Monte Carlo integration and event generation on GPU and their application to particle physics

Junichi Kanzaki (KEK)

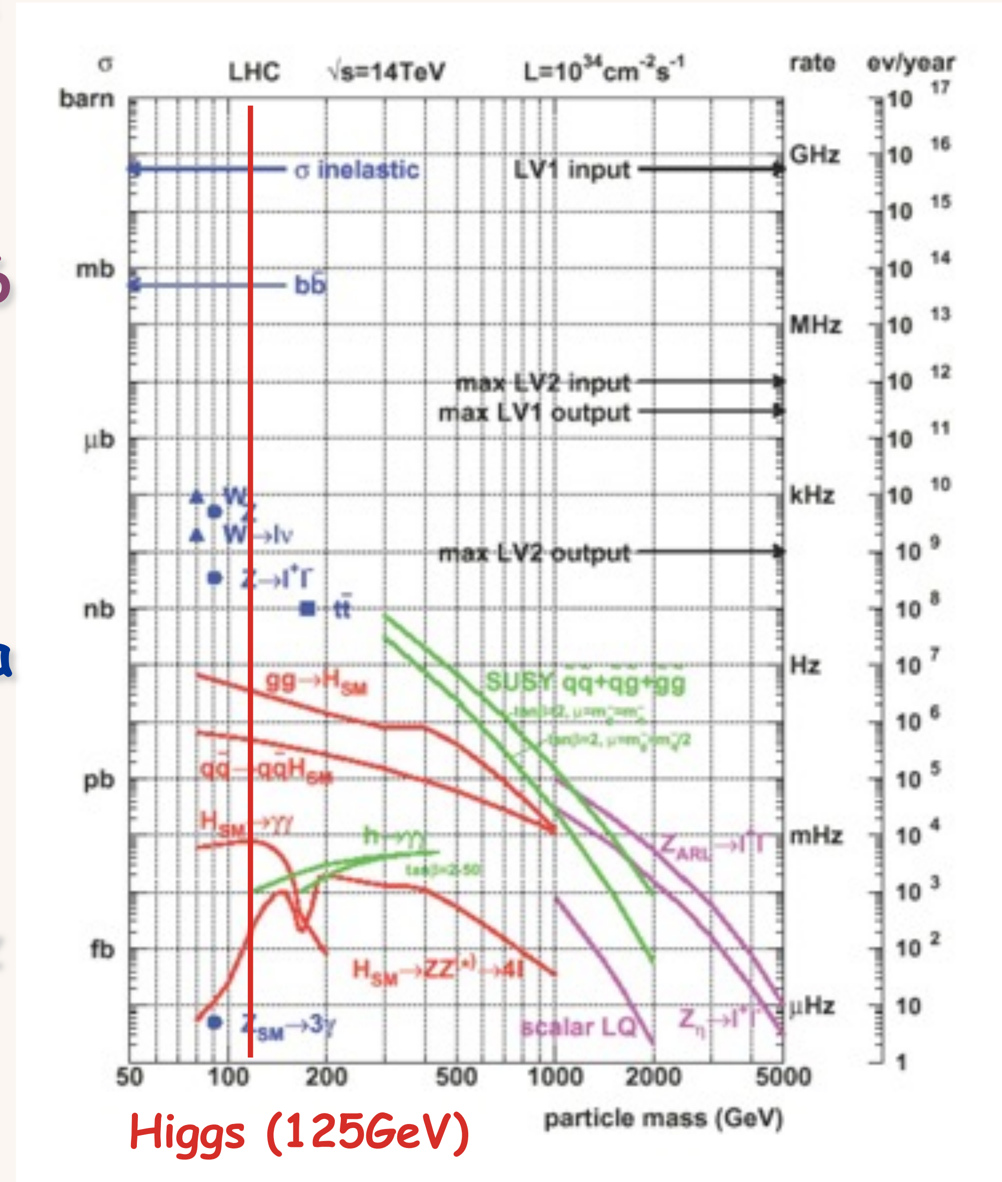
GPU2016

@ Rome, Italy

Sep. 26, 2016

Motivation

- Increase of amount of LHC data (raw & simulated events)
  - Run1: 5+20fb^{-1} up to 2012
  - Run2: 3+18fb^{-1} 2015 and 2016 >100fb^{-1} / 3 years from 2015
  - And more: 300fb^{-1} until 2022, 3000fb^{-1} until 2035
  - Huge amount of simulation data for physics analysis.
- Very high raw event rate…
  - Vector bosons (W, Z): ~100Hz
  - Top quark: ~10Hz
  - Higgs: ~0.1Hz



