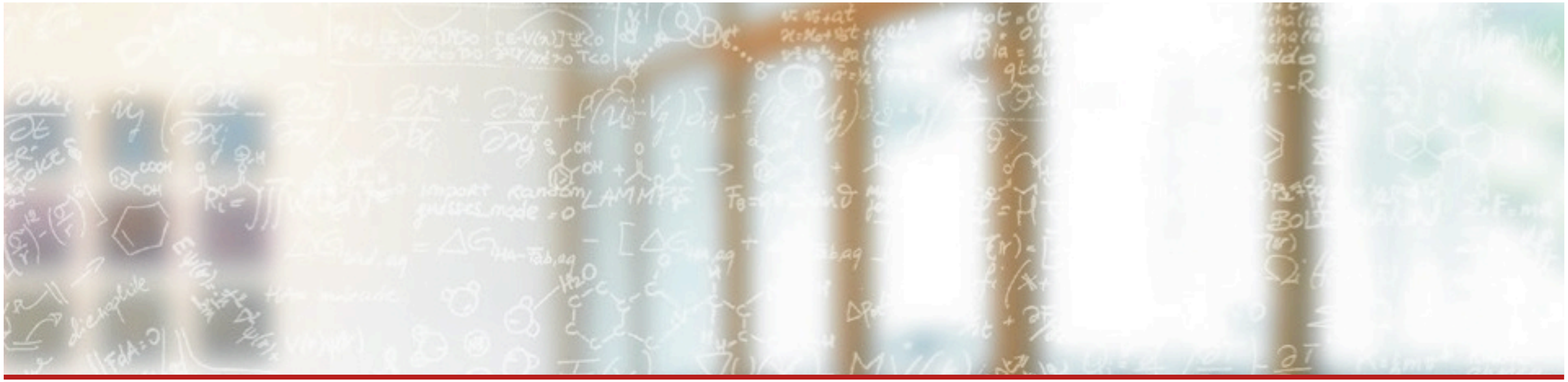
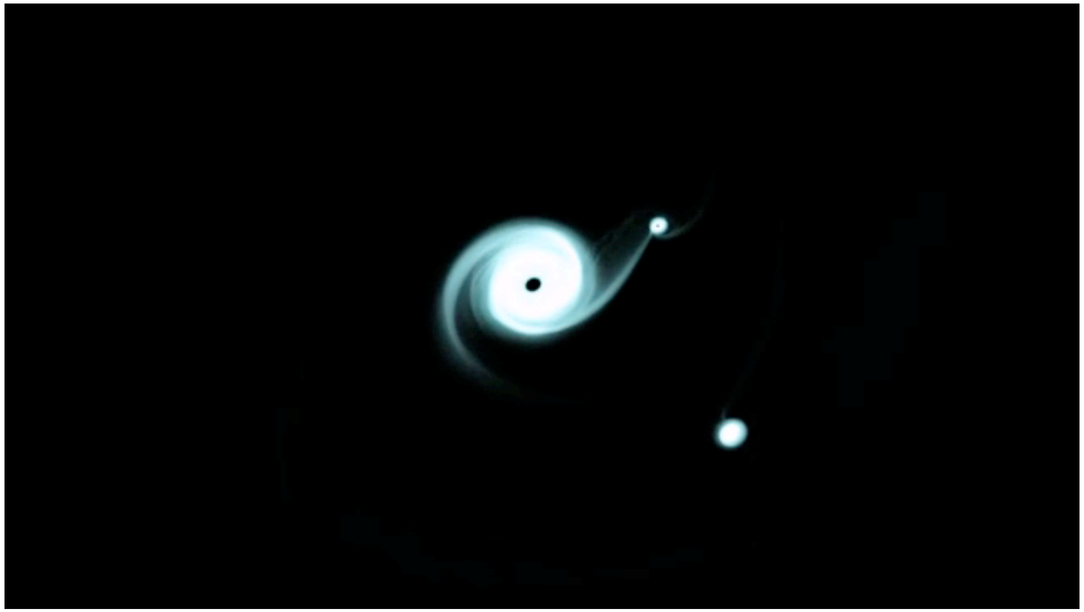
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



HPC visualization of particle like data



CFD & Astrophysics School

Claudio Gheller, CSCS
cgheller@cscs.ch

November 13-15, 2017

How do we squeeze all the information from data?

- **Various tools** offer powerful instruments for automatically analyzing large volumes of data, for classification, association, clustering, etc. (e.g. data mining or statistical tools)
- In general, data analysis is characterized by accurate and sophisticated algorithms that often:
  - scale as N^2 or even N^3 (non-linear behavior)
  - are complex and computationally expensive
  - cannot be optimized/parallelized (not suitable for HPC system)

