CuBA

The Boltzmann Approach for Many Parton Scattering written with CUDA

CFTP, Departamento de Física
Instituto Superior Técnico
Lisboa

Pedro Bicudo
Nuno Cardoso
Janni Bouras
Ulrike Eilhauer

Motivation

- Simulate heavy ion collision, particullary gluon plasma
- Cpu code BAMPS was developed by C. Greiner and Z. Xu from Frankfurt.
- CuBA is physically equivalent to BAMPS, but recoded in CUDA
- The code is benchmarked with the relativistic Riemann Problem
- Comparsion to the original BAMPS

Considerations

Number of particles per cell > 10

Cell size:

$$\lambda = \frac{1}{n\sigma} = \frac{L^3}{N_{num}\sigma}$$

Monte Carlo:

$$P_{22} = v_{rel}\frac{\sigma}{r_{test}}\frac{\Delta t}{L^3} \qquad v_{rel} = \frac{s}{2E_1 E_2}$$

Relativity effects

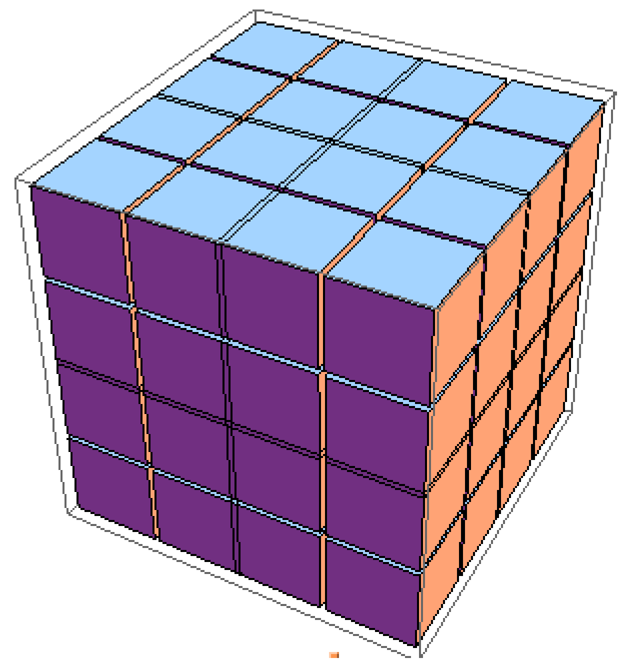


Figure1: Dividing 3D-Space into cells

The code in CUDA

- Advantages of using CUDA:
  - CUDA has a fast shared memory region that can be shared amongst threads.
  - Fast accesses to and from memory of the GPU
  - Calculations and data is GPU
  - Try to minimize tranfers

- Memory accommodates states of all particles corresponding to each particle:
  - Cell number and particle ID
  - 4-vector position
  - 4-vector momentum

The code in CUDA

- CUDA logic:
  - Each block has until 1024 threads and each block has to be lower than 1024x1024x64
  - Each block has 256 threads
  - Each grid has about 65535x65535x65535 blocks for Fermi and 65535x65535x1 for older architectures