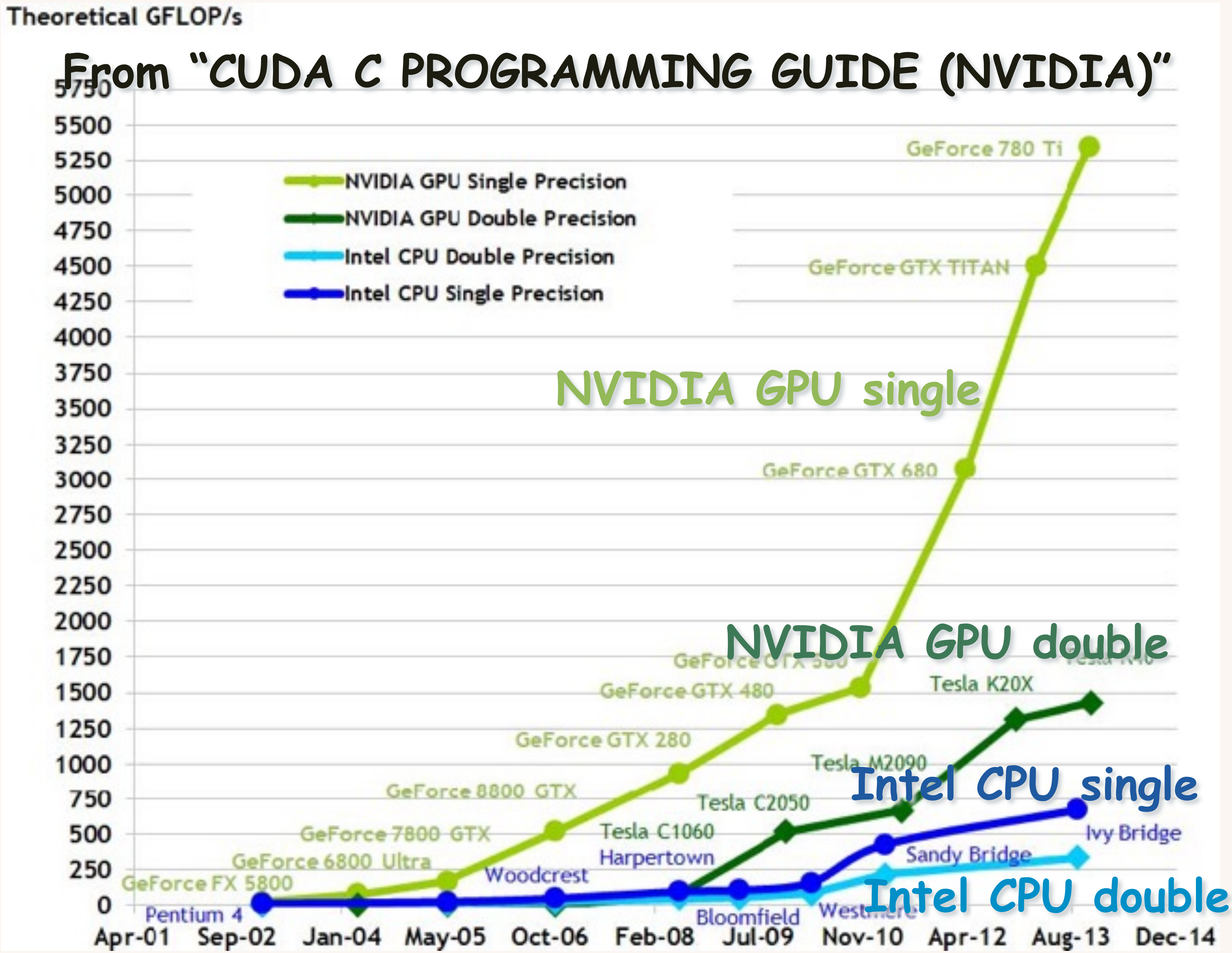
Motivation

- Official large scale event production uses "GRID": CPU and storage resources around the world.