But an extremely accurate approach is not always necessary…

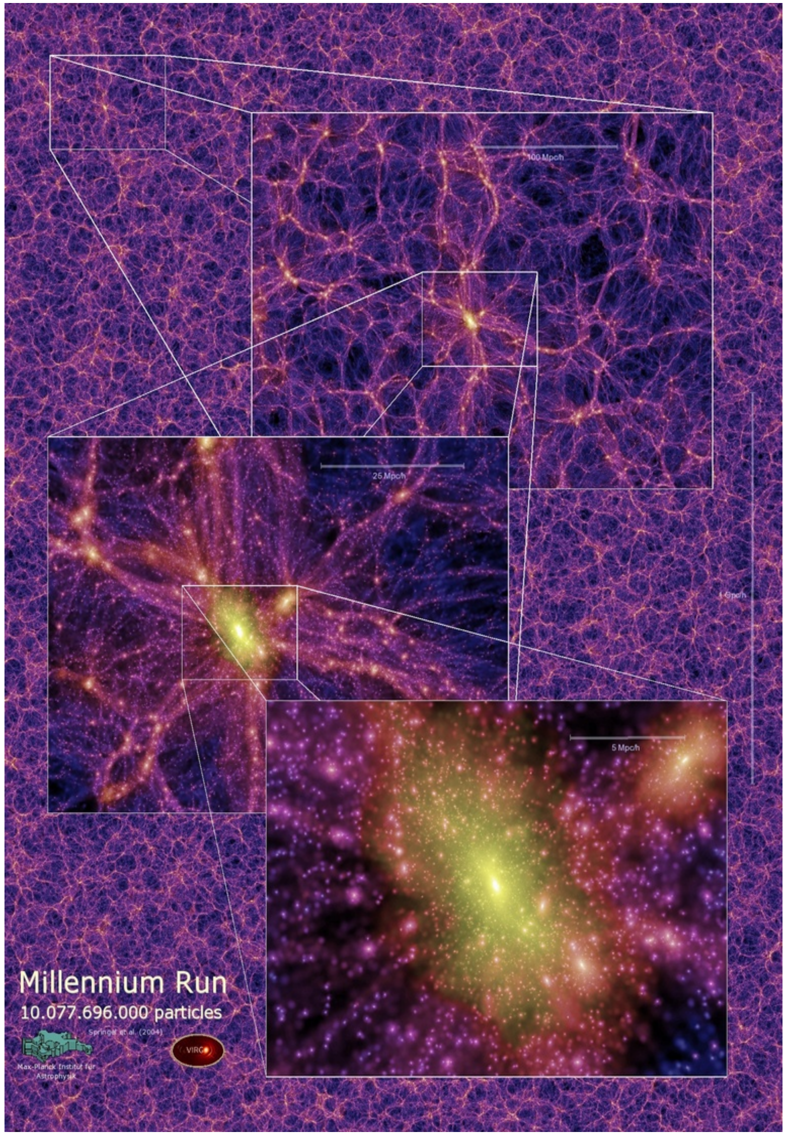
The power of visualization…

some problems require a overall data exploration approach, as that provided by visualization…

Visualization offers an intuitive and immediate insight into data

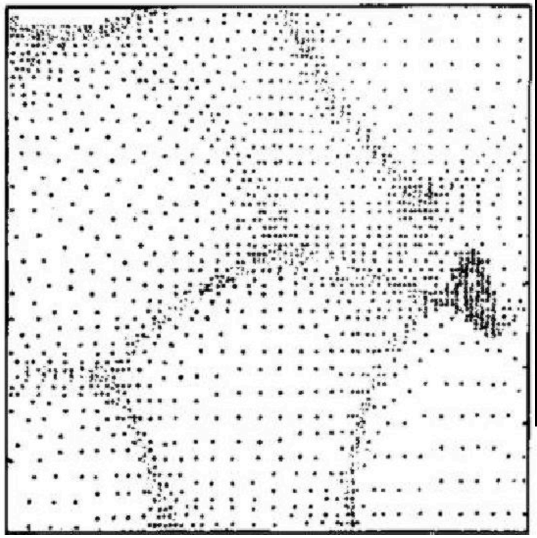
What takes hours for a CPU can take a glance for the human eye!!!

The visualization process plays a fundamental role in understanding data.

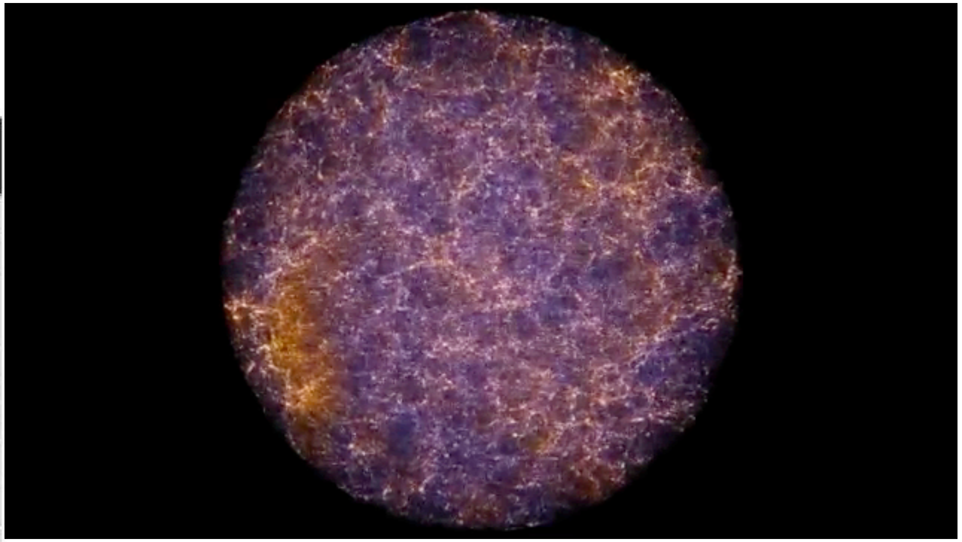


Particle simulations

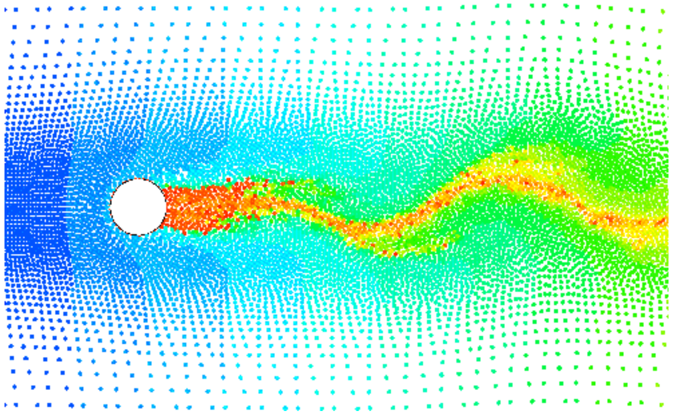
- N-body and SPH method are widely used by the CFD and astrophysics communities
- These methods sample the fluid with points following their dynamics



