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

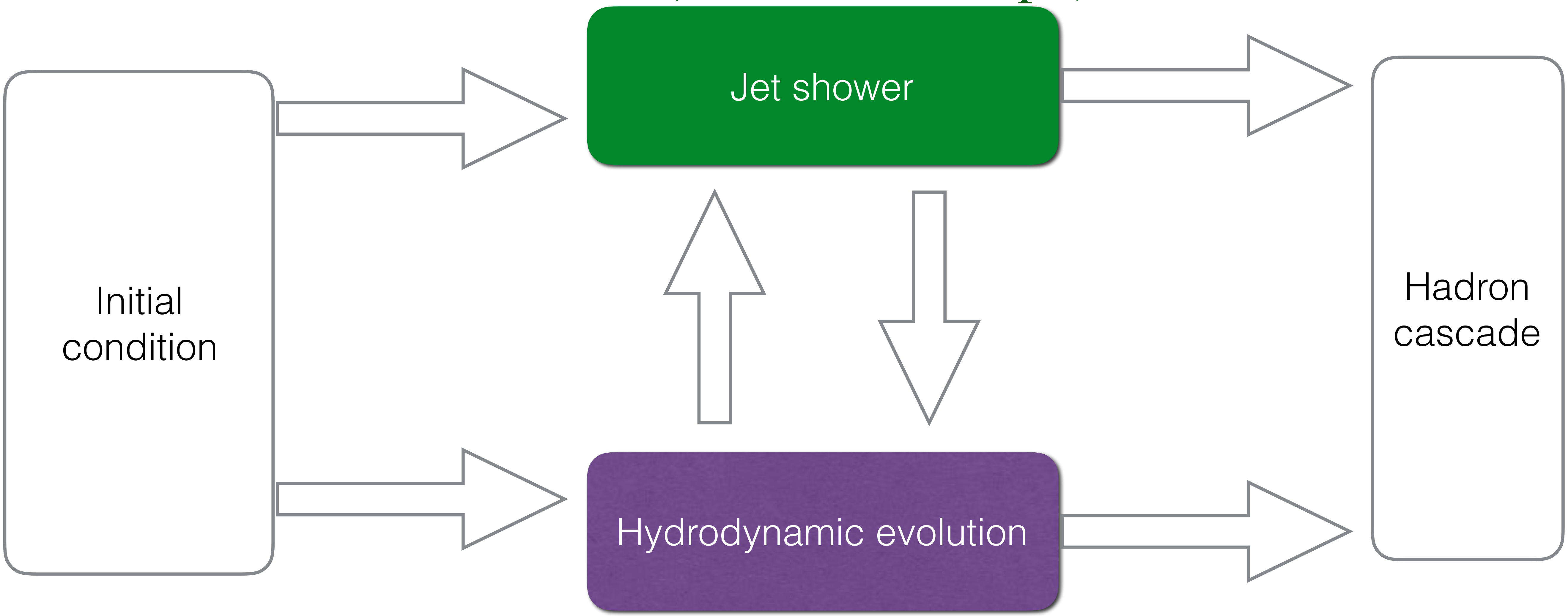
LongGang Pang
UC Berkeley & LBNL

2017.05.23 in Seattle, UW

JETSCAPE

Hydrodynamics is the bottleneck in JETSCAPE

Fast (< seconds on cpu)



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)

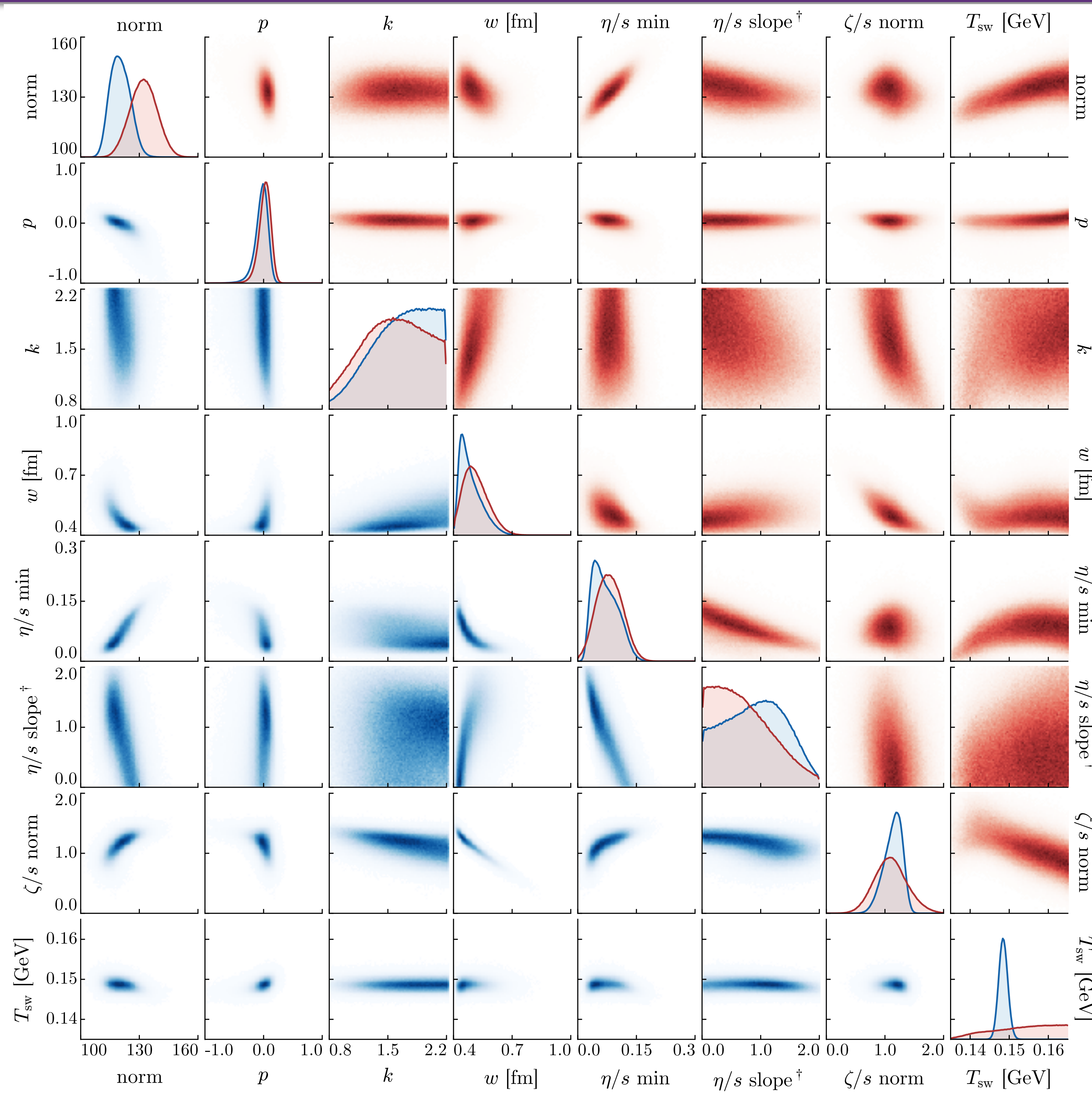


FIG. 7. Posterior distributions for the model parameters from calibrating to identified particles yields (blue, lower triangle) and charged particles yields (red, upper triangle). The diagonal has marginal distributions for each parameter, while the off-diagonal contains joint distributions showing correlations among pairs of parameters. †The units for η/s slope are [GeV^{-1}].

TABLE I. Input parameter ranges for the initial condition and hydrodynamic models.