- For coming huge data analysis still need technical innovations of High Performance Computing (HPC) in large scale data processing and also in personal analysis environments.

  -> Multi-core & many-core CPU, PC Farms, and "GPGPU".

GFLOP/s


Theoretical GFLOP/s
From "CUDA C PROGRAMMING GUIDE (NVIDIA)"
5750
5500
5250
5000
4750
4500
4250
4000
3750
3500
3250
3000
2750
2500
2250
2000
1750
1500
1250
1000
750
500
250
0
NVIDIA GPU Single Precision
NVIDIA GPU Double Precision
Intel CPU Double Precision
Intel CPU Single Precision
GeForce 780 Ti
GeForce GTX TITAN
NVIDIA GPU single
GeForce GTX 680
NVIDIA GPU double
GeForce GTX 480
Tesla K20X
GeForce GTX 280
Tesla M2090
GeForce 8800 GTX
Tesla C2050
Intel CPU single
GeForce 7800 GTX
Tesla C1060
Ivy Bridge
GeForce 6800 Ultra
Harpertown
Sandy Bridge
GeForce FX 5800
Woodcrest
Intel CPU double
Pentium 4
Bloomfield
Apr-01
Sep-02
Jan-04
May-05
Oct-06
Feb-08
Jul-09
Nov-10
Apr-12
Aug-13
Dec-14

Motivation

- Current major application of GPU in particle physics:
  - Fast trigger decision
  - Pattern recognition/reconstruction of detector data

- Not only large scale data processing and also personal physics analysis requires computing resources.

→ Fast event generation with Monte Carlo integration method

Overview

- Since 2008 we are working on the development of codes on GPU to improve performance physics event generations with MC integrations.

- With simple physics processes (QED and QCD) we learned GPU code programing with CUDA and experienced their good performance. And we extended our code to all physics processes.

Bibliography

- QED: K. Hagiwara, J. Kanzaki, N. Okamura, D. Rainwater and T. Stelzer, Eur. Phys. J. C66 (2010) 477, e-print arXiv:0908.4403.
- QCD: K. Hagiwara, J. Kanzaki, N. Okamura, D. Rainwater and T. Stelzer, Eur. Phys. J. C70 (2010) 513, e-print arXiv:0909.5257.
- MC integration (VEGAS & BASES): J. Kanzaki, Eur. Phys. J. C71 (2011) 1559, e-print arXiv:1010.2107.
- SM: K. Hagiwara, J. Kanzaki, Q. Li, N. Okamura, T. Stelzer, Eur.Phys.J. C73 (2013) 2608 (2013), e-print arXiv:1305.0708v2.

Our GPU Environment

	Titan	C2075	GTX580	GTX285	GTX280	9800GTX
CUDA cores	2688	448	512	240	←	128
Global Memory	6.1GB	5.4GB	1.5GB	2GB	1GB	500MB
Constant Memory	64KB	64KB	64KB	64KB	←	64KB
Shared Memory/block	48KB	48KB	48KB	16KB	←	16KB
Registers/block	65536	32768	32768	16384	←	8192
Warp Size	32	32	32	32	←	32
Clock Rate	0.88GHz	1.15GHz	1.54GHz	1.30GHz	←	1.67GHz
Announced	2013	2011	2010	2009		2008

- Code development: NVDIA GPUs + CUDA
- GTX Titan: Peak floating point performance 4.5TFLOPS (single), max. 1.3TFLOPS (double)

Monte Carlo integration on GPU

- Application of GPU to MC integration is straightforward: each GPU thread evaluates integrand function at each multi-dimensional space point.
- FORTRAN programs (VEGAS and BASES/SPRING) are ported to CUDA and tested using SM processes at LHC with decaying massive particles in double precision.
- BASES is based on the same algorithm as VEGAS with histograming and unweighting event generation capability by SPRING.