Distribution of particles in a N-body simulation by the Zeldovich group in 1975: 1550 particles



One of the biggest N-body simulation so far: 2 trillion particles (Stadel et al. 2016)



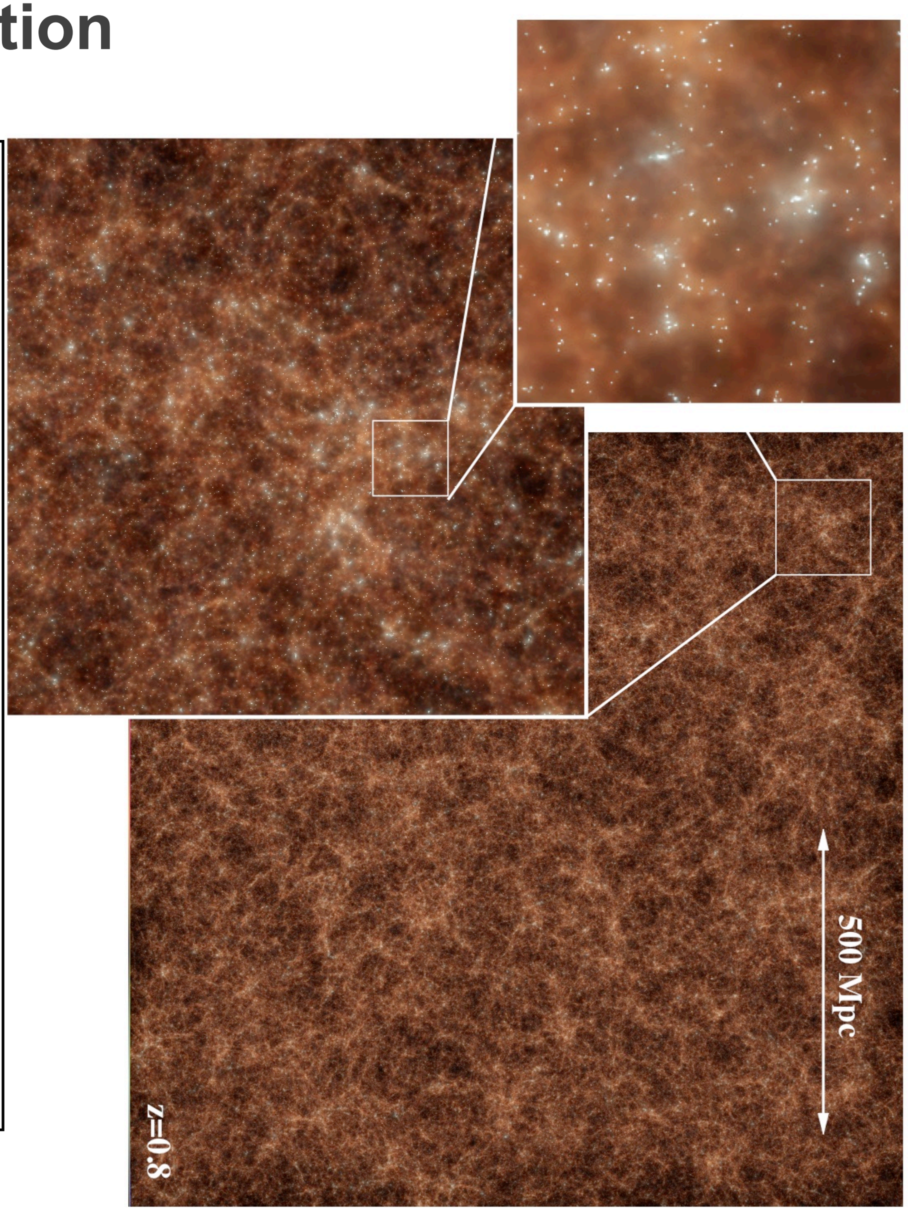
SPH simulation of flow over a cylinder

Splotch for 3D Points Visualization

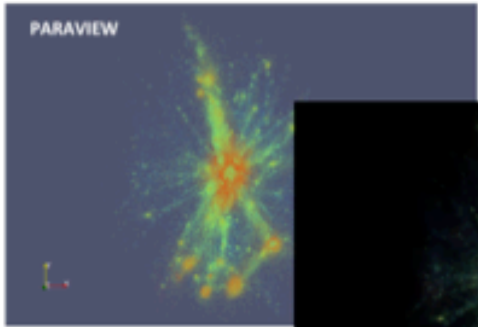
Splotch is a **ray-casting algorithm** for effective visualization of **point-like** datasets, based on the solution of the **radiative transport equation:**

$$\frac{d\mathbf{I}(\mathbf{x})}{dx} = (\mathbf{E_p} - \mathbf{A_p}\mathbf{I}(\mathbf{x}))\rho_{\mathbf{p}}(\mathbf{x})$$

where the density is calculated smoothing the particle quantity on a "proper" neighborhood, by a Gaussian distribution function.