Parameter	Description	Range
Norm	Overall normalization	100–250
p	Entropy deposition parameter	-1 to $+1$
k	Multiplicity fluct. shape	0.8–2.2
w	Gaussian nucleon width	0.4–1.0 fm
η/s hrg	Const. shear viscosity, $T < T_c$	0.3–1.0
η/s min	Shear viscosity at T_c	0–0.3
η/s slope	Slope above T_c	0–2 GeV^{-1}
ζ/s norm	Prefactor for $(\zeta/s)(T)$	0–2
T_{switch}	Particlization temperature	135–165 MeV

- Bayesian method

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}$$

X: model —— Y: data

$10^4 \sim 10^7$ events

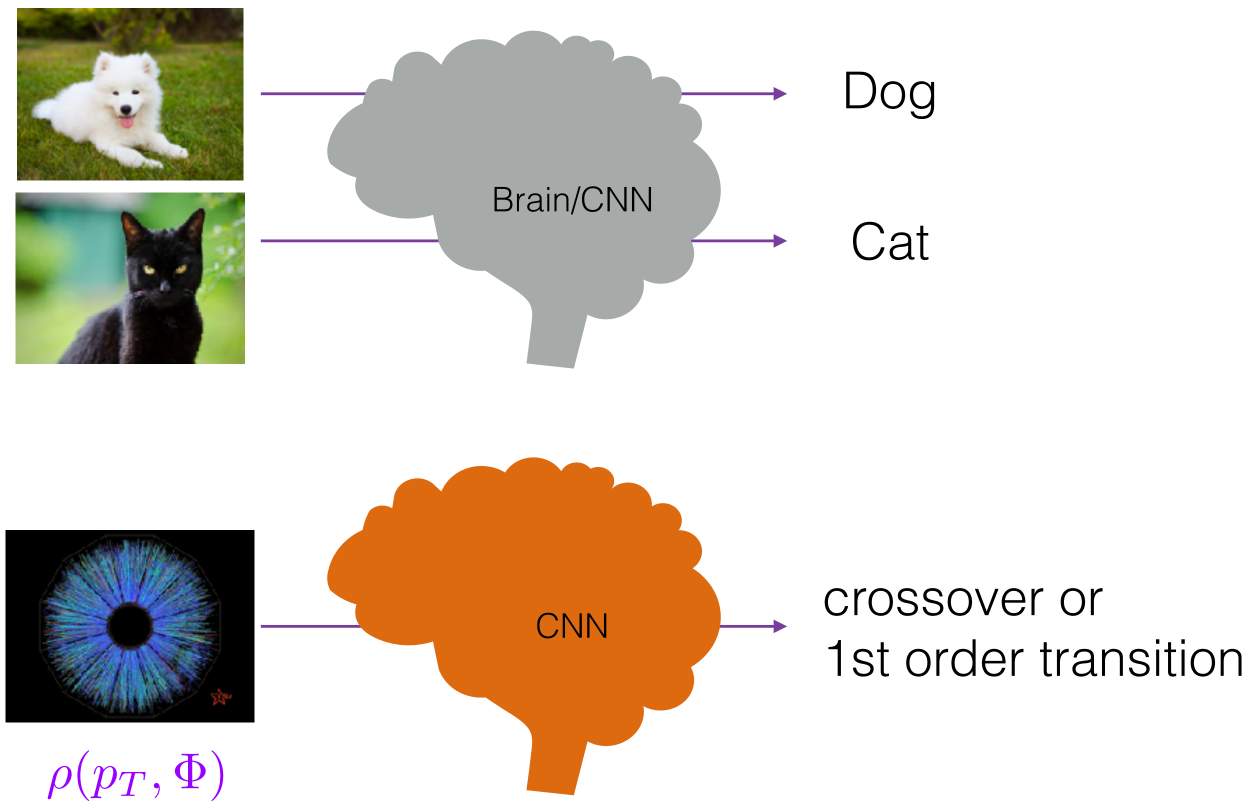
PRC 94.024907, J.E.Bernhard. et.el.

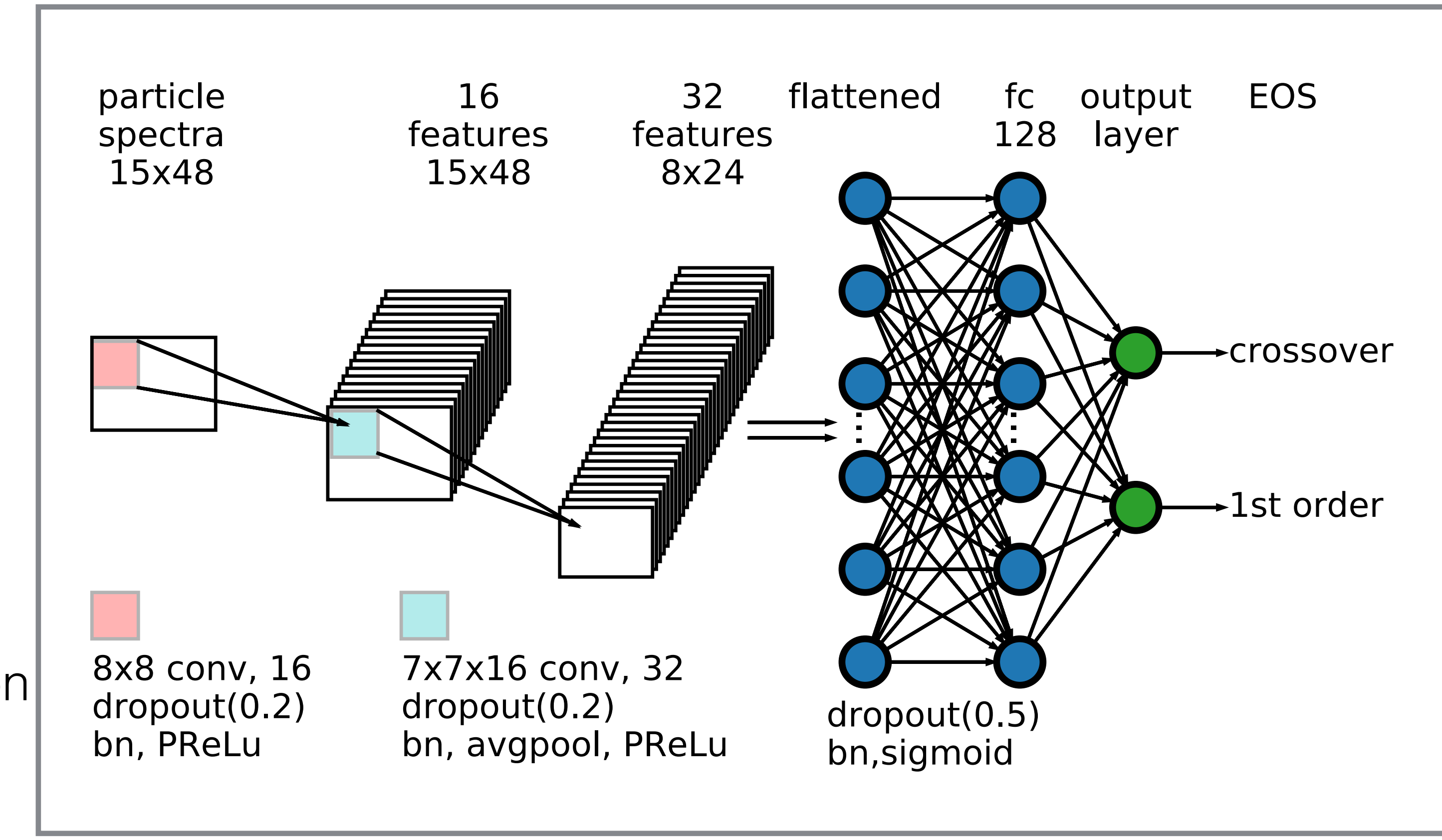
PRL. **114**, 202301, S. Pratt, et.el

Big-data in heavy ion collisions (Deep Convolution Neural Network)

<u>An EoS-meter of QCD transition from deep learning</u>

Long-Gang Pang, K.Zhou, N.Su, H.Petersen, H.Stocker and X.-N.Wang, arxiv:1612.04626v2