Example: W+jets

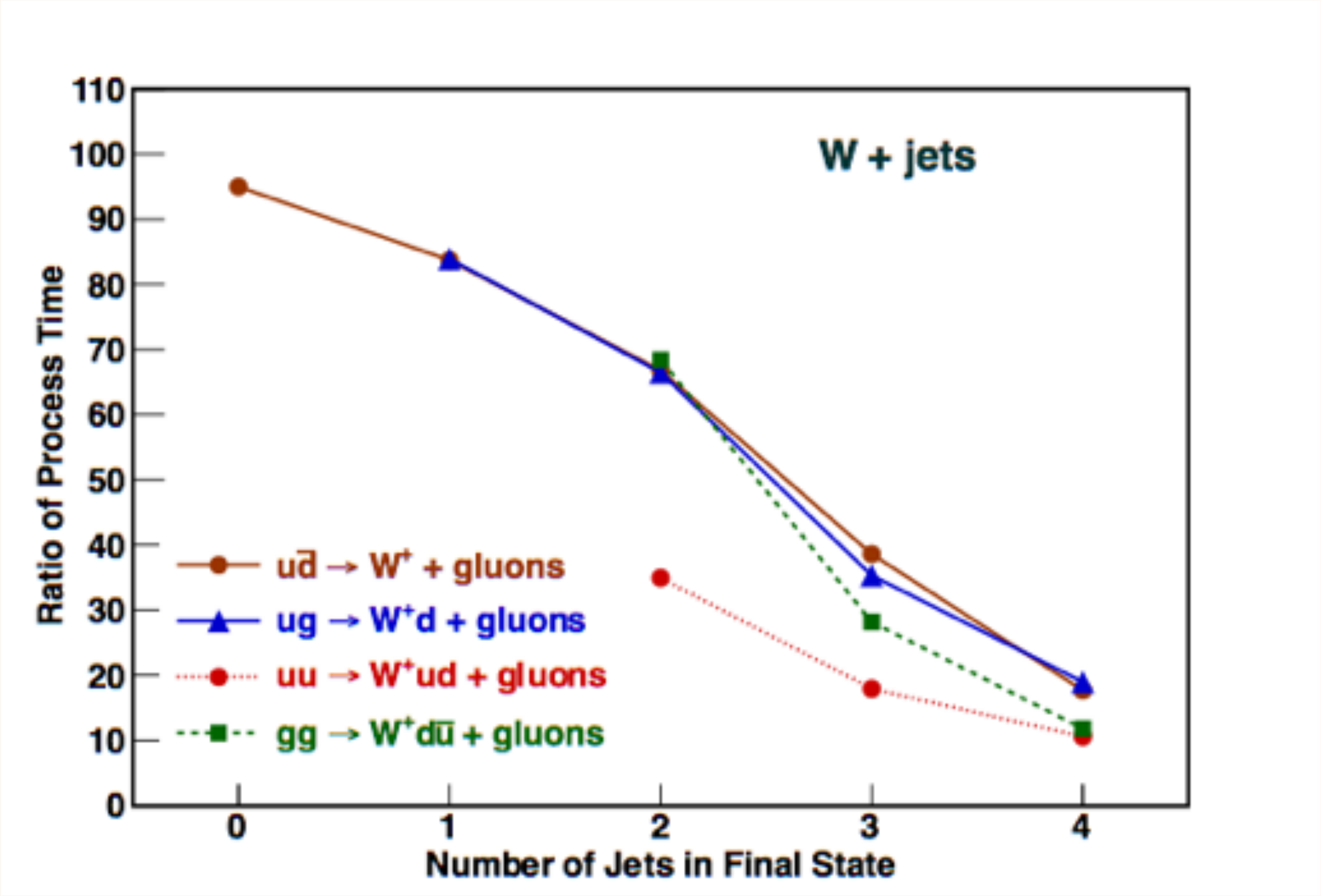
- W+jets

  u d~ -> W^+ (->l^+v) + gluons

	0-jet	1-jet	2-jet	3-jet	4-jet
ud~	W^+	W^+g	W^+gg	W^+ggg	W^+gggg
ug		W^+d	W^+dg	W^+dgg	W^+dggg
ud			W^+dd	W^+ddg	W^+ddgg
gg			W^+ud	W^+udg	W^+udgg

