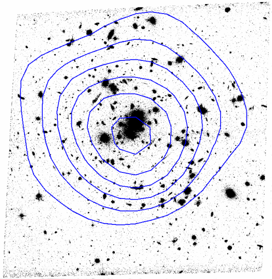


CL0024 (with: T. Broadhurst, A. Zitrin, K. Umetsu)

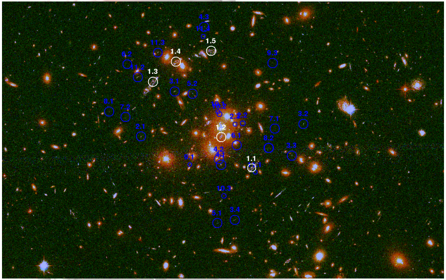


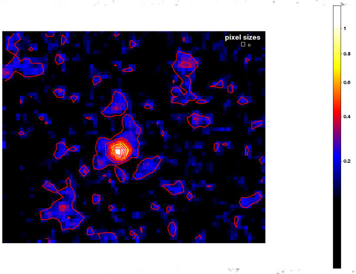


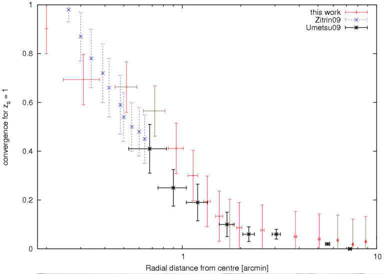


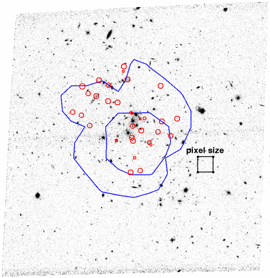


CL0024 *(with: T. Broadhurst, A. Zitrin, K. Umetsu)*



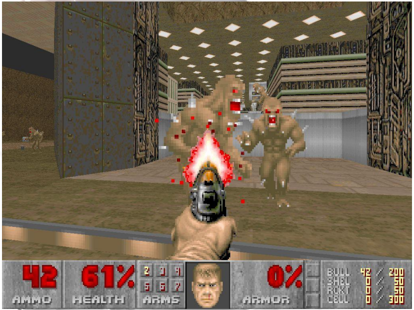






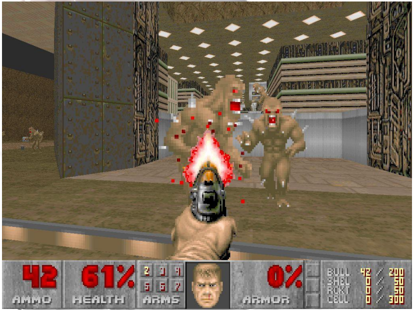
The advent of GPU's...or the art of shooting monsters

1993



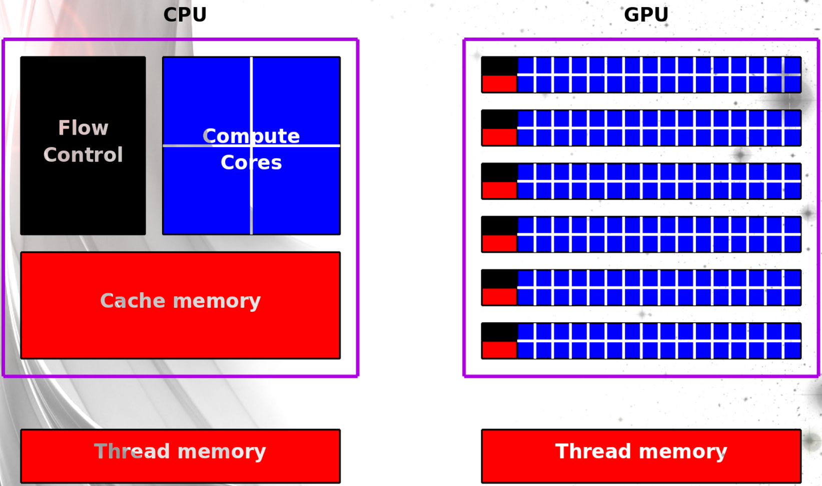
The advent of GPU's...or the art of shooting monsters

1993



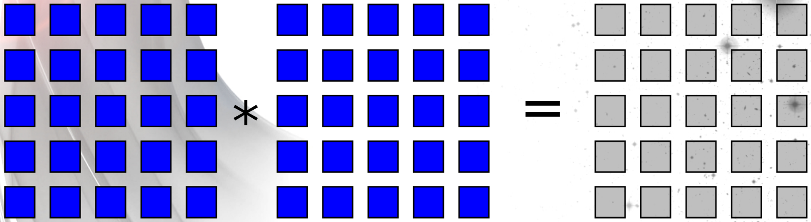
2004

Data-parallel single node GPU Parallelisation



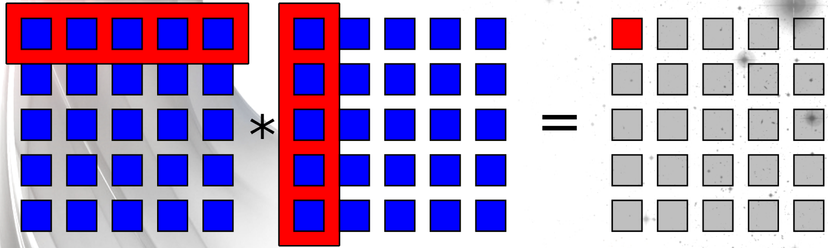
- One single GPU allows for massive parallelisation at a fraction of the cost of a CPU cluster, if problem is suited for ⇒ **Data-parallel**.