Splotch goals



HIGH QUALITY

3D VISUALIZATION

Visualization offers an intuitive and immediate insight into data

of POINT LIKE SCIENTIFIC DATA

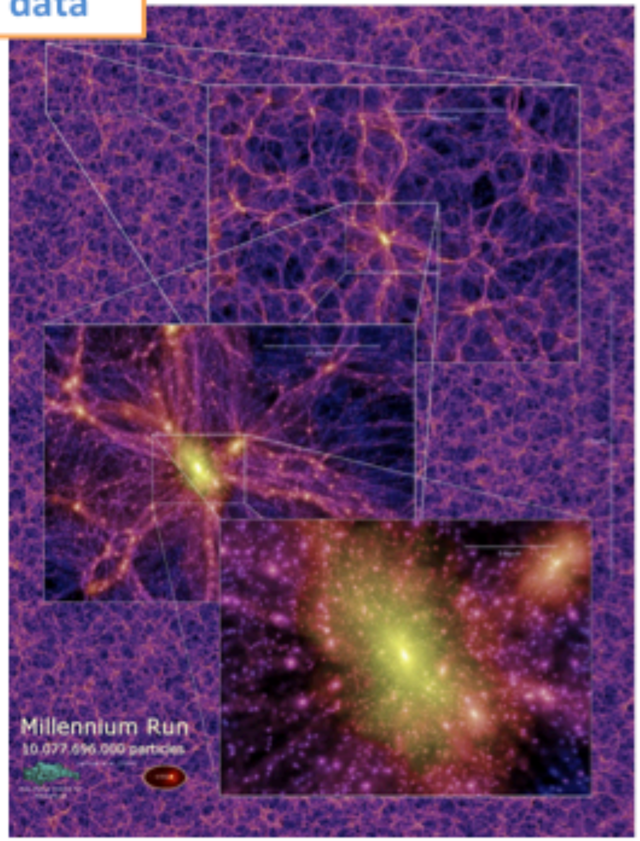
of ANY SIZE

Point data are common in simulations… but not only…

Bigger computers, more and more sophisticated codes, larger instruments…

mean HUGE data…………..

Our approach: develop a specific algorithm for 3D points visualization and rely on supercomputers' brute force…



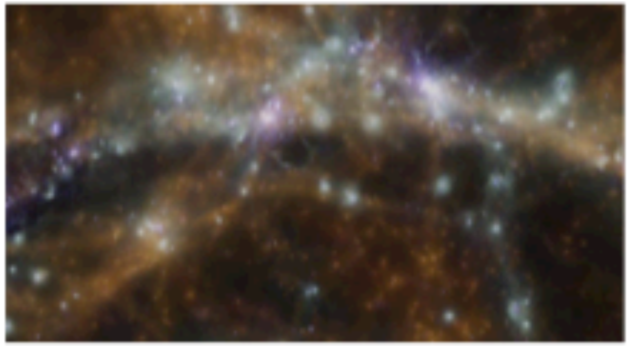
Splotch at a glance

- **Completely open source and self contained**
  - No dependencies
  - Easy to compile
- **Standard C++ based; portable to any platform/ compiler**
  - Easily extensible
  - Usable anywhere
  - Scriptable
  - Support for animations
- **Can exploit (almost) any HPC system**
  - Reduced time to solution
  - Big Data (any size?) can be processed

Download from:
https://github.com/splotchviz/splotch

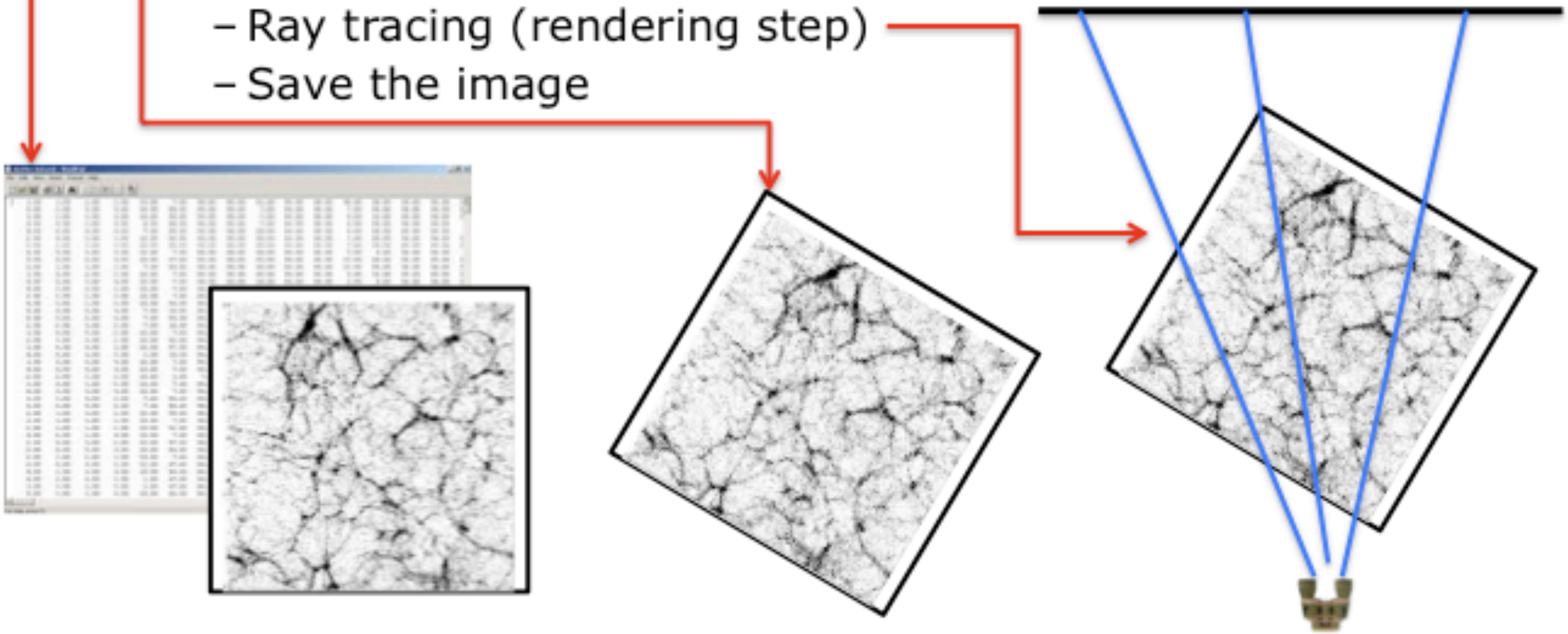
Demo Video (courtesy of Klaus Dolag)