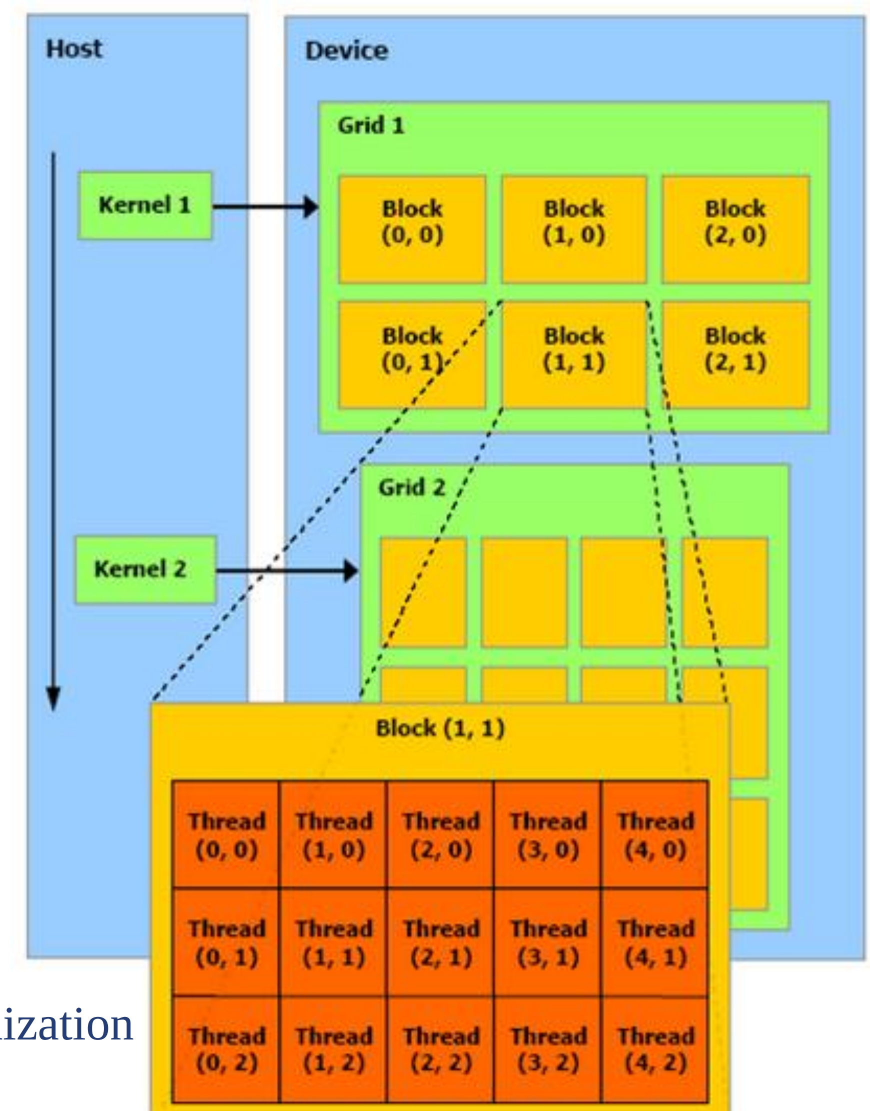