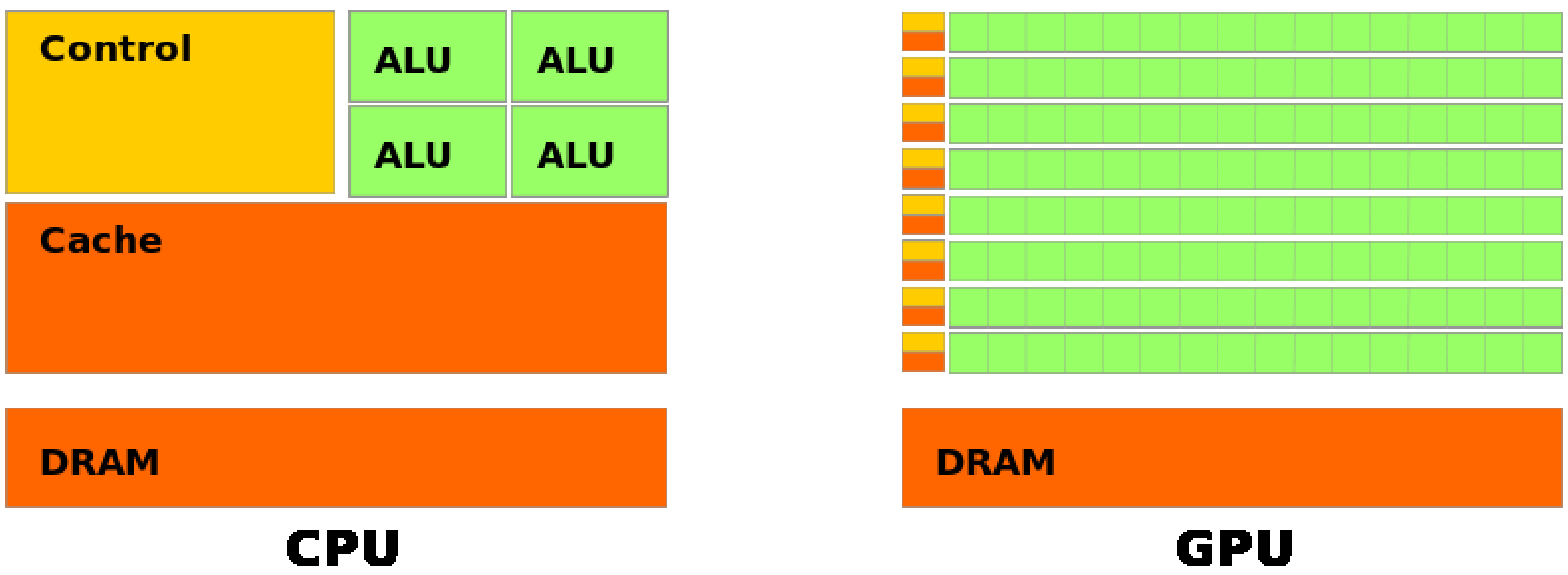


- $O(10^4)$ events from CLVisc and iEBE-VISHNU (by C.Shen, Z.Qiu, H.C.Song, J.Bernhard, S.Bass, U.Heinz) are used, more are needed in the future for other studies.

- 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).

Graphics Processing Unit (GPU) for parallelization

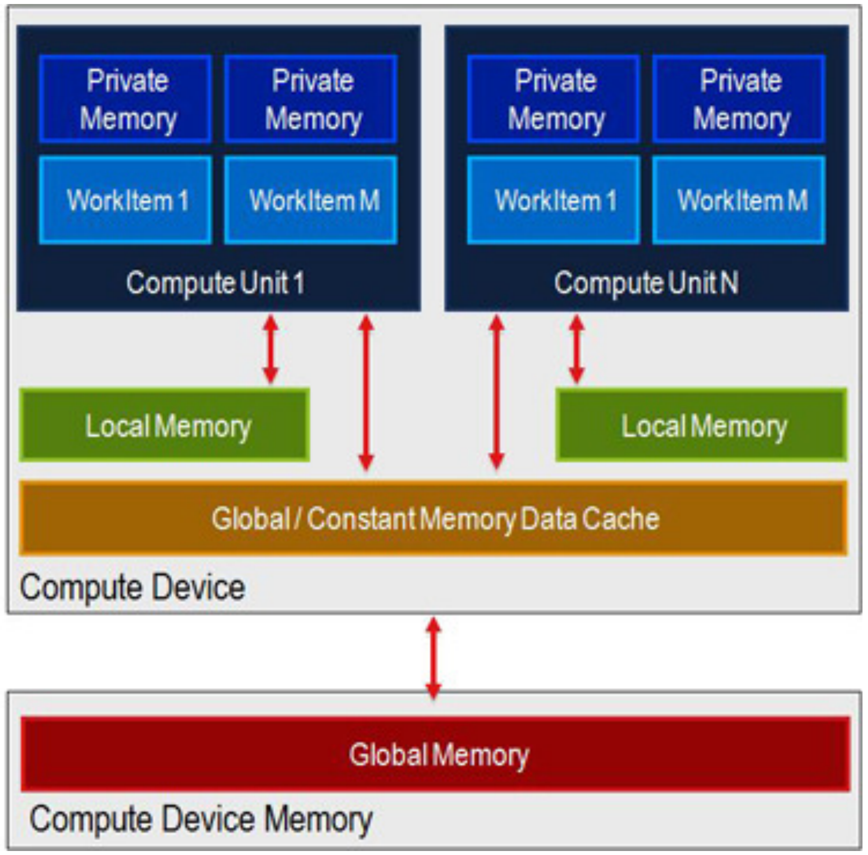
Fig from Nvidia company



- 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)

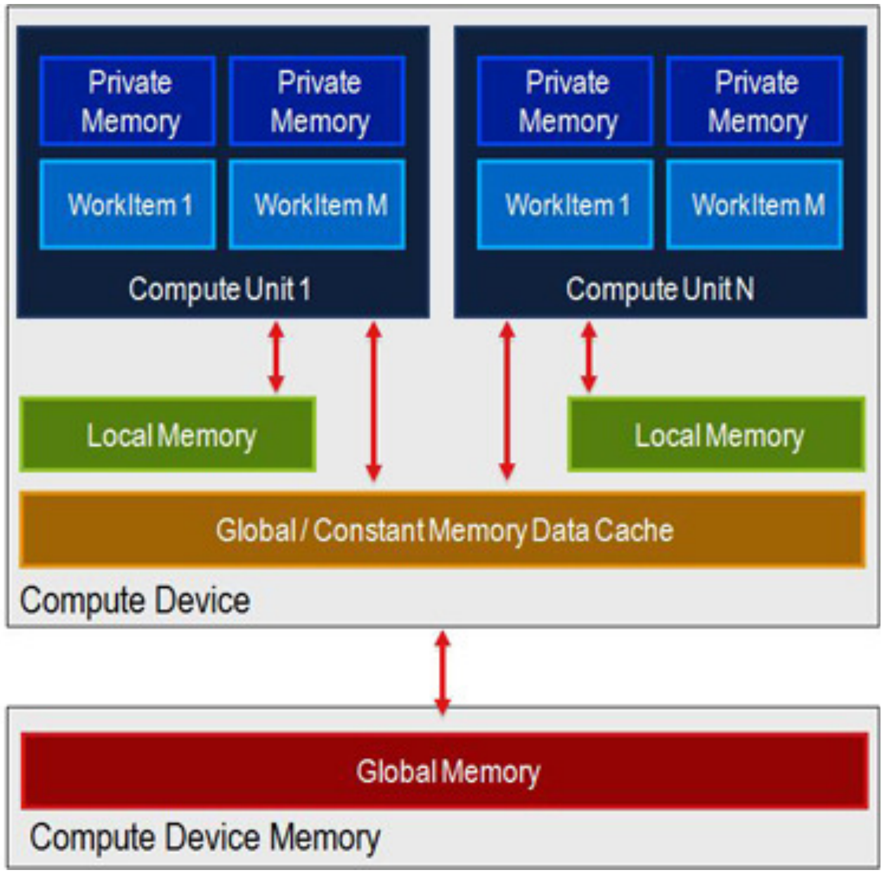
GPU memory (Global memory)