← 512 subgraphs

Ratio of Total Integration Time



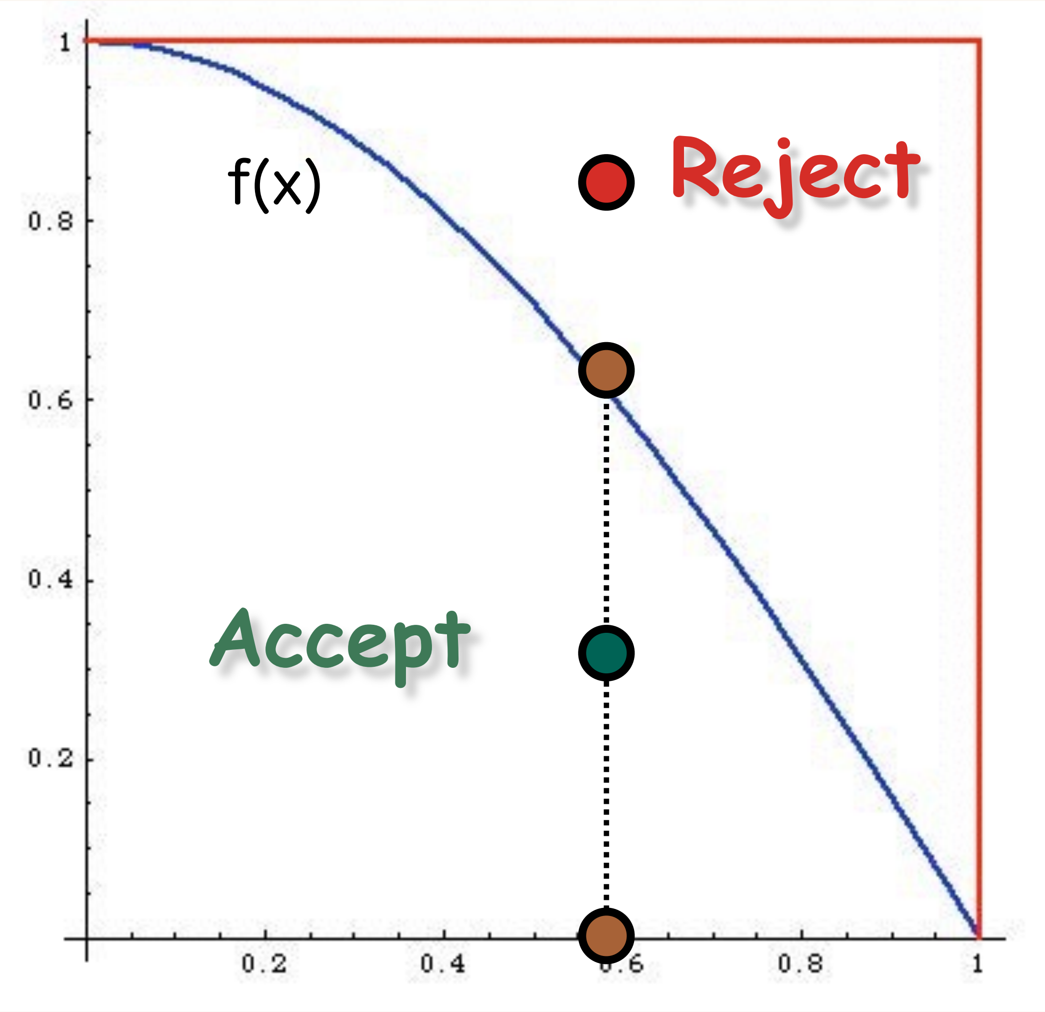
- Comparison of total execution time for equivalent BASES programs on CPU (core i7 2.7GHz, single core) and GPU (C2075) (2011).

Event Generation by SPRING

- SPRING generates unweighted events based on BASES integration results.

- One thread generates one event in a hyper-cube of multi-dimension space and unweighting step requires "acceptance-rejection":

  -> the most inefficient hyper-cell determines the total performance.



Event Generation by SPRING

- Iterative reuse of threads of GPU:
  improve unweighting algorithm by using threads as many as possible for each CUDA kernel execution.

- Threads that finished event generation can be assigned to other inefficient hyper-cell at the next iteration.

  -> improves total performance

SPRING on GPU (gSPRING)

- Allocate events to hyper cubes in the variable space in the same way as the CPU program and one processor (one thread) takes care of generation of one event.

