Fast event generation system using GPU

Junichi Kanzaki (KEK)

ACAT 2013

May 16, 2013, IHEP, Beijing
Motivation

• The mount of LHC data is increasing.
  - $5 \text{fb}^{-1}$ in 2011
  - $22 \text{fb}^{-1}$ in 2012

• High statistics data
  \rightarrow Reduction of systematic errors becomes essential for good physics measurements.

• Better understandings of backgrounds from QCD multi-jet productions
  \rightarrow Fast event generation by changing model parameters
Overview

- Basic tests of HEGET (helicity amplitude library) with simple QED (n-photon) and QCD (n-jet) processes
- Development of GPU versions of VEGAS and BASES/SPRING
- Test of cross section computation and event generation with SM processes
- Summary & Prospect
Bibliography


• Event generation (SPRING): in preparation
## Our GPU Environment

<table>
<thead>
<tr>
<th></th>
<th>C2075</th>
<th>GTX580</th>
<th>GTX285</th>
<th>GTX280</th>
<th>9800GTX</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Streaming Processors</strong></td>
<td>448</td>
<td>512</td>
<td>240</td>
<td>←</td>
<td>128</td>
</tr>
<tr>
<td><strong>Global Memory</strong></td>
<td>5.4GB</td>
<td>1.5GB</td>
<td>2GB</td>
<td>1GB</td>
<td>500MB</td>
</tr>
<tr>
<td><strong>Constant Memory</strong></td>
<td>64KB</td>
<td>64KB</td>
<td>64KB</td>
<td>←</td>
<td>64KB</td>
</tr>
<tr>
<td><strong>Shared Memory/block</strong></td>
<td>48KB</td>
<td>48KB</td>
<td>16KB</td>
<td>←</td>
<td>16KB</td>
</tr>
<tr>
<td><strong>Registers/block</strong></td>
<td>32768</td>
<td>32768</td>
<td>16384</td>
<td>←</td>
<td>8192</td>
</tr>
<tr>
<td><strong>Warp Size</strong></td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>←</td>
<td>32</td>
</tr>
<tr>
<td><strong>Clock Rate</strong></td>
<td>1.15GHz</td>
<td>1.54GHz</td>
<td>1.30GHz</td>
<td>←</td>
<td>1.67GHz</td>
</tr>
</tbody>
</table>

- NVDIA GPUs + CUDA
- **C2075**: Peak floating point performance
  1.03 TFLops (single), 515 GFlops (double)
Test with QED and QCD

• Test with simple final states:
  – n-photon production (QED)
  – n-jet production (QCD)

• Development of basic components to calculate cross sections on GPU (CUDA)
  – Amplitude calculation: Heget (HELAS in FORTRAN)
  – Phase space generation
  – Random number generation

* Simple event loop program to calculated cross sections
Test with QED and QCD

• Check the total cross sections with MadGraph
• Compare process time / loop between CPU and GPU.
• Learn and experience GPU computation:
  – double/single performance ratio
  – parameter dependence of performance: register allocation, no. of threads/block
  – loop unrolling
QED Processes

- $uu\rightarrow n$-photons
- Test with two kinds of amplitude:
  - MadGraph amplitude in FORTRAN $\rightarrow$ C/CUDA
  - Amplitude by permutation of photons (short)
- Divide a long amplitude program into smaller pieces $\rightarrow$ successive kernel calls

<table>
<thead>
<tr>
<th># photons</th>
<th># diagrams = (# photons)!</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>120</td>
</tr>
<tr>
<td>6</td>
<td>720</td>
</tr>
<tr>
<td>7</td>
<td>5040</td>
</tr>
<tr>
<td>8</td>
<td>40320</td>
</tr>
</tbody>
</table>
• Large reduction of process time / event loop from CPU to GPU (single precision)
## QCD Processes

<table>
<thead>
<tr>
<th># final jets</th>
<th>gg → gluons</th>
<th>uu~ → gluons</th>
<th>uu → uu+gluons</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#diagram</td>
<td>#color</td>
<td>#diagram</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>45</td>
<td>24</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>510</td>
<td>120</td>
<td>159</td>
</tr>
<tr>
<td>5</td>
<td>7245</td>
<td>720</td>
<td>1890</td>
</tr>
</tbody>
</table>

- uu~→n-gluons, gg>n-gluons, uu>uu+gluons
- gg>5g: the program cannot be executed on GPU.
\begin{itemize}
  \item Performance degraded due to the size of amplitude and color factor multiplications.
\end{itemize}
Monte Carlo integration on GPU

• For the practical event generation on GPU
  -> GPU versions of BASES/SPRING

• Application of GPU to MC integration:
  each GPU thread evaluates function value at each space point

• Test of BASES programs using SM processes with decaying massive particles.