Global Memory

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

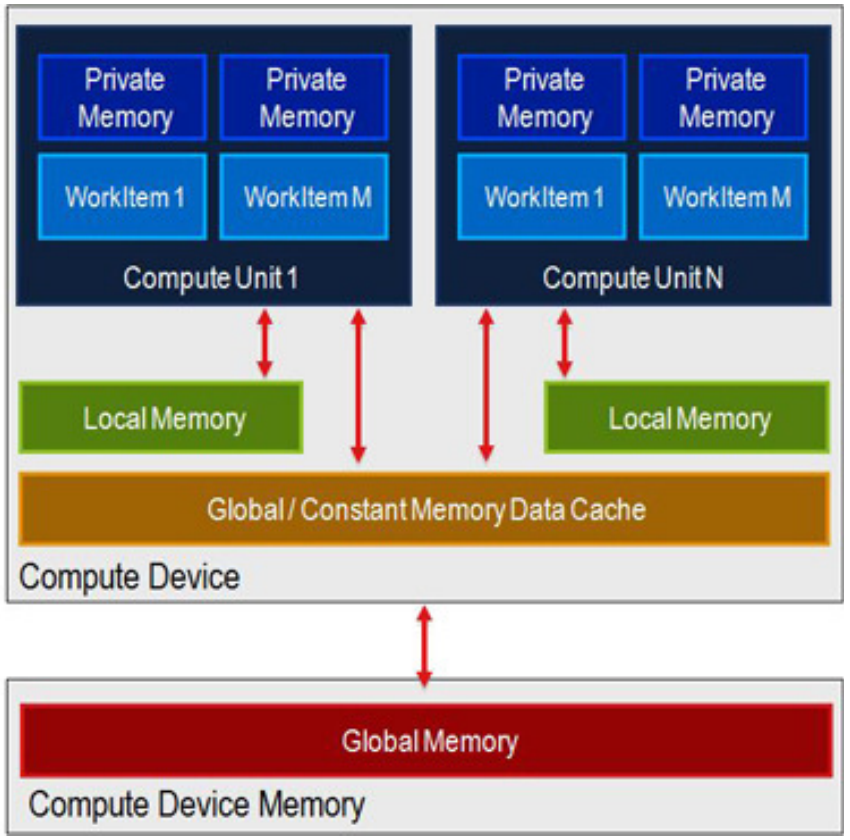
GPU memory (Local memory)



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

GPU memory (private memory)

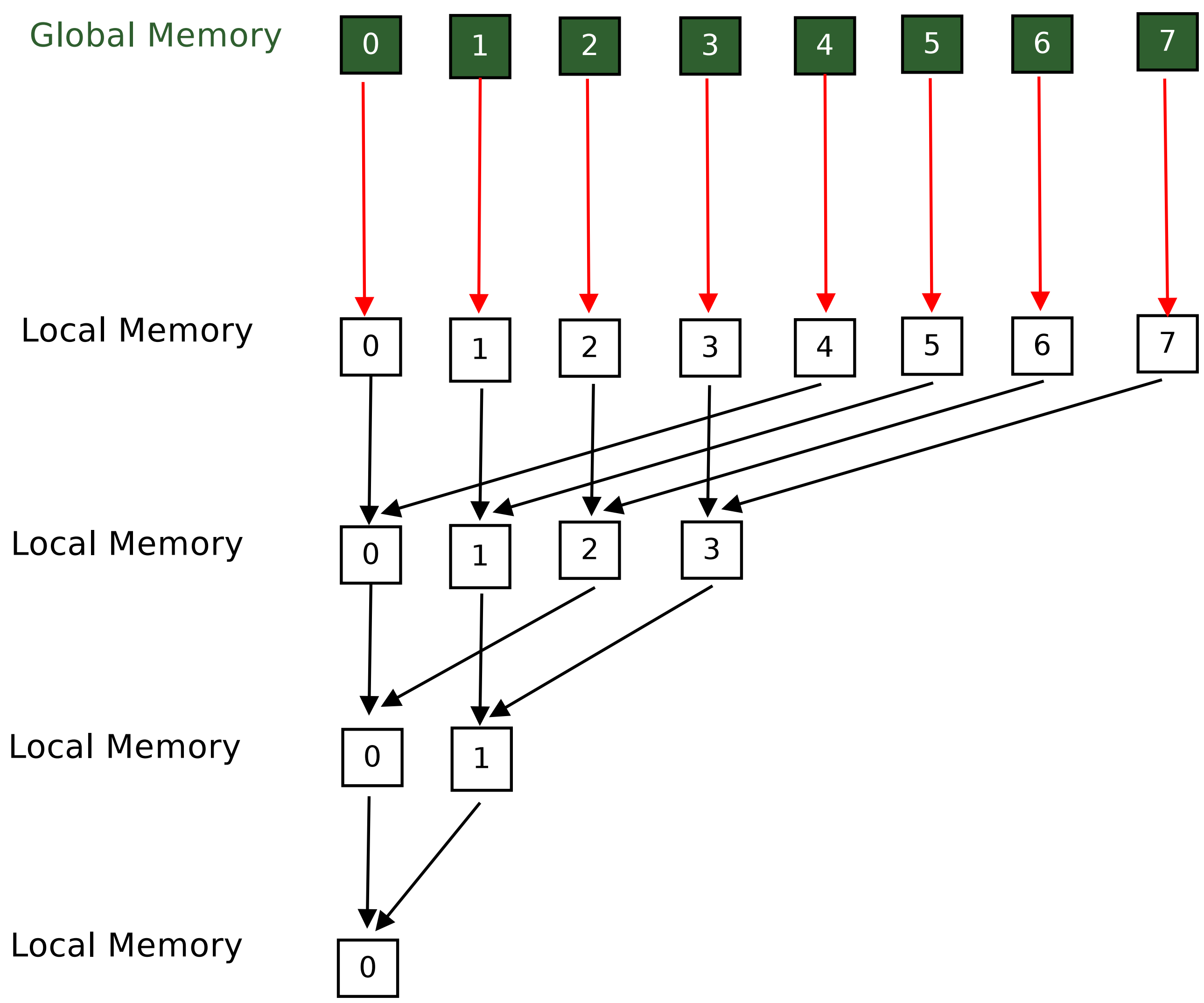


Private memory

- 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

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

- 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(ϕ) bins.
- More than 300 resonance particles.
- For each event, needs to calc. exp function $200,000 * 41 * 15 * 48 * 300$ times.

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

	CPU (i5-430M)	GPU (GT-240M)	GPU (K20)
Smooth spec. for π^+	7 minutes	30 seconds	0.5 seconds

Table : GPU(48 cuda cores) in my laptop is 10-30 times faster than CPU. NVIDIA K20 GPU has 2496 cuda cores.

OpenCL vs Cuda vs OpenAcc

CLVisc, L.G. Pang, B.W. Xiao, Y. Hatta, X.N.Wang, PRD 2015   CLVisc uses OpenCL

Open Computing Language (OpenCL) from wikipedia

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)

OpenACC:
Like OpenMP, pragma for loop parallelization.

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

Model (3+1D viscous hydrodynamics)

CLVisc: a (3+1)D viscous hydrodynamics parallelized on GPU using OpenCL

$$\nabla_\mu T^{\mu\nu} = 0 \tag{1}$$

$$\Delta^{\mu\nu\alpha\beta} u^\lambda \nabla_\lambda \pi_{\alpha\beta} = -\frac{\pi^{\mu\nu} - \pi^{\mu\nu}_{\rm NS}}{\tau_\pi} - \frac{4}{3}\pi^{\mu\nu}\nabla_\lambda u^\lambda \tag{2}$$

where

$$T^{\mu\nu} = (\varepsilon + P)u^\mu u^\nu - P g^{\mu\nu} + \pi^{\mu\nu} \tag{3}$$

$$\Delta^{\mu\nu\alpha\beta} = \frac{1}{2}(\Delta^{\mu\alpha}\Delta^{\nu\beta} + \Delta^{\nu\alpha}\Delta^{\mu\beta}) - \frac{1}{3}\Delta^{\mu\nu}\Delta^{\alpha\beta} \tag{4}$$