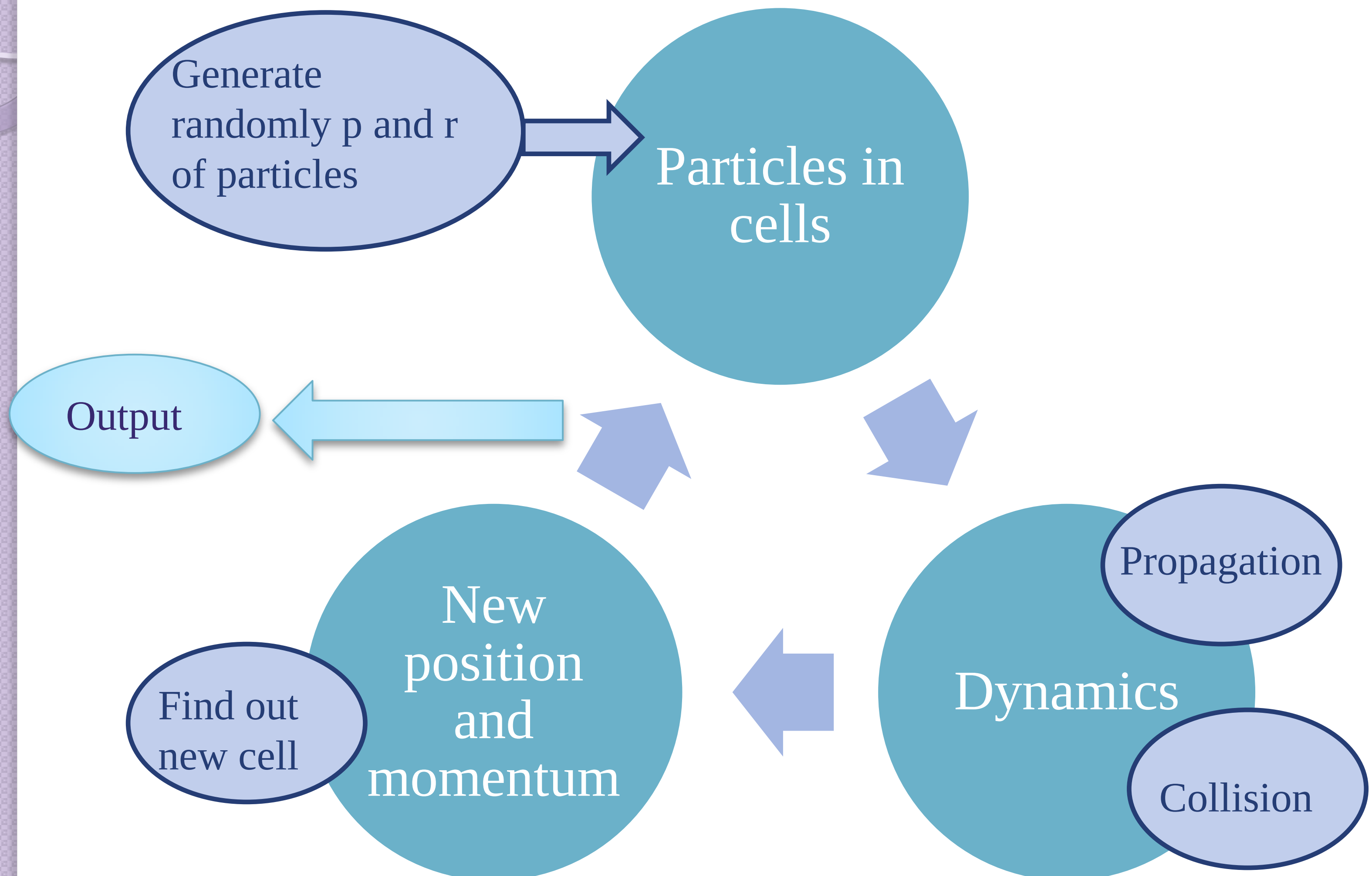