How does Splotch work?



Main steps:

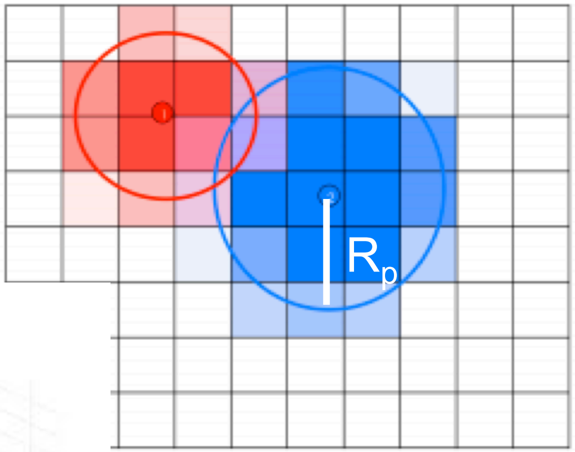
- Read data
- Set the view (rasterization step)
- Ray tracing (rendering step)
- Save the image



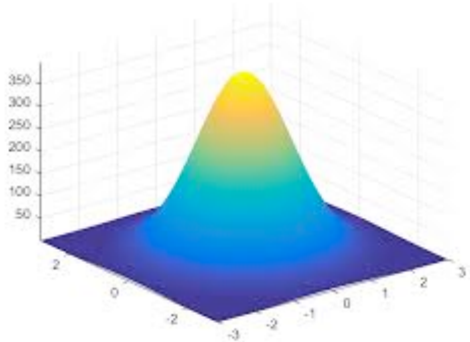
The rendering procedure: weights

$$\frac{d\mathbf{I}(\mathbf{x})}{dx} = (\mathbf{E_p} - \mathbf{A_p}\mathbf{I}(\mathbf{x}))\rho_\mathbf{p}(\mathbf{x})$$

- Each particle influence a given volume, characterized by a smoothing radius R_p (just like for SPH)



- A smoothing kernel is defined to distribute a particle associated quantity on the volume. The kernel is currently a gaussian: $\rho_\mathbf{p}(\mathbf{x})$ is calculated



- The sigma of the gaussian is a fraction of R_p
- The contribution of the particle on a pixel is calculated

The rendering procedure: colors

$$\frac{d\mathbf{I}(\mathbf{x})}{dx} = (\mathbf{E_p} - \mathbf{A_p}\mathbf{I}(\mathbf{x}))\rho_{\mathbf{p}}(\mathbf{x})$$

- E_p and A_p are the emission and absorption coefficients
- They give the color
- In the current implementation $A_p = f(E_p)$
- E_p = available variable (density, temperature)
- Both color tables and RGB are supported (for the latter RT equation is solved for each component and three variables are loaded)
- Support to full physical based rendering almost done