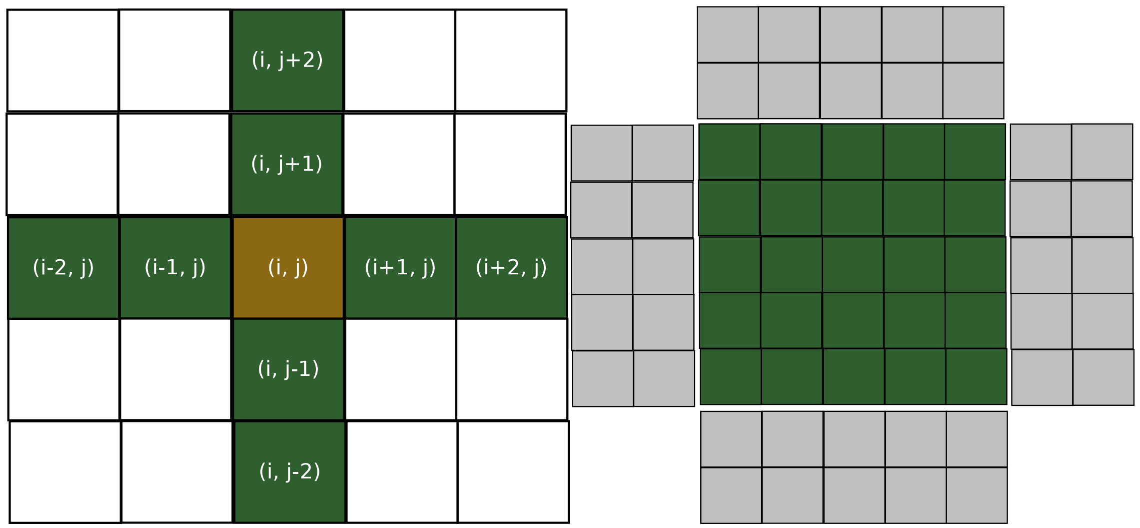
$$\Delta^{\mu\nu} = g^{\mu\nu} - u^\mu u^\nu,\ g^{\mu\nu} = diag(1, -1, -1, -\tau^{-2}) \tag{5}$$

ε and P are the energy density and pressure, u^μ is the fluid velocity vector. ∇_μ 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

KT algorithm for PDE (old implementation in CLVisc)



- In 3d, each cell shares data with 12 neighbors, better to use local memory.

Performance of KT evolution for most central Pb+Pb collisions ideal hydrodynamics

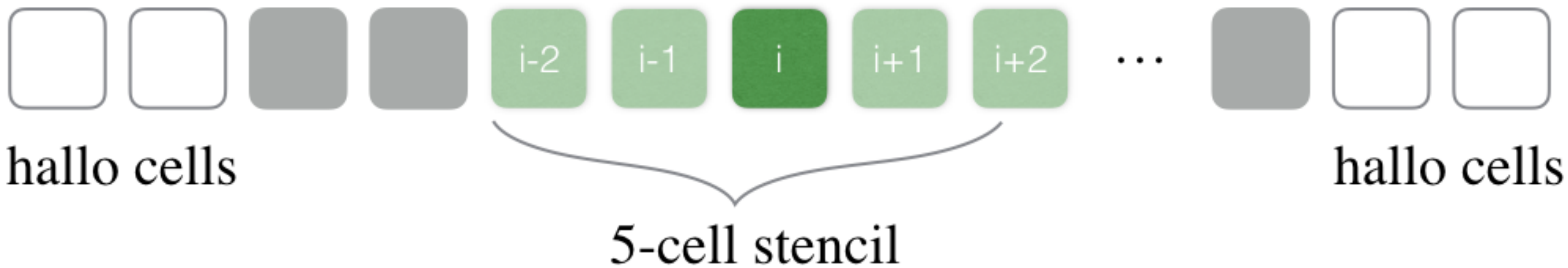
CPU (E5-2650)	K20 ($5*5*5$ block)	K20 ($7*7*7$ block)
1 hour	3 minutes	50 seconds

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

Too many halo cells, use too much local memory, difficult to implement in viscous hydro.

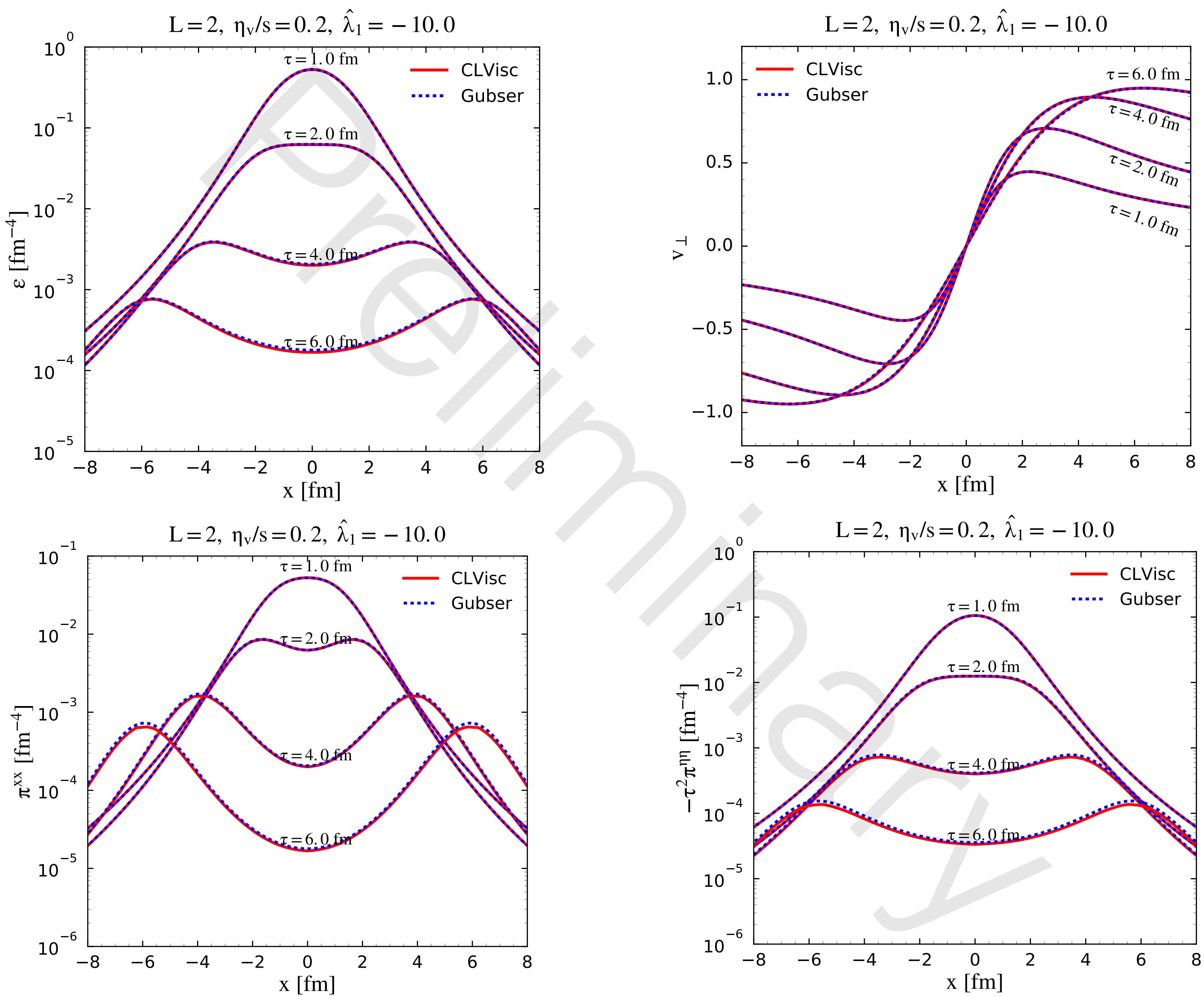
KT algorithm for PDE (new implementation in CLVisc)

$$\frac{d\bar{Q}}{d\tau} = -\frac{H^x_{i+1/2,j,k} - H^x_{i-1/2,j,k}}{dx} - \frac{H^y_{i,j+1/2,k} - H^y_{i,j-1/2,k}}{dy} - \frac{H^\eta_{i,j,k+1/2} - H^\eta_{i,j,k-1/2}}{\tau d\eta}$$