Figure2: CUDA grid organization

How to make the code work



How to make the code work

- Movement of particles:
  - Propagation
  - Collisions
    - Relativistic two body kinematics
    - With the wall

- Find out new cell

Propagation

Propagation is interesting for the particles without collision and after collision occurred.

$$x \to x + v_x \Delta t = x + c^2 \frac{p_x}{E} \Delta t$$

$$y \to y + v_y \Delta t = y + c^2 \frac{p_y}{E} \Delta t$$

$$z \to z + v_z \Delta t = z + c^2 \frac{p_z}{E} \Delta t$$

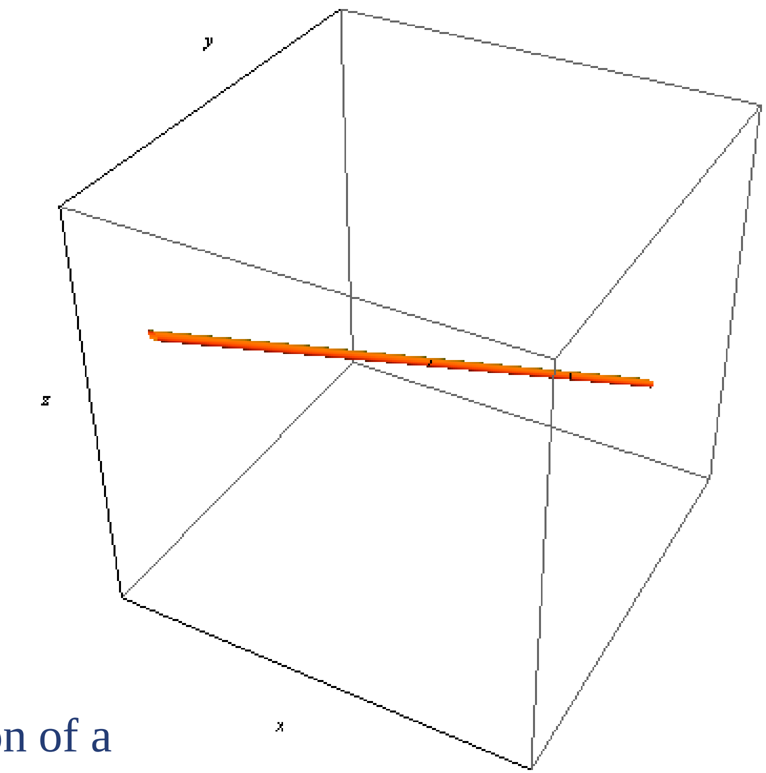
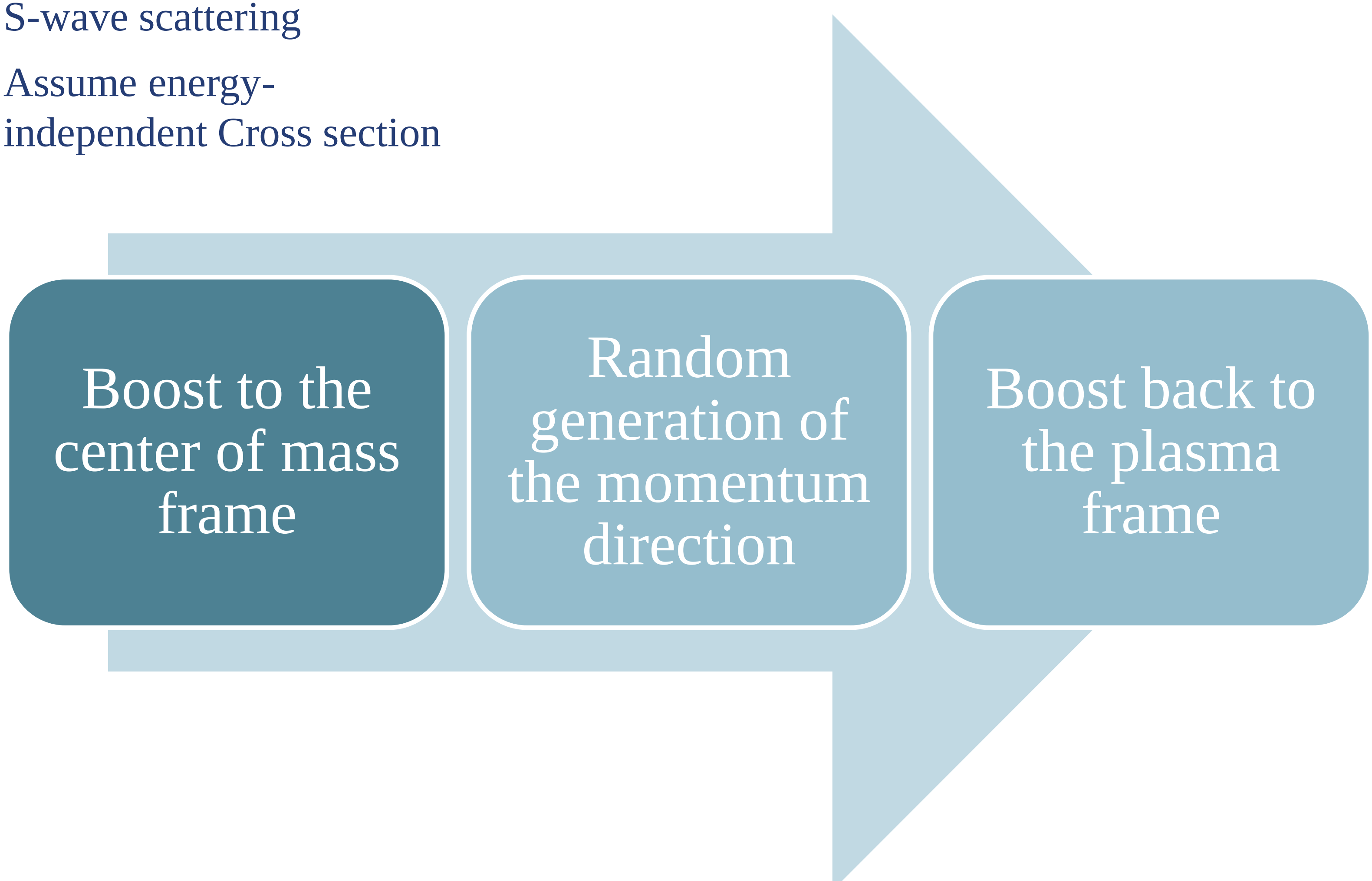


