(3+1)D Viscous Hydrodynamics On GPU for relativistic heavy ion collisions

LongGang Pang
UC Berkeley & LBNL

2017.05.23 in Seattle, UW
Hydrodynamics is the bottleneck in JETSCAPE

Fast (< seconds on cpu)

Jet shower

Initial condition

Hydrodynamic evolution

Hadron cascade

Slow (several hours on cpu)

- Hydrodynamic evolution is much slower than jet shower propagation which hinders concurrent running.
Big-data in heavy ion collisions (Bayesian method)

- Bayesian method

\[ P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)} \]

X: model —— Y: data

\[ 10^4 \sim 10^7 \text{ events} \]

PRC 94.024907, J.E.Bernhard. et.el.
PRL. 114, 202301, S. Pratt, et.el
Big-data in heavy ion collisions (Deep Convolution Neural Network)

An EoS-meter of QCD transition from deep learning

- $O(10^4)$ events from CLVisc and iEBE-VISHNU (by C.Shen, Z.Qiu, H.C.Song, J.Bernhard, S.Bass, U.Heinz) are used, more are needed in the future for other studies.

- Huge amount of labeled events are required to get the most relevant feature in supervised learning (no matter what kind of initial state fluctuations or irrelevant parameters are employed in the model).
• GPUs have more processing elements (PE) than CPUs. 4992 PE/Cuda cores (GPU Tesla K80) vs 8-18 cores (Intel Xeon E5 server CPU)

• Peak performance: 5.6 Tflops (Tesla K80) vs ~700 Gflops (Intel Xeon E5 server CPUs)
Global Memory

- Global memory: GPU side, 1 – 12 GB, speed 100 – 300 GB/s, latency 400 clock cycles.
- 400 clock cycles \( \equiv (400 +) \) or \((100 \ast)\) or (20-40 square root).
- Use more workitems per workgroup to hide latency (warp switching).
- Do extra calculation other than Global memory access.
- Slowest
Local Memory

- Local memory: on CU, \(16 - 64KB\), speed \(600 - 800\) GB/s, latency \(1 - 40\) clock cycles
- Used when multi workitems in the same workgroup share data
- No data sharing, do not use local memory (slower than private memory).
- Faster
Private memory

- Private memory: on PE, 16-64K per CU.
- Used if global/local/constant memory is accessed by one workitem multiple times.
First application of GPU parallelization

Reduction for spectra and max energy density

- Parallel reduction to get maximum, minimum, summation for a big array.

- Stop hydro evolution when maximum temperature of QGP smaller than freeze out temperature.

- Calc. spectra by summation over all the freeze out hyper-surface elements.
Spectra calculation on GPU

Perfect job for GPU

\[
\frac{dN}{dY \, p_T \, dp_T \, d\phi} = \frac{g_s}{(2\pi)^3} \int_{\Sigma} p^\mu \, d\Sigma_\mu \, \frac{1}{\exp\left( (p \cdot u - \mu) / T_{FO} \right) \pm 1}
\] (2)

- Up to 200,000 small pieces of \(d\Sigma_\mu\).
- Usually need 41 rapidity(\(Y\)) bins, 15 transverse momentum(\(p_T\)) bins, 48 azimuthal angle(\(\phi\)) bins.
- More than 300 resonance particles.
- For each event, needs to calc. \(\exp\) function \(200,000 \times 41 \times 15 \times 48 \times 300\) times.

Pb+Pb 2.76TeV/n, 20-25%

<table>
<thead>
<tr>
<th>Smooth spec. for (\pi^+)</th>
<th>CPU (i5-430M)</th>
<th>GPU (GT-240M)</th>
<th>GPU (K20)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 minutes</td>
<td>30 seconds</td>
<td>0.5 seconds</td>
<td></td>
</tr>
</tbody>
</table>

Table: GPU(48 cuda cores) in my laptop is 10-30 times faster than CPU. NVIDIA K20 GPU has 2496 cuda cores.
OpenCL is a framework for writing programs that execute across heterogeneous platforms consisting of central processing units (CPUs), graphics processing units (GPUs), digital signal processors (DSPs), field-programmable gate arrays (FPGAs) and other processors.