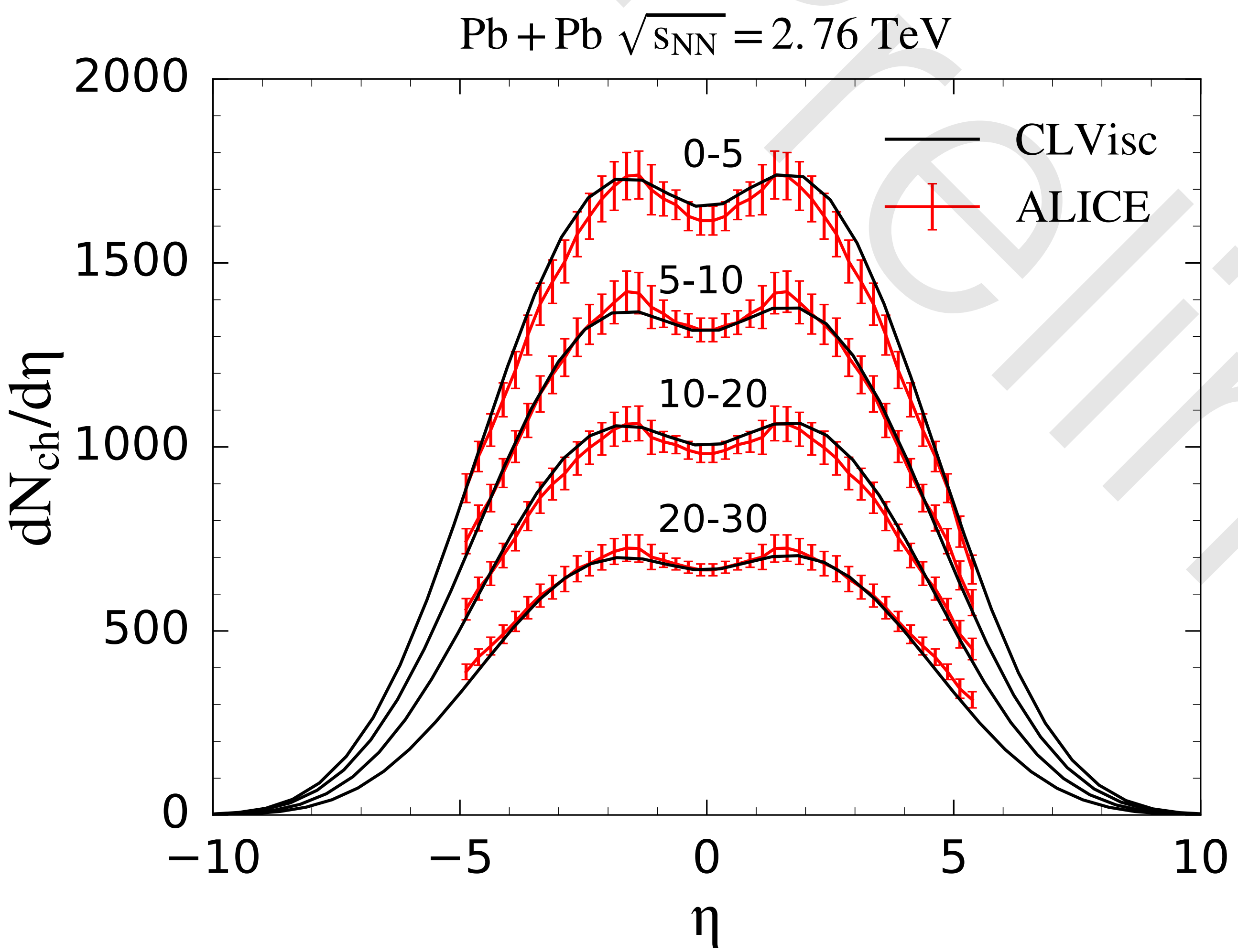


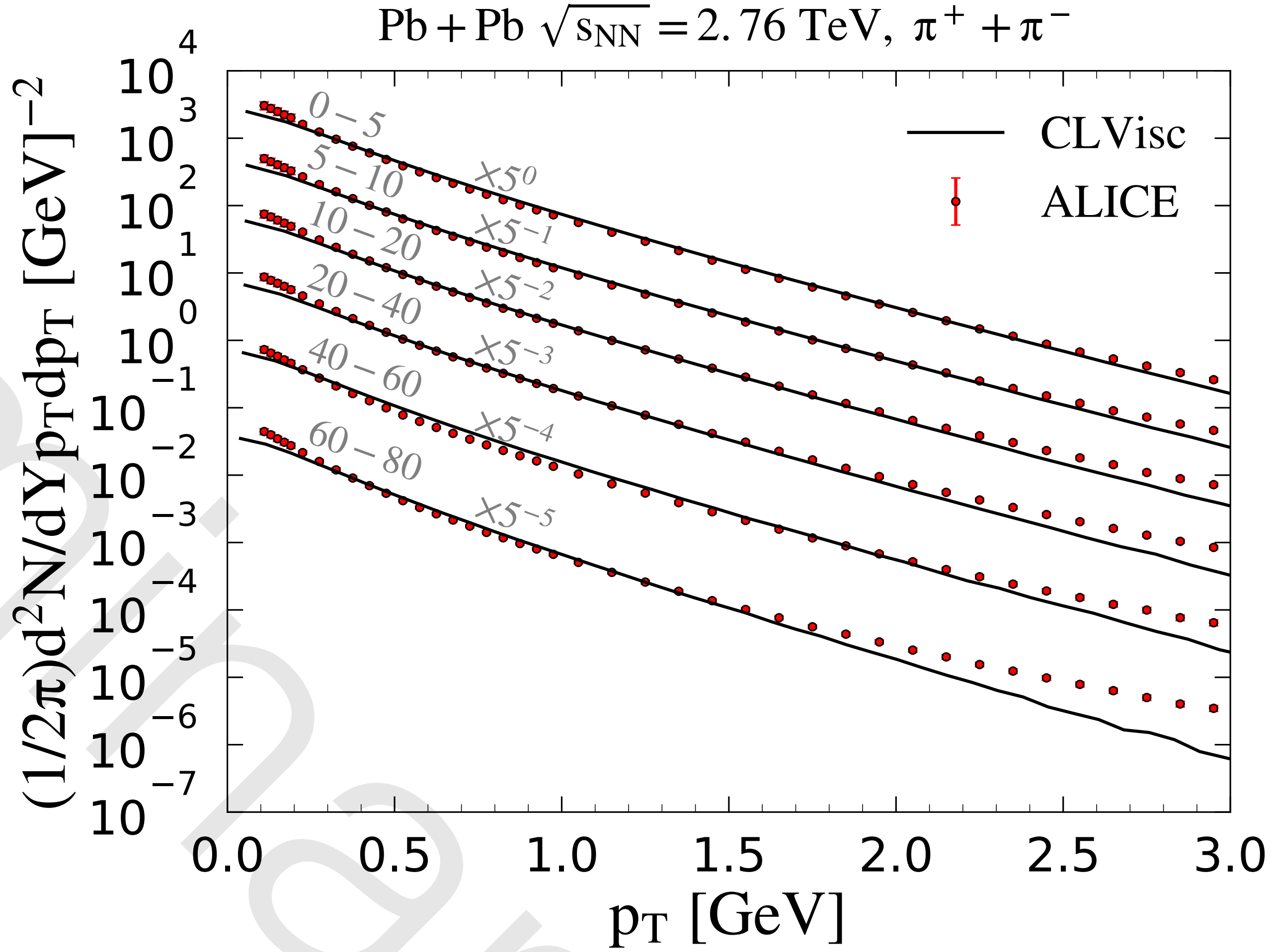
- Using dimension splitting, put each strip of data to local memory. Only 4 hallo cells in each local memory.
- Easier to implement, no performance loss.

CLVisc vs analytical Gubser solution for 2nd order viscous hydro



Particle dN/dEta and Spectra





- With Trento (Duke Group) initial condition
- Centrality ranges are the same as used in JETSCAPE.