Figure3: Propagation of a particle in space

Relativistic Collision Kernel

S-wave scattering

Assume energy-independent Cross section



Relativistic Collision Kernel

1. Boost to the center of mass

$$\mathbf{p}_1' + \mathbf{p}_2' = 0 \qquad \beta = \frac{V}{c} = c\frac{p_{1\|} + p_{2\|}}{E_1 + E_2} \qquad \gamma = \frac{1}{\sqrt{1-\beta^2}}$$

Getting:

$$\begin{aligned} E_1 \to E_1{}' &= \gamma(E_1 - \beta c p_{1\|}) \\ c p_{1\|} \to c p_{1\|}{}' &= \gamma(-\beta E_1 + c p_{1\|}) \\ c\mathbf{p}_{1\perp} \to c\mathbf{p}_{1\perp}{}' &= c\mathbf{p}_{1\perp} \end{aligned}$$

2. Random generation of the momentum direction

$$p_1{}''_x = \frac{1}{c}\sqrt{s/4 - M^2c^4}\,\cos\phi\,\sqrt{1-\omega^2}$$

$$p_1{}''_y = \frac{1}{c}\sqrt{s/4 - M^2c^4}\,\sin\phi\,\sqrt{1-\omega^2}$$

$$p_1{}''_z = \frac{1}{c}\sqrt{s/4 - M^2c^4}\,\omega$$

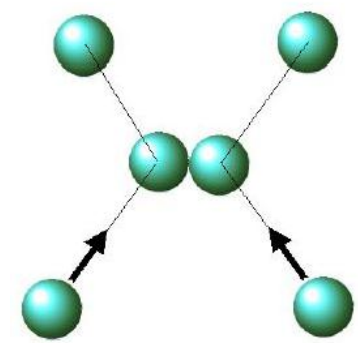


Figure4: Colliding particles

Relativistic Collision Kernel

3. Boost back to plasma frame

Using again the Lorentz transformation

$$E_1'' = E_1' \rightarrow E_1''' \quad = \gamma(E_1' + \beta c p_1''_{\parallel})$$

$$c p_1''_{\parallel} \rightarrow c p_{1\parallel}''' = \gamma(+\beta E_1' + c p_1''_{\parallel})$$

$$c\mathbf{p_1}''_{\perp} \rightarrow c\mathbf{p}_{1\perp}''' = c\mathbf{p_1}''_{\perp}$$

$$\mathbf{p}_1''' = p_{1\parallel}'''\widehat{P} + \mathbf{p}_{1\perp}'''$$

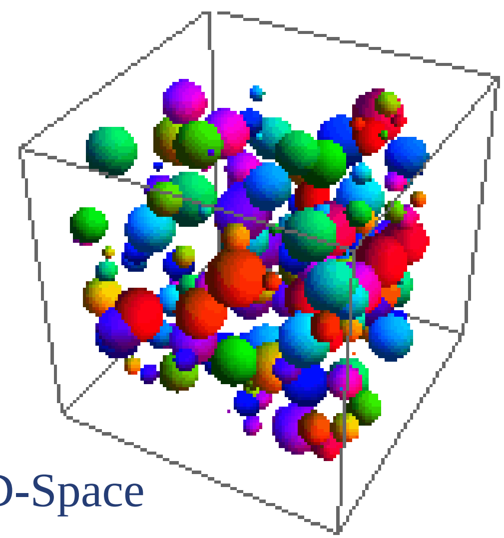


Figure5: Particles in 3D-Space

Collisions with the wall

We have six walls, so we have six 'ifs':

$$\text{if } x > xf$$
$$\text{if } x < xb$$
$$\text{if } y > yr$$
$$\text{if } y < yl$$
$$\text{if } z > zu$$
$$\text{if } z < zd$$