- Open Standard maintained by Khronos org.
- Host language: C/C++/Python/Julia/Java.
- Device language: C99 (subset)

CUDA:
Specific for Nvidia GPUs, C/C++/Python/
Modern deep learning libraries all uses CUDA

Manuscript Title: Massively parallel simulations of relativistic fluid dynamics on graphics processing units with CUDA
Authors: Dennis Bazow, Ulrich Heinz, Michael Strickland
Program Title: GPU-VH
CLVisc: a (3+1)D viscous hydrodynamics parallelized on GPU using OpenCL

\[ \nabla_\mu T^{\mu \nu} = 0 \]  

(1)

\[ \Delta^{\mu \nu \alpha \beta} u^\lambda \nabla_\lambda \pi_{\alpha \beta} = - \frac{\pi^{\mu \nu} - \pi^{\mu \nu}_{\text{NS}}}{\tau_\pi} - \frac{4}{3} \pi^{\mu \nu} \nabla_\lambda u^\lambda \]  

(2)

where

\[ T^{\mu \nu} = (\varepsilon + P)u^\mu u^\nu - Pg^{\mu \nu} + \pi^{\mu \nu} \]  

(3)

\[ \Delta^{\mu \nu \alpha \beta} = \frac{1}{2} (\Delta^{\mu \alpha} \Delta^{\nu \beta} + \Delta^{\nu \alpha} \Delta^{\mu \beta}) - \frac{1}{3} \Delta^{\mu \nu} \Delta^{\alpha \beta} \]  

(4)

\[ \Delta^{\mu \nu} = g^{\mu \nu} - u^\mu u^\nu, \quad g^{\mu \nu} = \text{diag}(1, -1, -1, -\tau^{-2}) \]  

(5)

\( \varepsilon \) and \( P \) are the energy density and pressure, \( u^\mu \) is the fluid velocity vector. \( \nabla_\mu \) is the covariant derivative.

- Constraints: \( P = P(\varepsilon) \), \( u_\mu u^\mu = 1 \), \( u_\mu \pi^{\mu \nu} = 0 \), \( \pi^\mu_\mu = 0 \).

CLVisc, L.G. Pang, B.W. Xiao, Y. Hatta, X.N.Wang, PRD 2015
In 3d, each cell shares data with 12 neighbors, better to use local memory.

Performance of KT evolution for most central Pb+Pb collisions

<table>
<thead>
<tr>
<th></th>
<th>CPU (E5-2650)</th>
<th>K20 $(5 \times 5 \times 5$ block)</th>
<th>K20 $(7 \times 7 \times 7$ block)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 hour</td>
<td>3 minutes</td>
<td>50 seconds</td>
</tr>
</tbody>
</table>

- $5 \times 5 \times 5$ block with $T^{\mu \tau}$, $\varepsilon$, $P$, $U^\mu$, $cs^2$ in local memory
- $7 \times 7 \times 7$ block with $T^{\mu \tau}$, $\varepsilon$, $P$, $v^i$ in local memory

Too many halo cells, use too much local memory, difficult to implement in viscous hydro.
KT algorithm for PDE (new implementation in CLVisc)

\[
\frac{d\tilde{Q}}{d\tau} = -\frac{H_{i+1/2,j,k}^x - H_{i-1/2,j,k}^x}{dx} \quad \frac{d\tilde{Q}}{d\tau} = -\frac{H_{i,j+1/2,k}^y - H_{i,j-1/2,k}^y}{dy} \quad \frac{d\tilde{Q}}{d\tau} = -\frac{H_{i,j,k+1/2}^\eta - H_{i,j,k-1/2}^\eta}{\tau d\eta}
\]

- Using dimension splitting, put each strip of data to local memory. Only 4 hallo cells in each local memory.
- Easier to implement, no performance loss.
Since the energy density distribution is not uniform in this coordinate system, one simply gets different transport coefficients.

In principle, this solution is inspired to choose a negative extrapolation, which preserve the trace distribution of particle number.

After the Weyl rescaling, we can get another interesting property of this 2nd order viscous hydrodynamics. In principle, the hydrodynamic equations recover the ideal fluid solution. It is roughly two times accurate energy density as compared to the Gubser solution.

Another interesting property of this 2nd order viscous hydrodynamics is that for ideal hydrodynamics, since it is fixed by conformal transformation.