Longitudinal de-correlation of anisotropic flows

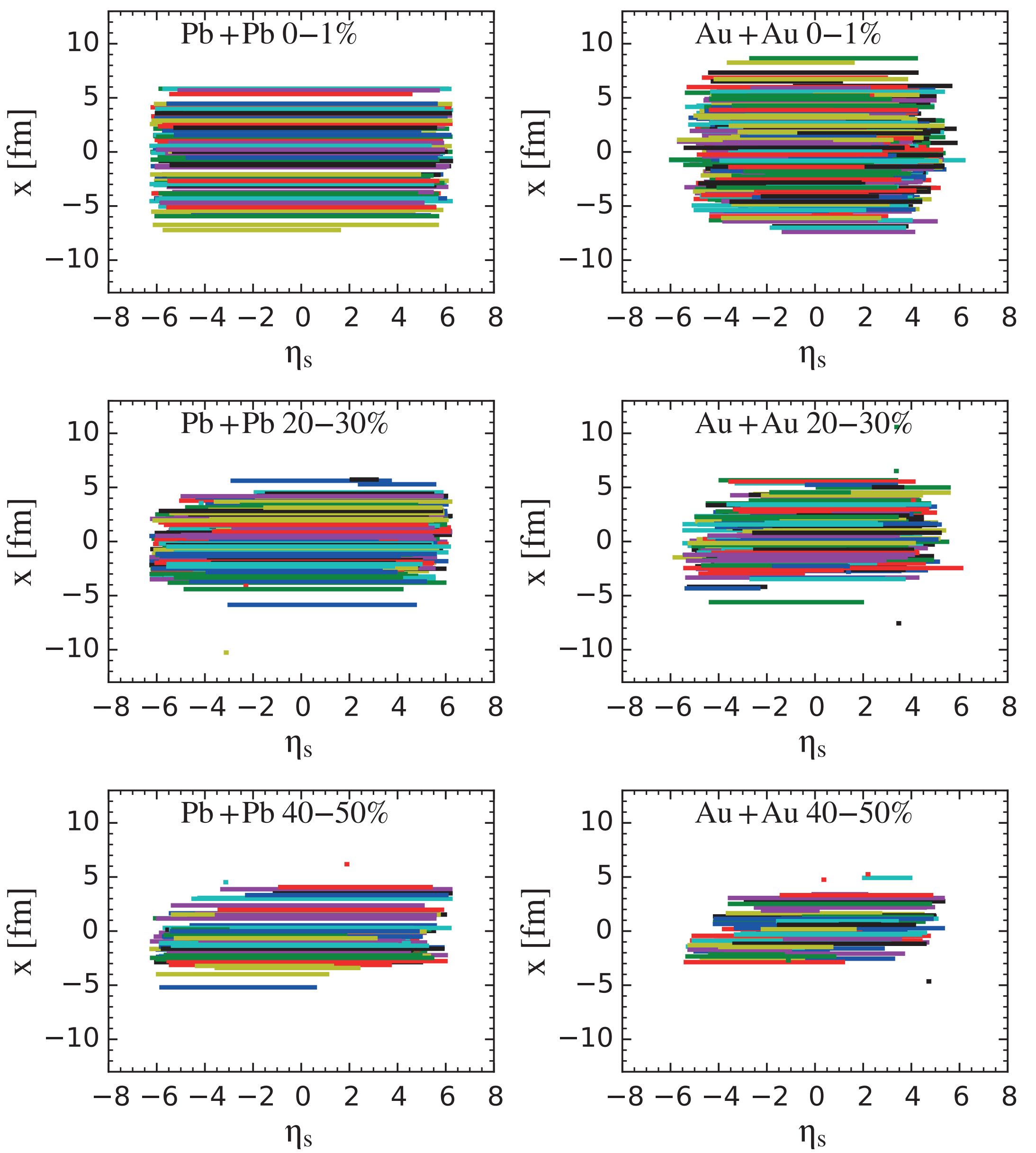
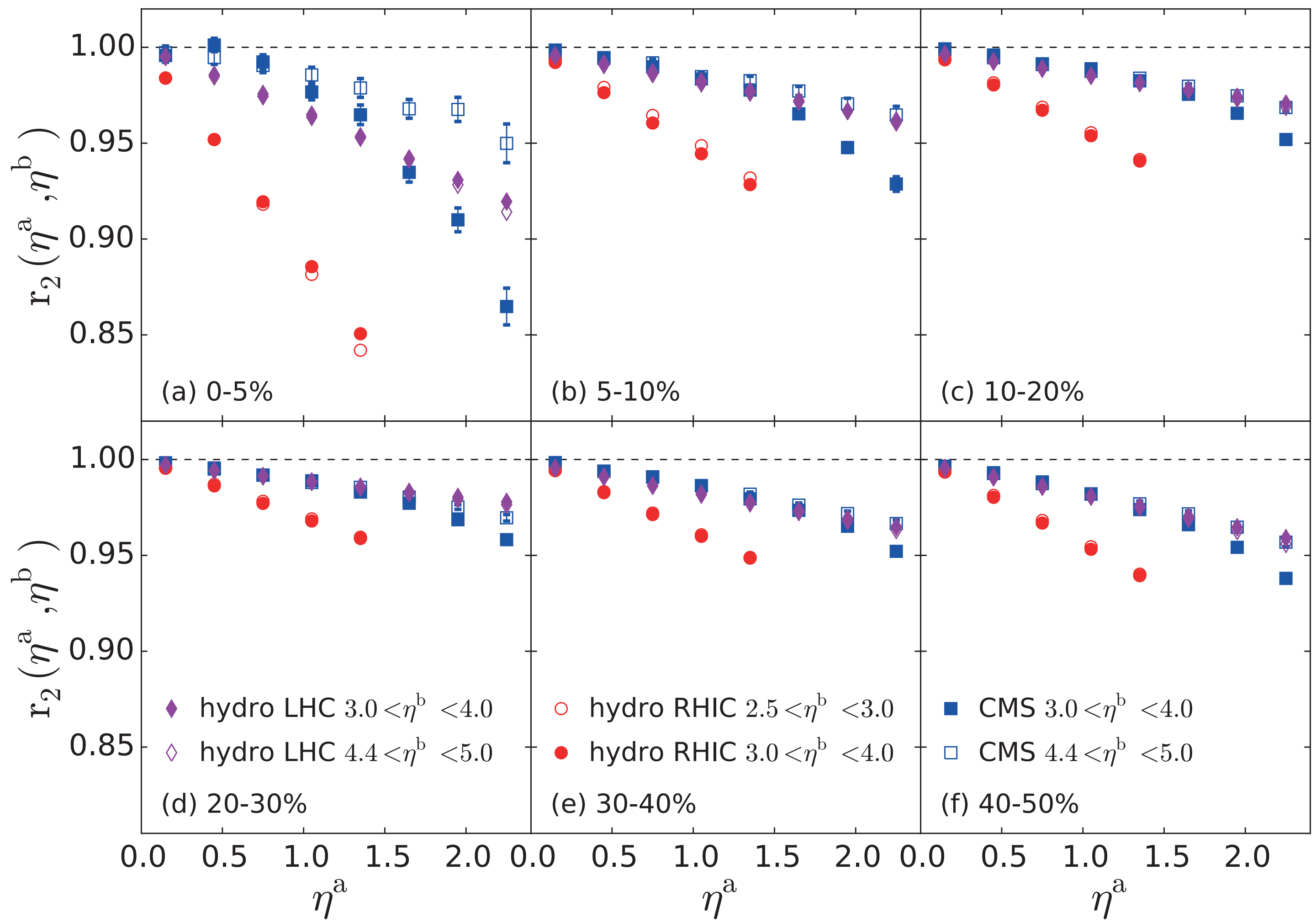