*Data-parallel single node **GPU** Parallelisation*



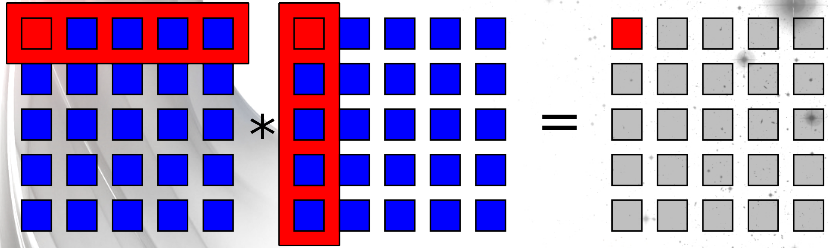
- One single GPU allows for massive parallelisation at a fraction of the cost of a CPU cluster, if problem is suited for ⇒ **Data-parallel**.

Data-parallel single node ***GPU*** *Parallelisation*



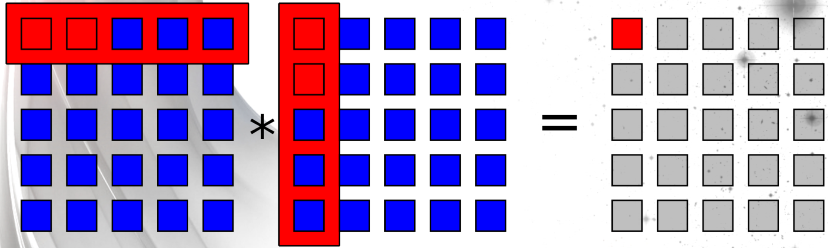
- One single GPU allows for massive parallelisation at a fraction of the cost of a CPU cluster, if problem is suited for ⇒ **Data-parallel**.

*Data-parallel single node **GPU** Parallelisation*



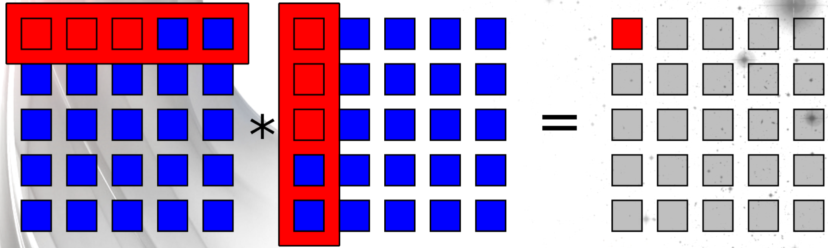
- One single GPU allows for massive parallelisation at a fraction of the cost of a CPU cluster, if problem is suited for ⇒ **Data-parallel**.

*Data-parallel single node **GPU** Parallelisation*



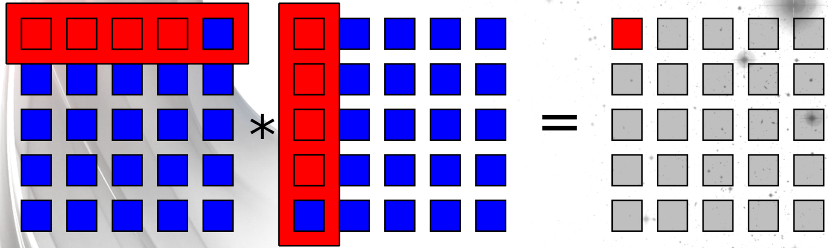
- One single GPU allows for massive parallelisation at a fraction of the cost of a CPU cluster, if problem is suited for ⇒ **Data-parallel**.

*Data-parallel single node **GPU** Parallelisation*



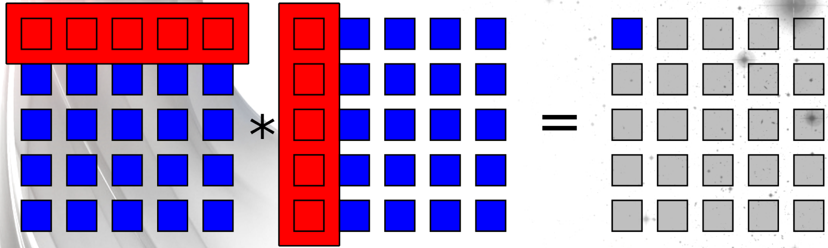
- One single GPU allows for massive parallelisation at a fraction of the cost of a CPU cluster, if problem is suited for ⇒ **Data-parallel**.