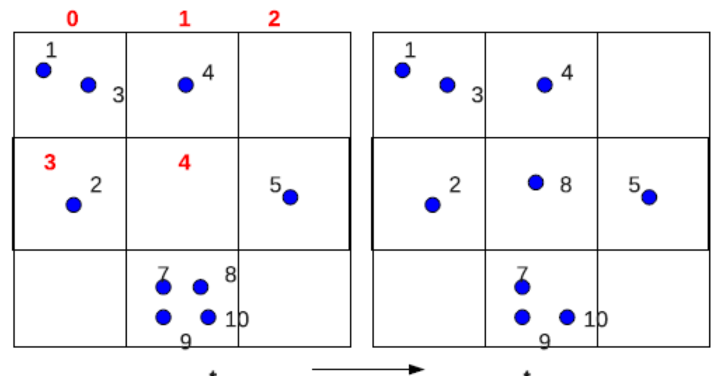
For any of these conditions we reverse the perpendicular momentum:

$$px = -px$$
$$px = -px$$
$$py = -py$$
$$py = -py$$
$$pz = -pz$$
$$pz = -pz$$

And the position:

$$x = -x + 2xf$$
$$x = -x + 2xb$$
$$y = -y + 2yr$$
$$y = -y + 2yl$$
$$x = -z + 2zu$$
$$x = -z + 2zb$$

Find out new cell



t_0

array index:	0	1	2	3	4	5	6	7	8
.y (particle id)	1	3	4	2	5	7	8	9	10
.x (cell id)	0	0	1	3	5	7	7	7	7

Cell id = array index:	0	1	2	3	4	5	6	7	8
cell starts on array 2	0	2	-1	3	-1	4	-1	5	-1