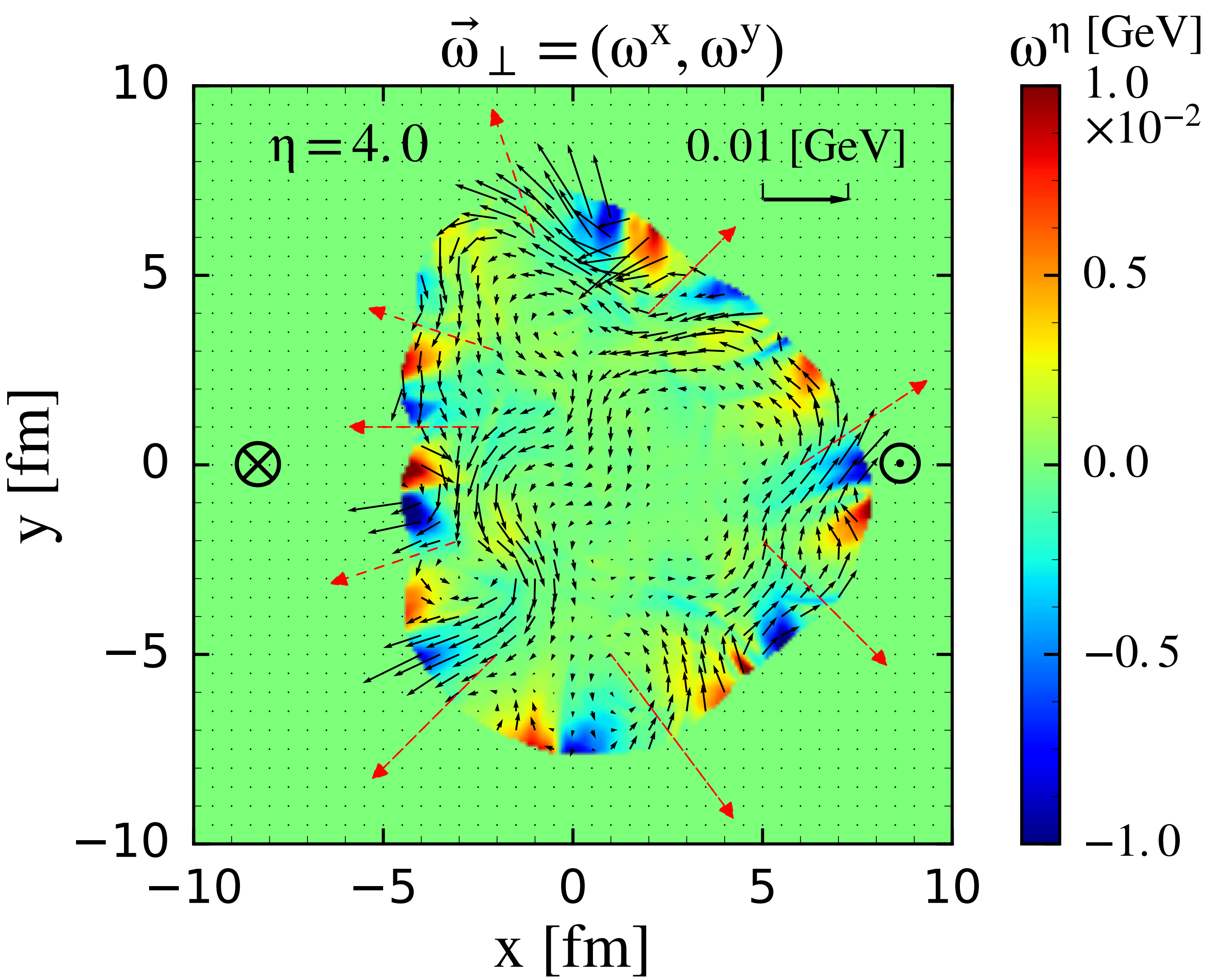
Fig. 2. (Color online) The longitudinal fluctuations for (left) Pb+Pb 2.76 TeV collisions and (right) Au+Au 200 GeV collisions for three typical events at centrality classes 0–1%, 20–30% and 40–50%.



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

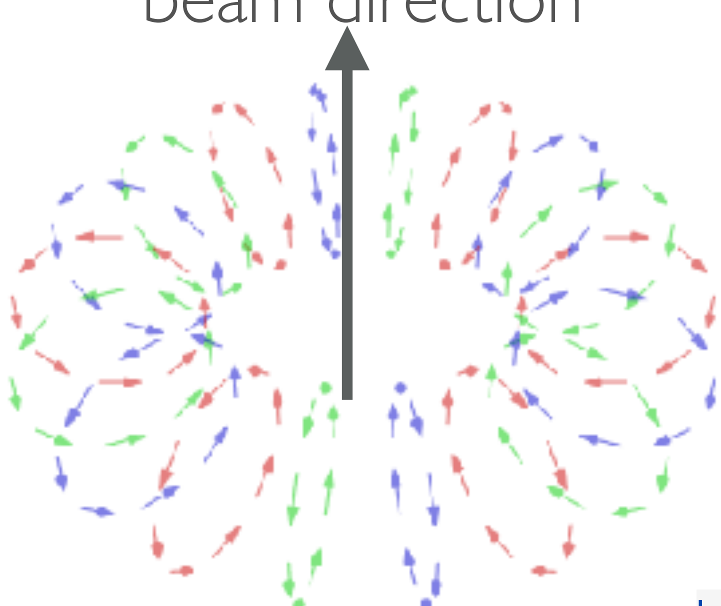
EPJA52 (2016) no.4, 97, Long-Gang Pang, H.Petersen, G.Y.Qin, V.Roy, XN Wang

Complex vortical fluid in heavy ion collisions



- 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

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

Profiling of CLVisc on GPU and CPU

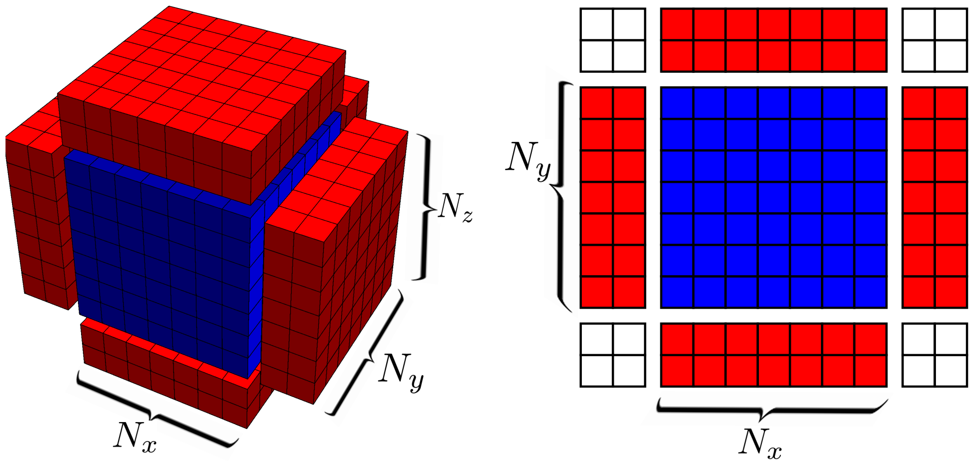
block size	8	16	32	64	128
Ideal(s)-GPU	0.37	0.218	0.178	0.155	0.157
Visc(s)-GPU	3.12	1.65	1.17	1.01	1.17
Visc(s)-CPU	6.64	6.45	6.63	7.0	7.58

- 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

- 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.

The CUDA implementation GPU-VH by Ohio group



		Clock speeds (MHz)		Memory Configuration		Processing power (GFLOPS)	
Model	Processor Cores	Core	Memory	Size (GB)	Bandwidth (GB/s)	Single precision	Double precision
GeForce GTX 560M	192	775	2500	3.076	60	595.2	N/A
GeForce GTX 980 Ti	2816	1000	7012	6.144	336	5632	176
Tesla K20M	2496	706	2600	5.120	208	3524	1175

Number of grid points	C/CPU (ms/step)	CUDA/GPU (ms/step)	Speedup
$128 \times 128 \times 32$	7690.069	96.923	79.342
$128 \times 128 \times 64$	16315.976	192.751	84.648
$128 \times 128 \times 128$	38428.056	384.255	100.007
$256 \times 256 \times 32$	30401.898	378.178	80.390
$256 \times 256 \times 64$	72240.973	744.168	97.076
$256 \times 256 \times 128$	144744.290	1485.703	97.423
$256 \times 256 \times 256$	322536.875	2970.727	108.572

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

Summary

- Big data analysis (Bayasian 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).