We use Cooper-Frye formula to calculate the mean transverse momentum distribution of particle from CLVisc numerical results and Gubser analytical solution for 2nd order viscous hydrodynamics.
Particle $dN/d\text{Eta}$ and Spectra

- With Trento (Duke Group) initial condition
- Centrality ranges are the same as used in JETSCAPE.
Longitudinal de-correlation of anisotropic flows

- With string length fluctuations, CLVisc+AMPT initial condition describes rapidity de-correlation of anisotropic flows.

Fig. 2. (Color online) The longitudinal fluctuations for (left) Pb+Pb 2.76 TeV collisions and (right) Au+Au 200 GeV collisions for three typical events at centrality classes 0–1%, 20–30% and 40–50%.
Complex vortical fluid in heavy ion collisions

- Vortex pairs in transverse plane to conserve angular momentum.
- Signal can be found using spin-correlation.

Vortex pair in 2D
Vortex ring in 3D = Toroidal (smoke ring) vortical fluid
Azimuthal angle dependence.
Rapidity dependence

LG.Pang, H.Petersen, Q.Wang & XNW PRL 117 (2016) no.19, 192301

by Lucas V. Barbosa from Wiki Pedia
Profiling of CLVisc on GPU and CPU

<table>
<thead>
<tr>
<th>block size</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ideal(s)-GPU</td>
<td>0.37</td>
<td>0.218</td>
<td>0.178</td>
<td>0.155</td>
<td>0.157</td>
</tr>
<tr>
<td>Visc(s)-GPU</td>
<td>3.12</td>
<td>1.65</td>
<td>1.17</td>
<td>1.01</td>
<td>1.17</td>
</tr>
<tr>
<td>Visc(s)-CPU</td>
<td>6.64</td>
<td>6.45</td>
<td>6.63</td>
<td>7.0</td>
<td>7.58</td>
</tr>
</tbody>
</table>

- 385 * 385 * 115 grids, for Pb+Pb 2.76 TeV
- block size=64 is best for GPU (AMD firepro s9150)
- block size=16 is best for 10 core CPU (Intel Xeon E5-2650)
- CLVisc on GPU is about 6.4 times faster than the same code on a 10 core server CPU. (both are parallelized and use SIMD) for each time step.

block size: how many working elements are assigned to the same working group sharing the same local memory.

- AMD Firepro s9150 used in GSI GreenCube
- 2,816 stream processors (44 compute units)
- 5.07 TFLOPS SP
- 16GB memory
- 320GB/s memory bandwidth
- 235W maximum power consumption
The CUDA implementation GPU-VH by Ohio group

Table 4: Same as Table (3+1)D Viscous Hydrodynamics on GPU

<table>
<thead>
<tr>
<th>Number of grid points</th>
<th>C/CPU (ms/step)</th>
<th>CUDA/GPU (ms/step)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>128 × 128 × 32</td>
<td>7690.069</td>
<td>96.923</td>
<td>79.342</td>
</tr>
<tr>
<td>128 × 128 × 64</td>
<td>16315.976</td>
<td>192.751</td>
<td>84.648</td>
</tr>
<tr>
<td>128 × 128 × 128</td>
<td>38428.056</td>
<td>384.255</td>
<td>100.007</td>
</tr>
<tr>
<td>256 × 256 × 32</td>
<td>30401.898</td>
<td>378.178</td>
<td>80.390</td>
</tr>
<tr>
<td>256 × 256 × 64</td>
<td>72240.973</td>
<td>744.168</td>
<td>97.076</td>
</tr>
<tr>
<td>256 × 256 × 128</td>
<td>144744.290</td>
<td>1485.703</td>
<td>97.423</td>
</tr>
<tr>
<td>256 × 256 × 256</td>
<td>322536.875</td>
<td>2970.727</td>
<td>108.572</td>
</tr>
</tbody>
</table>

Tesla K20M vs 1.8GHz Intel Xeon CPU E5-2630L v3.
Summary

• Big data analysis (Bayesian statistics and deep learning) for heavy ion collisions require fast (3+1)D viscous hydrodynamics

• Concurrently running jet shower propagation and hydrodynamic evolution needs fast hydro

• GPU is good at data parallelization

• We have OpenCL and CUDA backends for the final JETSCAPE hydrodynamic module.

• GPU parallelization brings 100 times performance boost (vs single core CPU).