• Compare total process time of original FORTRAN on CPU and CUDA on GPU, and cross sections between MG5 and BASES (CPU and GPU).
SM Processes

• Decay of all massive particles:
  \( W \rightarrow l(e, \mu)\nu \), \( Z \rightarrow ll \) (e, \( \mu \)), \( t \rightarrow W(l\nu)b \), \( H \rightarrow \tau\tau \)

• Automatic conversion of MadGraph amplitude matrix.f -> CUDA functions (MG2CUDA):

• We fixed the kernel parameters:
  No. of register=64, the thread block size = 256

• Double precision computations
SM Processes

• $W, Z + \text{ up to 4 jets:}$
  - $\bar{u}d \rightarrow W^+, \ u\bar{g} \rightarrow W^+d, \, u\bar{u} \rightarrow W^+\bar{u}d, \, gg \rightarrow W^+d\bar{u}$
  - $\bar{u}u \rightarrow Z, \, u\bar{g} \rightarrow Zu, \, u\bar{u} \rightarrow Zuu, \, gg \rightarrow Zuu$

• $WW, WZ, WW + \text{ up to 3 jets:}$
  - $\bar{u}u \rightarrow W^+W^-, \, u\bar{g} \rightarrow W^+W^-u, \, u\bar{u} \rightarrow W^+W^-\bar{u},$
  - $u\bar{u} \rightarrow W^+W^+dd, \, gg \rightarrow W^+W^-uu$
  - $\bar{u}d \rightarrow W^+Z, \, u\bar{g} \rightarrow W^+Zd, \, u\bar{u} \rightarrow W^+Z\bar{u}d, \, gg \rightarrow W^+Zd\bar{u}$
  - $\bar{u}u \rightarrow ZZ, \, u\bar{g} \rightarrow ZZd, \, uu \rightarrow ZZuu, \, gg \rightarrow WWuu$

• $t\bar{t} + \text{ up to 3 jets:}$
  - $uu \rightarrow t\bar{t}, \, u\bar{g} \rightarrow t\bar{t}u, \, uu \rightarrow t\bar{t}uu, \, gg \rightarrow t\bar{t}$
SM Processes (contn'd)

• HW, HZ + up to 3 jets:
  - ud ~ > HW⁺, ug > HW⁺d, uu > HW⁺ud, gg > HW⁺du~
  - uu ~ > HZ, ug > HZu, uu > HZuu, gg > HZuu~

• Httx + 2 jets: uu ~ > Htt~, ug > Htt~u, uu > Htt~uu, gg > Htt~

• H(WBF) + 2 jets: ud > Hud, uu > Huu, ug > Hudd~, gg > Huu~dd~

• HH + up to 3 jets: ud ~ > HHud, uu ~ > HHuu

• HHH + up to 2 jets: ud ~ > HHHud, uu ~ > HHHuu
• Comparison of total execution time with double precision.
Event Generation by SPRING

- Generate unweighted events by BASES results

- One thread generates one event in a certain hyper-cell of multi-dimension space (acceptance-rejection):
  \[\rightarrow\] the most inefficient hyper-cell determines the total process time

- Iterative reuse of threads:
  threads that have finished event generation can be assigned to inefficient hyper-cell at the next iteration
  \[\rightarrow\] improves total performance
**SPRING performance**

- Total execution time [sec]: generation of unweighted $10^6$ events

<table>
<thead>
<tr>
<th>No. of gluons</th>
<th>FORTRAN</th>
<th>GTX580</th>
<th>CPU/GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>9.72</td>
<td>0.346</td>
<td>28</td>
</tr>
<tr>
<td>1</td>
<td>43.2</td>
<td>0.768</td>
<td>56</td>
</tr>
<tr>
<td>2</td>
<td>4224.8</td>
<td>26.53</td>
<td>160</td>
</tr>
</tbody>
</table>

* Preliminary test in single precision

Large improvement is expected for processes with more particles in its final state.
Summary & Prospect

• Program components of cross section computation and event generation based on MadGraph system can be executed on GPU with high performance:
  - GPU version of VEGAS and BAES/SPRING

• Improvement factor of performance can become between 10~100 for total execution time of BASES integration.

• Large improvement of SPRING can be expected.

* Hardware is improving and more applications of GPU to HEP software should be useful.