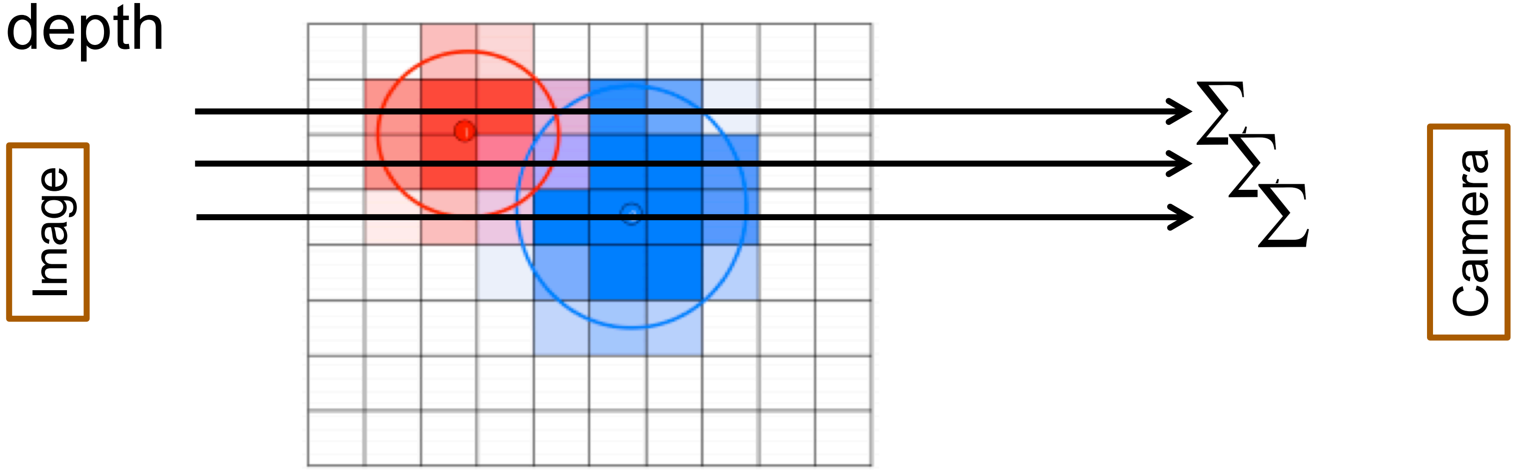
Physical Based Rendering

- The emission E_p and absorption coefficients A_p are calculated according to the physics of the component.
- Examples:
  - X-ray
    - Emission ~ $T^{1/2} \rho^2$
    - Absorption ~ 0
  - Dust (visible light)
    - Emission ~ 0
    - Absorption ~ ρ
- Video: M83 reconstruction

The rendering procedure: integration

$$\frac{d\mathbf{I}(\mathbf{x})}{dx} = (\mathbf{E_p} - \mathbf{A_p}\mathbf{I}(\mathbf{x}))\rho_\mathbf{p}(\mathbf{x})$$

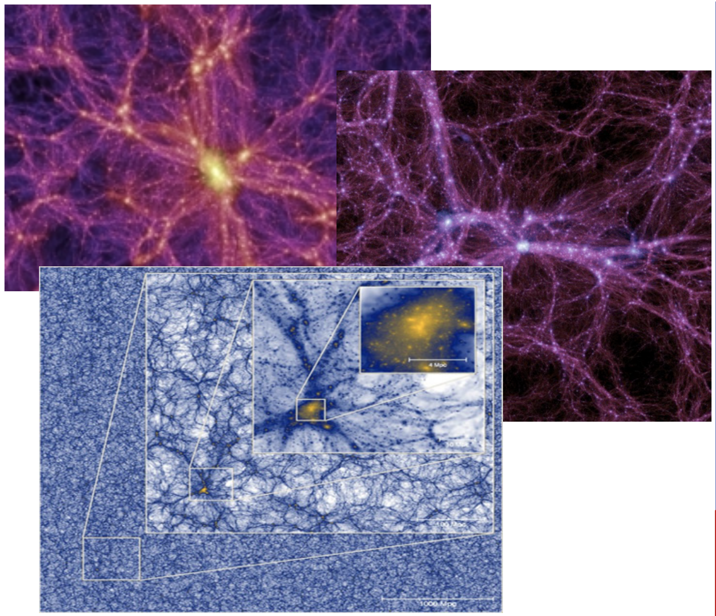
- Particles are sorted with distance from the camera
- Contribution from each particle are summed along the line of sight
- Integration order can be neglected in case of low optical depth



The need for HPC. Use case, N-body simulations

Pure gravity (N-body):	Size:
10^{12} particles	10^{12} elements
Three 3D coordinates + one variable	$4x10^{12}$ elements
Each variable is a float number: 4 bytes	**14.5 TB of MEMORY!**