Itr:

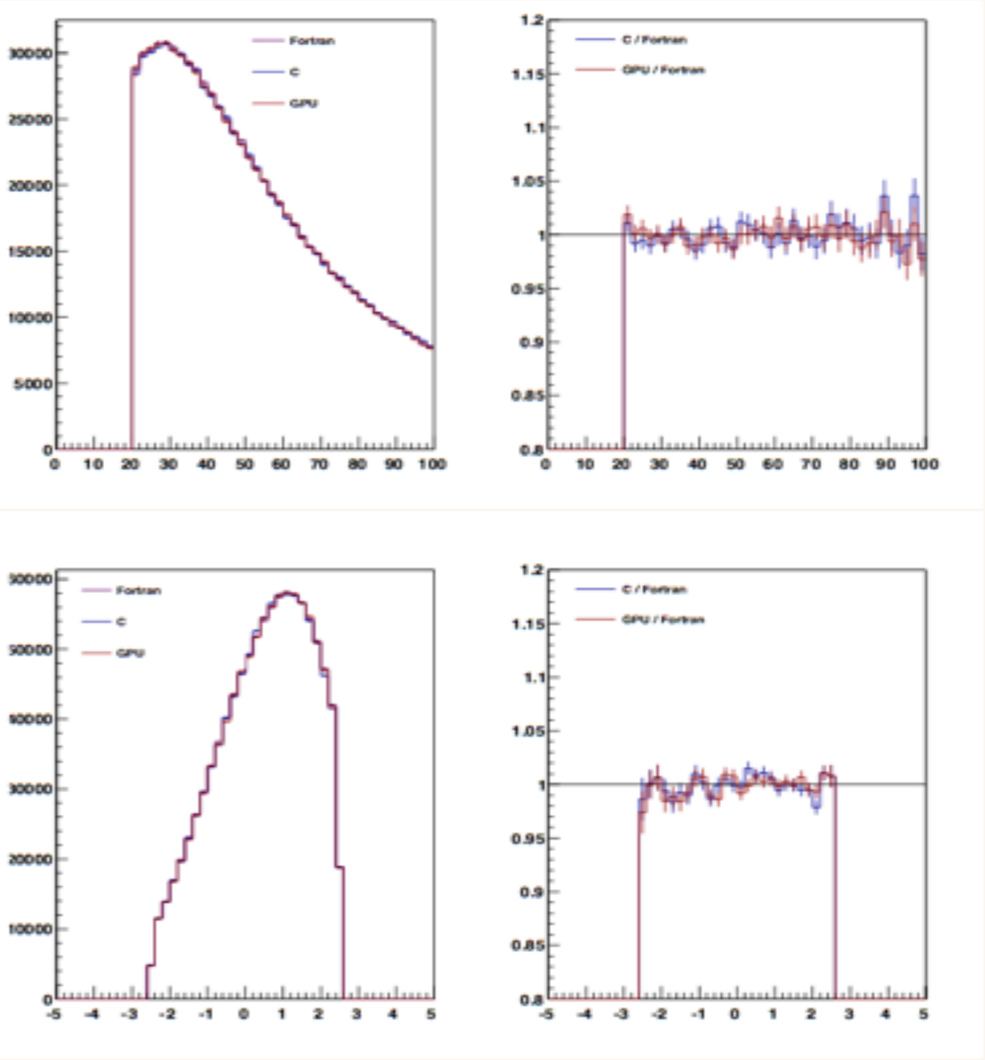
threads

#1	e1	e2	e3	e4	e5	e6	e7	e8	e9	e10	e11	e12	• • • •
	×	○	○	×	×	○	×	○	×	○	×	○	Accept/Reject
#2	e1	e1	e1	e4	e5	e5	e7	e7	e9	e9	e11	e11	Reassign threads
	×	×	×	×	×	○	○	×	○	×	○	×	
#3	e1	e1	e1	e4	e4	e4	e4	e4	e4	e4	e4	e4	• • • •

Distributions of kinematics

p_T (μ^+):

