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, particularly 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:
\[
P_{22} = v_{rel} \frac{\sigma \Delta t}{r_{test} L^3}
\]
\[
v_{rel} = \frac{s}{2E_1 E_2}
\]

Relativity effects

Figure 1: 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 transfers

- 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

Figure 2: CUDA grid organization
How to make the code work

Generate randomly p and r of particles

Particles in cells

Output

New position and momentum

Find out new cell

Dynamics

Propagation

Collision
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.

\[ x \rightarrow x + v_x \Delta t = x + \frac{c^2 p_x}{E} \Delta t \]
\[ y \rightarrow y + v_y \Delta t = y + \frac{c^2 p_y}{E} \Delta t \]
\[ z \rightarrow z + v_z \Delta t = z + \frac{c^2 p_z}{E} \Delta t \]

Figure 3: Propagation of a particle in space
Relativistic Collision Kernel

S-wave scattering
Assume energy-independent Cross section

- Boost to the center of mass frame
- Random generation of the momentum direction
- Boost back to the plasma frame
Relativistic Collision Kernel

1. Boost to the center of mass

\[ p_1' + p_2' = 0 \]
\[ \beta = \frac{V}{c} = c \frac{p_1 \parallel + p_2 \parallel}{E_1 + E_2} \]
\[ \gamma = \frac{1}{\sqrt{1 - \beta^2}} \]

Getting:

\[ E_1 \rightarrow E_1' = \gamma (E_1 - \beta c p_1 \parallel) \]
\[ c p_1 \parallel \rightarrow c p_1 \parallel' = \gamma (-\beta E_1 + c p_1 \parallel) \]
\[ c p_1 \perp \rightarrow c p_1 \perp' = c p_1 \perp \]

2. Random generation of the momentum direction

\[ p_1''_x = \frac{1}{c} \sqrt{s/4 - M^2 c^4} \cos \phi \sqrt{1 - \omega^2} \]
\[ p_1''_y = \frac{1}{c} \sqrt{s/4 - M^2 c^4} \sin \phi \sqrt{1 - \omega^2} \]
\[ p_1''_z = \frac{1}{c} \sqrt{s/4 - M^2 c^4} \omega \]

Figure 4: Colliding particles
Relativistic Collision Kernel

3. Boost back to plasma frame

Using again the Lorentz transformation

\[ E_{1'''} = E_{1'} \rightarrow E_{1'''} = \gamma(E_{1'} + \beta c p_{1'''}_\parallel) \]
\[ c p_{1'''}_\parallel \rightarrow c p_{1'''}_\parallel = \gamma(+\beta E_{1'} + c p_{1'''}_\parallel) \]
\[ c p_{1'''}_\perp \rightarrow c p_{1'''}_\perp = c p_{1'''}_\perp \]

\[ p_{1'''} = p_{1'''}_\parallel \hat{P} + p_{1'''}_\perp \]

Figure 5: Particles in 3D-Space
Collisions with the wall

We have six walls, so we have six ‘ifs’:

- if $x > x_f$
- if $x < x_b$
- if $y > y_r$
- if $y < y_l$
- if $z > z_u$
- if $z < z_d$

For any of these conditions we reverse the perpendicular momentum:

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

And the position:

$$x = -x + 2x_f$$
$$x = -x + 2xb$$
$$y = -y + 2yr$$
$$y = -y + 2yl$$
$$x = -z + 2zu$$
$$x = -z + 2zb$$
Find out new cell

Figure 6: Scheme particles in a cell

5/11/2012 CuBA
Our code structure

- cell.h
- cell.cu
- cell_gpu.cu
- reduction.cu
- reduction.h
- reduction_kernel.cu
- cell_kernel.cu
- cell_device.cu
Results

- Testing the code with the Riemann Problem
- Divide the system with a barrier, assuming special initial conditions
  - Massless Boltzmann gas
  - $T_{\text{left}} = 0.4\text{GeV}$ and $T_{\text{right}} = 0.2\text{GeV}$
  - Notice $x$ is lower than $\Delta t$

- Objective: Comparsion BAMPS and CuBA: Compatibility and Time
Results

- Initial conditions:

```plaintext
:: 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 variables to compare
The Riemann Problem

BAMPS result:
THE END