Not for a laptop!!! We need supercomputer

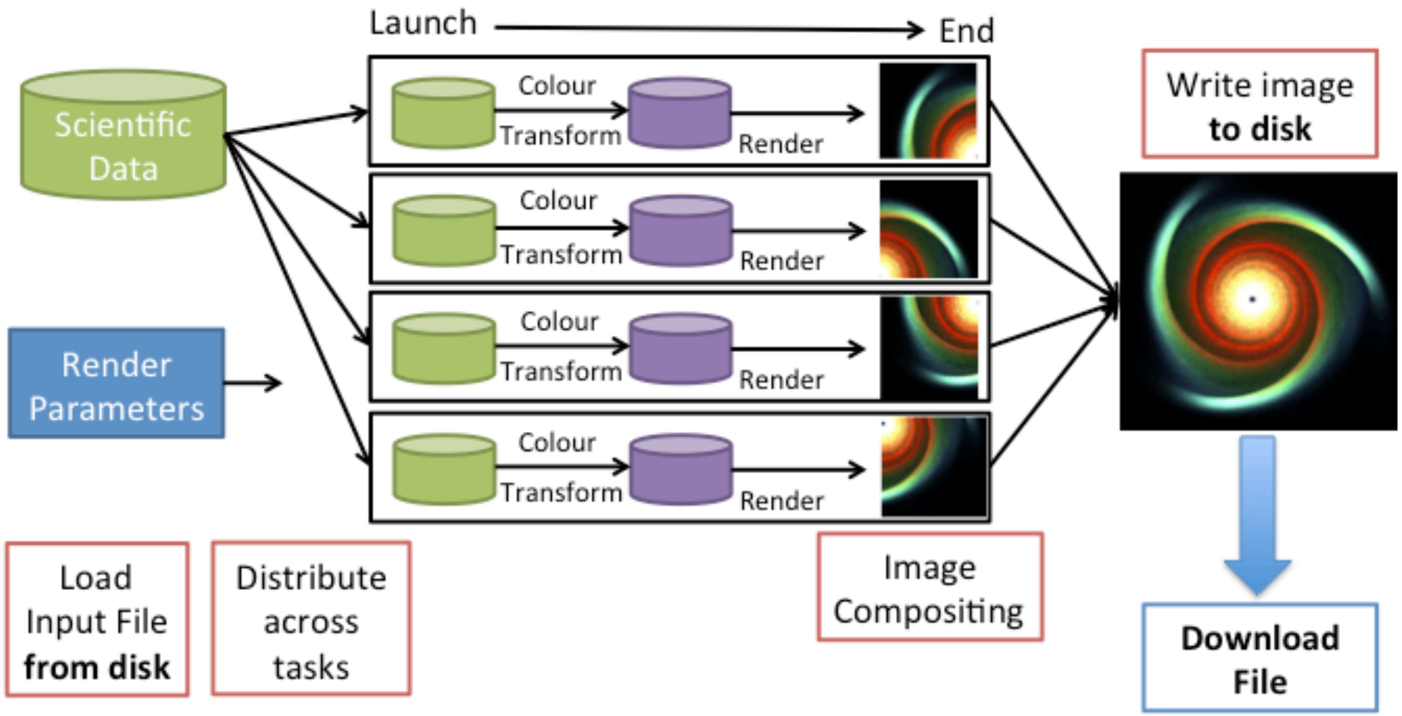


Millennium I (WMAP1-GAD) 500 /h Mpc 10 billion particles
Millennium II (WMAP1 –GAD) 100/h Mpc 10 billion particle
Millenium XXL (WMAP1-GAD) 3 /h Gpc 303 billion particles
Bolshoi (WMAP7-ART) 250/h Mpc 8 billion particles
Multidark (WMAP7-ART) 1 /h Gpc 8 billion particles
BigMD (WMAP7-GAD) 2.5/h Gpc 56.6 billion particles
MICE (WMAP5-GAD) 7 /h Gpc 8 billion particles
Horizon (FR) (WMAP3-RAMSES) 2 /h Gpc 68 billion particles
Horizon (KR) (WMAP5-GOTPM) 10.7 /h Gpc 372 billion.
DEUS (FR) (WMAP7-RAMSES) 21/h Gpc 550 billion particles
JUBILEE (WMAP7-CP3M) 6/h Gpc 216 billion particles

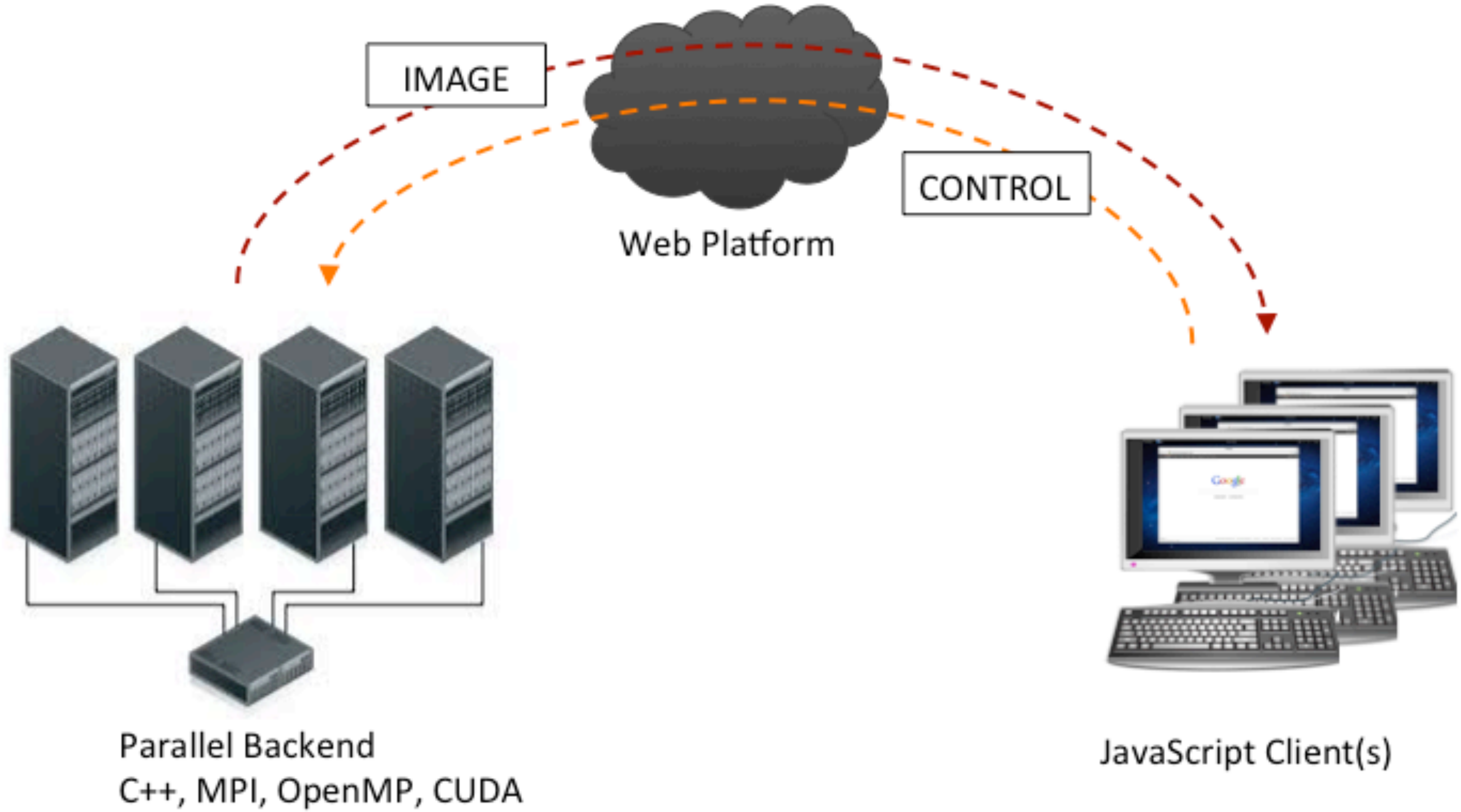
Just run on Piz Daint
EUCLID 2 trillion particles run