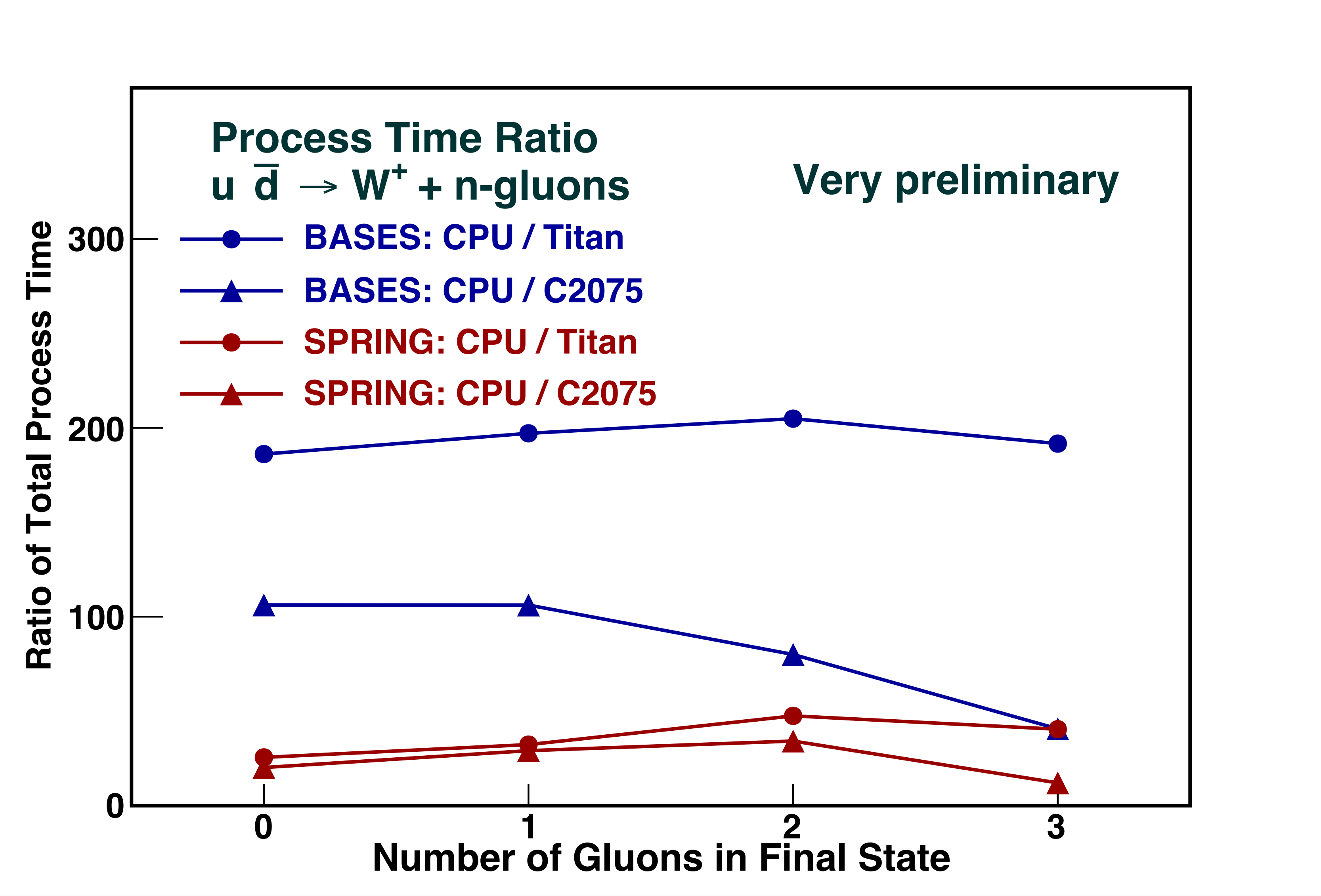
η (μ^+):



Kinematic variables of events generated with SPRING in FORTRAN, C and CUDA are compared.