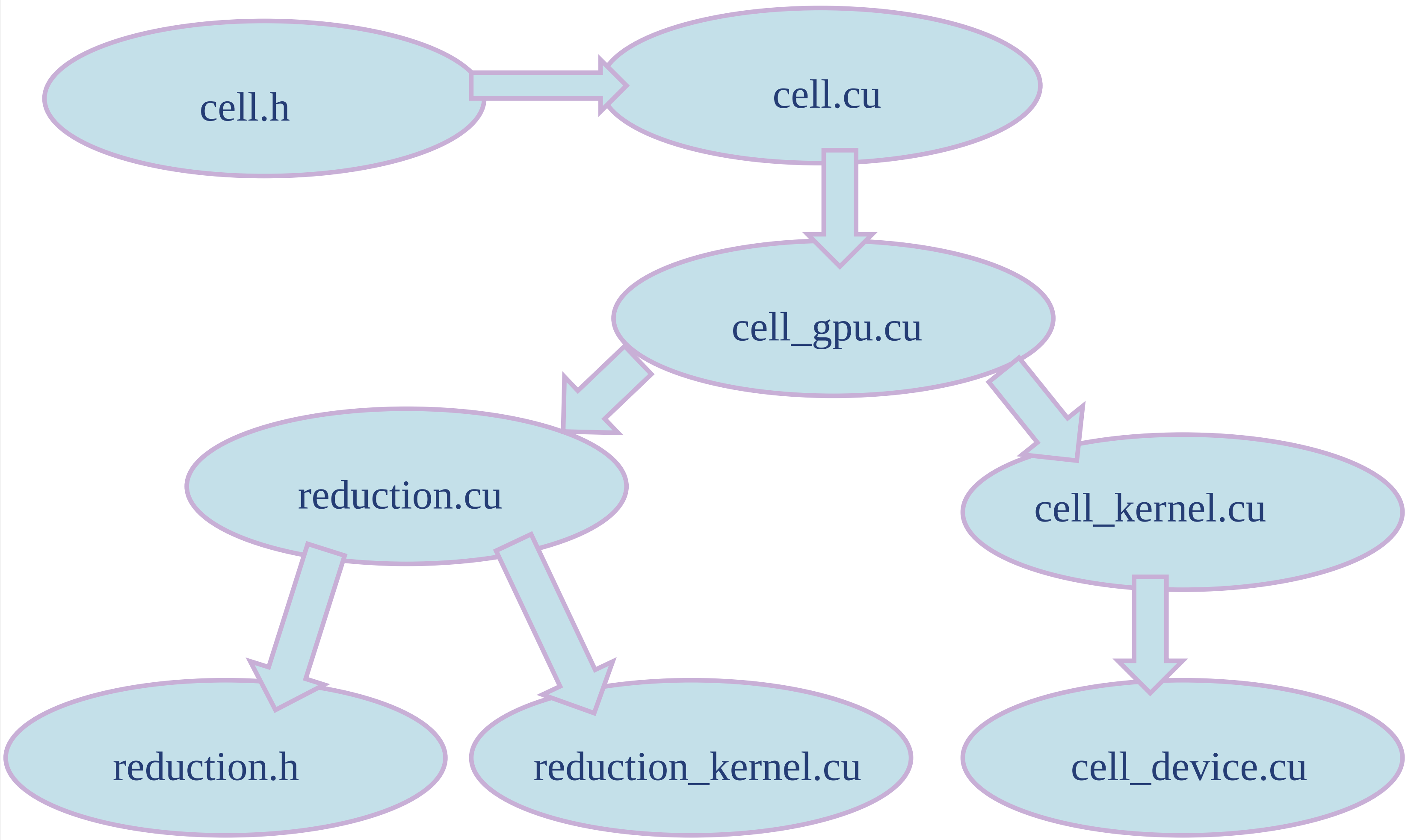
t_1

array index:	0	1	2	3	4	5	6	7	8
.y (particle id)	1	3	4	2	8	5	7	9	10
.x (cell id)	0	0	1	3	4	5	7	7	7

Cell id = array index:	0	1	2	3	4	5	6	7	8
cell starts on array 2	0	2	-1	3	4	5	-1	6	-1

Figure6: Scheme particles in a cell

Our code structure



Results

- Testing the code with the Riemann Problem
- Divide the system with a barrier, assuming special initial conditions
  - Massless Boltzmann gas
  - Tleft= 0.4GeV and Tright=0.2GeV
  - Notice x is lower than Δt

  - Objective: Comparsion BAMPS and CuBA: Compatibility and Time

Results

- Initial conditions:

```
:: cuBA ::

Number of CUDA devices:  2
        0 : GeForce GTX 580
        1 : GeForce GTX 580
Device supporting CUDA compute capability: 2.0

Double precision is supported.
----------------------------------------------------

> Using Double Precision.

------------ Initial Conditions -------------
  Boxlength in X:          32.500000 GeV^-1
  Boxlength in Y:          32.500000 GeV^-1
  Boxlength in Z:          32.500000 GeV^-1
  Volume of Box:           34328.125000 GeV^-3
  CellLength in X:         0.084635 GeV^-1
  CellLength in Y:         32.500000 GeV^-1
  CellLength in Z:         32.500000 GeV^-1
  Volume of cell:          89.396156 GeV^3
  Total cells per side:    384 x 1 x 1
  Total cells:             384
  Initial particles left:  232927
  Initial particles right: 29115
  Total particles:         262042
  TestParticles:           1
  Temperature left:        0.400000
  Temperature right:       0.200000
  Number of iterations:    500
  Time step:               0.010000 GeV^-1/c
  Total time:              5.000000 GeV^-1/c
```

Results

- Observing energy conservation:

```
Writing file....
<-> Iteration: 1
 -> Time: 1.0000e-02 fm/c
  > Collision Kernel
  > Propagate Kernel
  > Sort Cells and Particles per Cell
  > Start/End Cell
  > Calculate Energy and Mean Velocity per Cell
  > Calculate Total Energy
 -> Total Energy: 2.96720997e+05 GeV
```