Remote data processing

- Data is remote (on the supercomputer disk)
  - Hard to move
  - No local resources to process it
- Computing power is remote (the supercomputer)
- Visualization is on the laptop



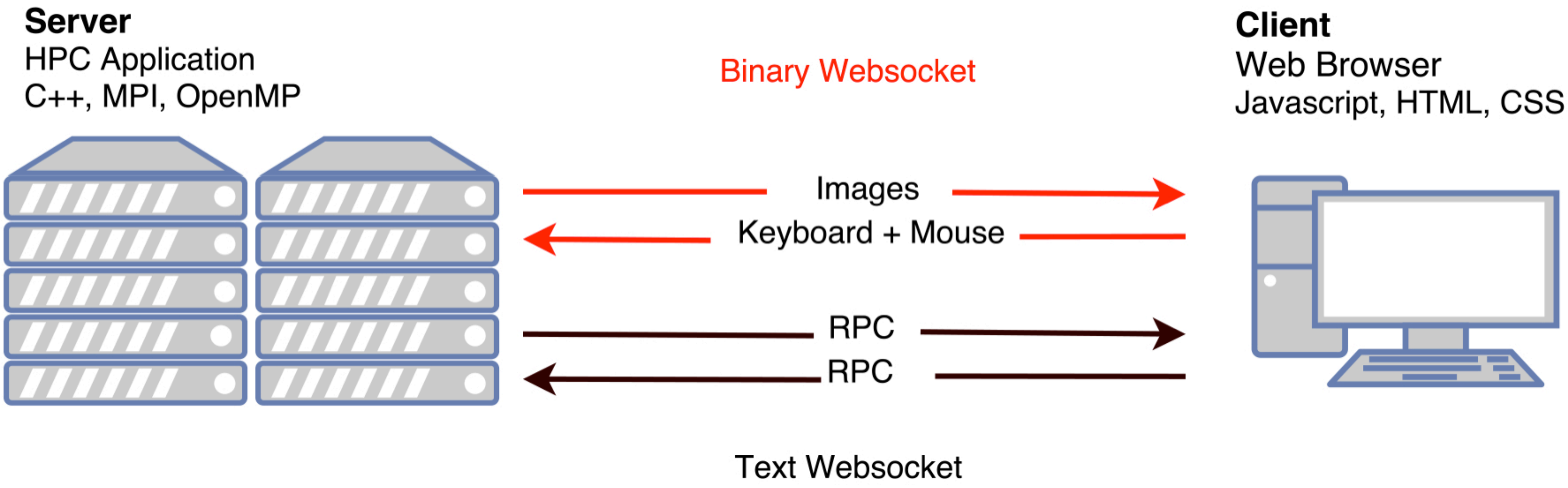
Getting interactive on the web: the simple view



Getting interactive on the web: A little more detail

- Compress image as JPEG
- Serialize to binary
- Send to Client

- Serialize javascript window events to binary (JS Array Buffers)
- Send to Server



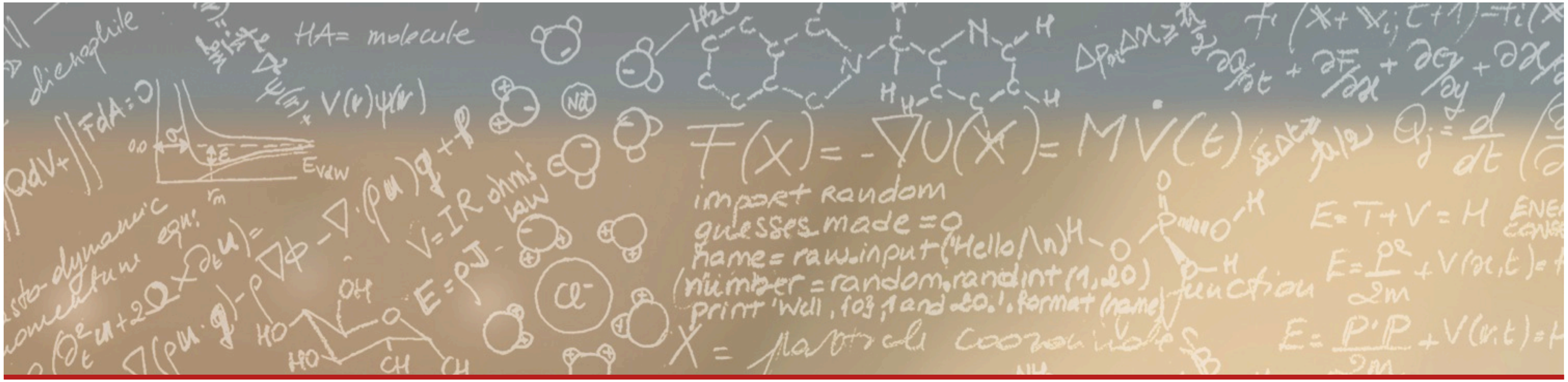
- 2 Way Remote Procedure Call (RPC)
- Generic text based interface:
  - JSON-RPC

- Dynamic interface generation
- C++ Interface descriptor
- JavaScript gat.gui library

LIVE DEMO (hopefully…)

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Thank you for your attention.