Ratio of process time (C2075 & Titan)

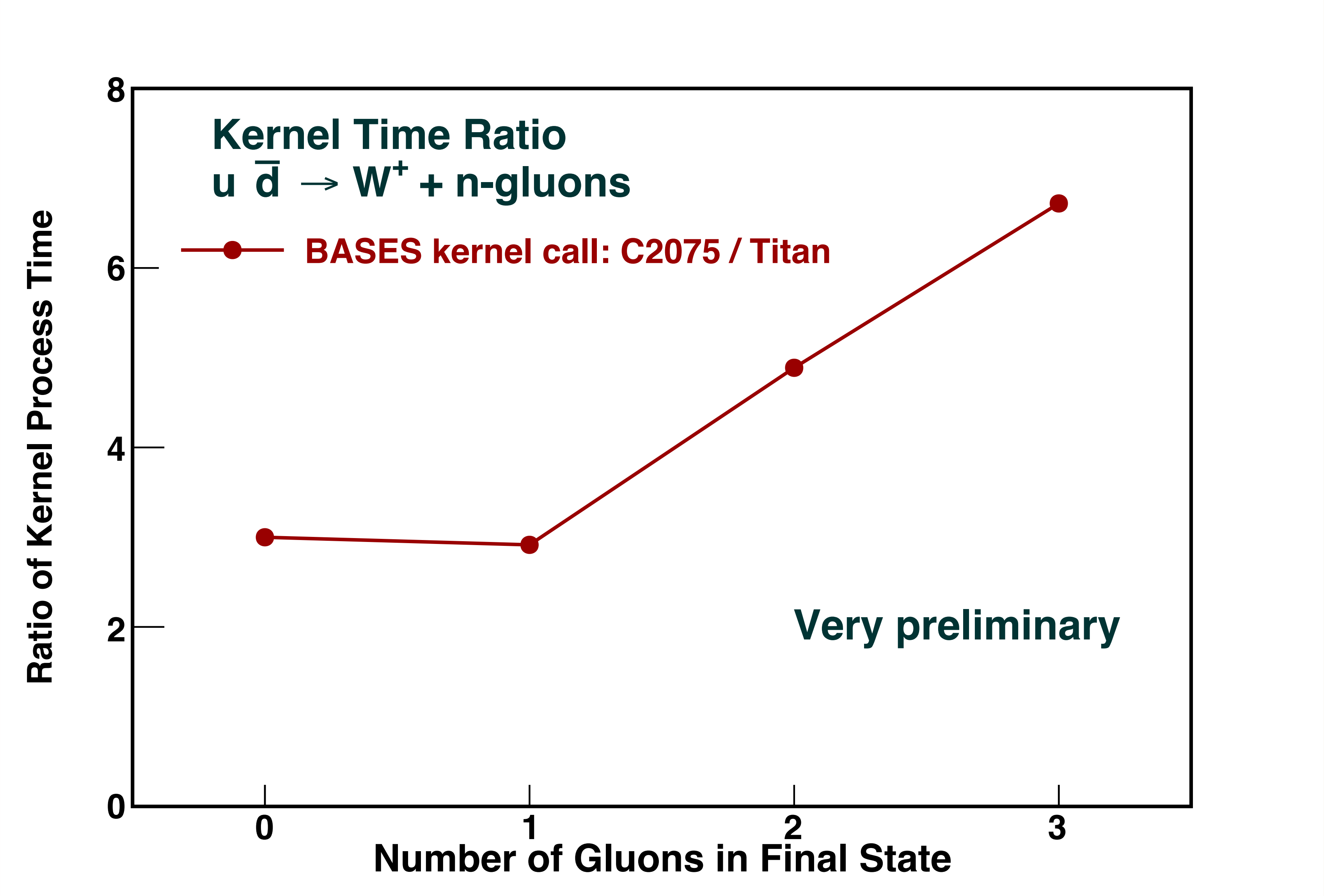
- Preliminary results on C2075 and Titan.



Presented by J. Kanzaki at GPU2016 in Sep.26, 2016

Ratio of kernel call time

- Preliminary results on C2075 and Titan.



Summary & Prospect

- Application of GPU to event simulation/generation is on going.
- Large improvements in performance of computations of cross sections and event generations are observed. Even simple parallelism works fine and optimized algorithm improves performance further.
- Optimization of new GTX Titan usage is ongoing.
- Installation of CUDA code into general event generation program, MadGraph5, is in progress.
- Hardware is improving and more applications of GPU to HEP software must be useful.
- Applications of GPU to those hat require CPU resources like GEANT (detector effect simulation) and/or ROOT (personal analysis platform) are expected.