*Data-parallel single node **GPU** Parallelisation*



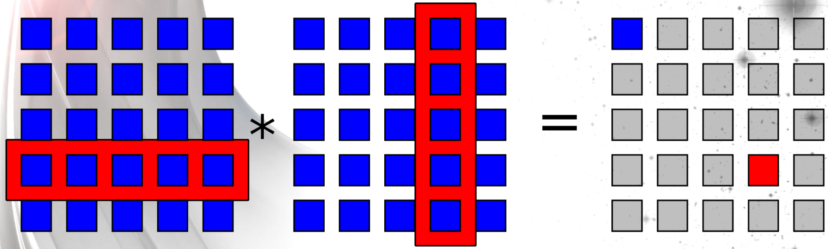
- One single GPU allows for massive parallelisation at a fraction of the cost of a CPU cluster, if problem is suited for ⇒ **Data-parallel**.

*Data-parallel single node **GPU** Parallelisation*



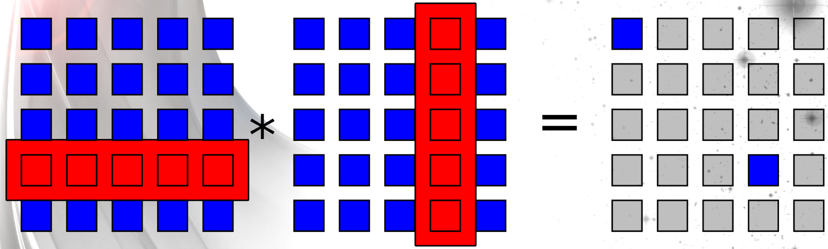
- One single GPU allows for massive parallelisation at a fraction of the cost of a CPU cluster, if problem is suited for ⇒ **Data-parallel**.

*Data-parallel single node **GPU** Parallelisation*



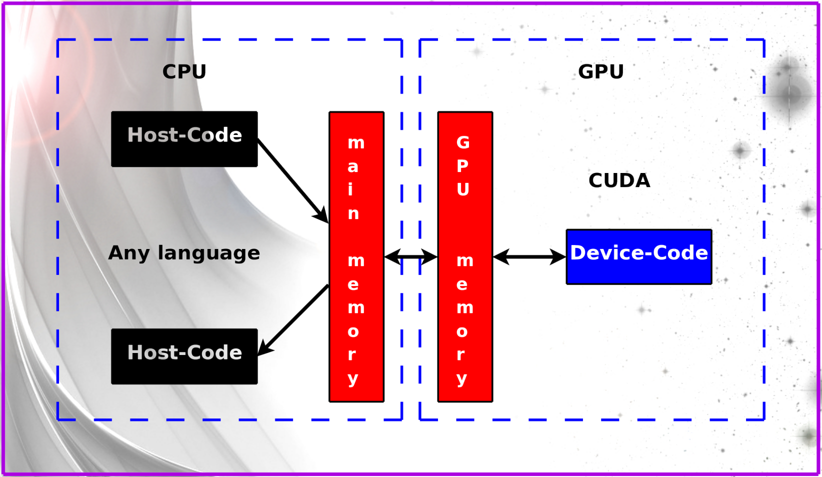
- One single GPU allows for massive parallelisation at a fraction of the cost of a CPU cluster, if problem is suited for ⇒ **Data-parallel**.

Data-parallel single node ***GPU*** *Parallelisation*



- One single GPU allows for massive parallelisation at a fraction of the cost of a CPU cluster, if problem is suited for ⇒ **Data-parallel**.

*Data-parallel single node **GPU** Parallelisation*



- One single GPU allows for massive parallelisation at a fraction of the cost of a CPU cluster, if problem is suited for ⇒ **Data-parallel**.

GPUs in practice *(JM10 in prep.)*

NVIDIA Tesla C1060

- 240 streaming cores
- 4 GB DDR3 GPU memory
- 933 GFLOPS peak performance
- `Fermi` cards out by now, update planned

Speed-up

- Calculate:

$$\mathcal{B}_{lk} = a_i b_j \mathcal{C}_{ij} \mathcal{D}_{ik} \mathcal{E}_{jl}$$

- one-core CPU: `82.3 s`
- 240 core GPU: `1.03 s`

To use or not to use GPU implementations

PROs

- Speed
  - Single-node servers $\sim$ 4 TFLOPS
  - GPU clusters are possible
- Cost
  - Notebooks
  - Common standards (`CUDA`, `OpenCL`)
  - Extremely low prices (1/10 - 1/100)
- Astrophysical applications are well suited
  - Every algo. on an image
  - Ray-tracing

CONs

- Codes have to be ported
  - Data-parallelism
  - Parallel thinking
  - Machine access
- It is still a young field (at least in Astrophysics)
  - Framework not perfect yet (missing libs etc.)
  - Debugging more problematic
  - No real support in the community (, yet)

Conclusions

1. Methods using multiple input constraints allow to constrain the cluster mass profile on all scales.
2. Nonparametric methods can reliably reconstruct this profile without any model assumptions.
   $\Rightarrow$ Good characterisation of a cosmic telescope.
3. GPUs are able to outperform "classical" compute clusters at a fraction of the cost.
4. Astrophysical algorithms appear very well suited for such an implementation.