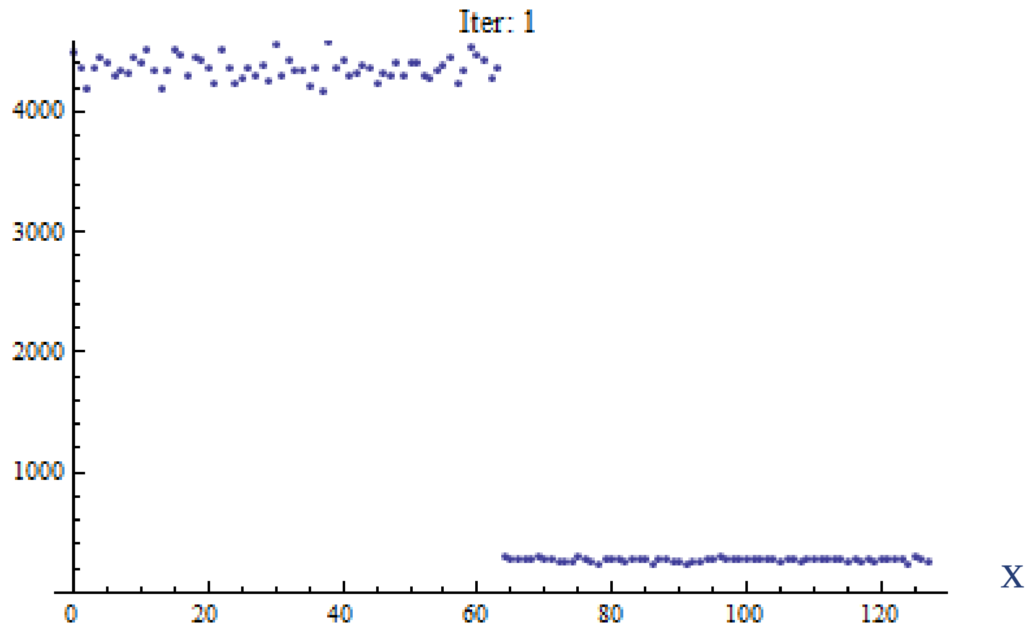
```
Writing file....
<-> Iteration: 500
 -> Time: 5.0000e+00 fm/c
  > Collision Kernel
  > Propagate Kernel
  > Sort Cells and Particles per Cell
  > Start/End Cell
  > Calculate Energy and Mean Velocity per Cell
  > Calculate Total Energy
 -> Total Energy: 2.96720997e+05 GeV
```

- Time:

```
Total time: 33.888449 (s)
Time to initialize: 0.157593 (s)
Time spent in iterate: 33.730855 (s)
```

The Riemann Problem

Energy (GeV)



Video: Energy in function of y

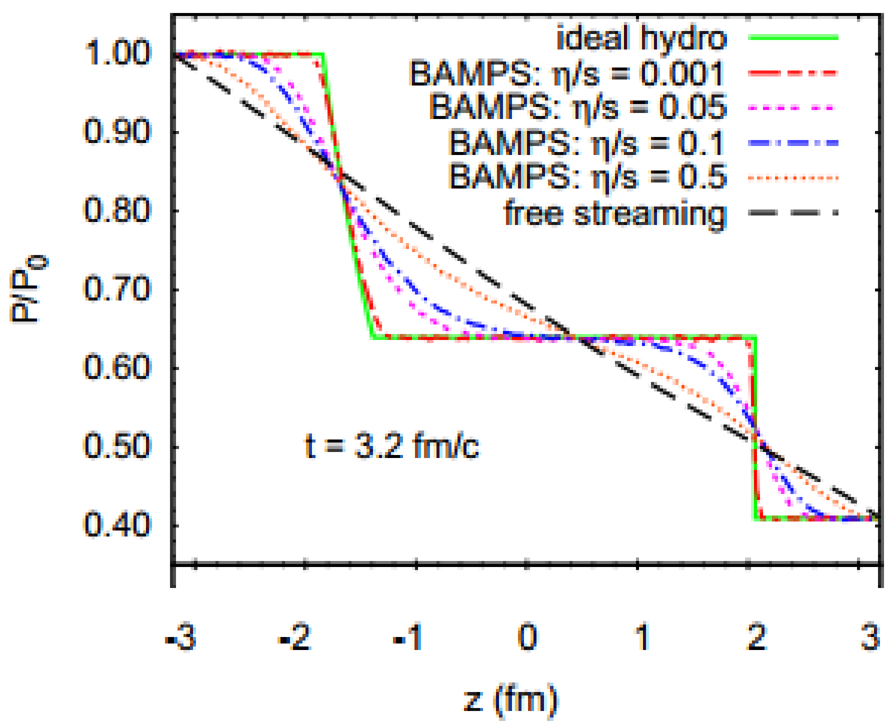
Comparison BAMPS vs. CuBA

- Energy conservation is present
- Time : CuBA is x times faster than BAMPS
- Improving : CuBA is still in evolution

- Next step: More variabels to compare

The Riemann Problem

BAMPS result:



THE END