GPU STRIP CLUSTERING

STATUS UPDATE

Weekly Group Meeting
Sep 20, 2019
Motivation:

- MkFit is a parallelized KF-based code that does particle tracking and reconstruction on multicore CPUs and manycore GPUs
- MkFit needs all hits as an input
- Currently, all hits needed by MkFit are obtained from a CPU code that performs strip unpacking and clustering **serially**

Goal:

- Investigating strip unpacking and clustering on GPU
- Could also use the GPU to produce hits in the MkFit data format
• Developed a parallel version of the clustering algorithm
  1. Find out all seed strips (Seed strip has ADC > 3*noise)
  2. Keep only one seed strip if there are multiple seed strips with consecutive strip id (Potential issue: we might still have duplicated clusters generated from non-consecutive strips)
  3. For all non-consecutive seed strips, find the left and right boundary of the candidate clusters
  4. For each candidate cluster, check if it is a true cluster. If so, do some adjustment for its ADC value. A true cluster satisfies:
     • Squared noise is large enough
     • chargePerCM is large enough
Current Progress (2)

- Implemented the parallel version on multicore CPU with OpenMP and manycore GPU with CUDA
  - All data are stored as Structure-of-Array (SoA) layout
  - Prefix sum on GPU is implemented using CUB library
- Source code is available at https://github.com/beiwang2003/strip_clustering_gpu
First Performance Comparison

- Evaluated system: Tigergpu at Research Computing, Princeton University
- Host CPU: 2.4 GHz Xeon Broadwell E5-2680 v4, 14 cores per socket
- GPU: 1328 MHz P100

<table>
<thead>
<tr>
<th></th>
<th>setSeedStripIndex (s)</th>
<th>clustering (s)</th>
<th>CPU to GPU mem transfer (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU: 1 threads</td>
<td>5.47e-3</td>
<td>1.67e-2</td>
<td>0</td>
</tr>
<tr>
<td>CPU: 2 threads</td>
<td>4.20e-3</td>
<td>1.02e-2</td>
<td>0</td>
</tr>
<tr>
<td>CPU: 4 threads</td>
<td>3.62e-3</td>
<td>5.89e-3</td>
<td>0</td>
</tr>
<tr>
<td>CPU: 8 threads</td>
<td>3.03e-3</td>
<td>3.68e-3</td>
<td>0</td>
</tr>
<tr>
<td>CPU: 14 threads</td>
<td>2.83e-3</td>
<td>3.24e-3</td>
<td>0</td>
</tr>
<tr>
<td>GPU</td>
<td>9.90e-4</td>
<td>3.37e-4</td>
<td>8.15e-4</td>
</tr>
</tbody>
</table>
Future Work

• Performance analysis and optimization for both CPU and GPU implementation
  • CPU: Use Intel Advisor and VTune to investigate performance bottleneck with respect to multicore scaling and vectorization
  • GPU: Use NVProf to understand and potentially improve kernel performance (e.g., texture and shared memory utilization)
• Current clustering algorithm assumes that all input data from a single event are stored in SoA format. More work is needed if the output from unpacking is in a different format.
• Fix bugs in the parallel version of the code
• Wrap the parallel implementation in C++ classes
• The ADC value of each strip must be \( > 2 \times \text{noise} \)
• At least one strip has ADC value \( > 3 \times \text{noise} \)
• The quadrature sum of ADC values of the cluster must be \( \geq 5 \times \text{the quadrature sum of the noise} \)
• The strips are consecutive or have gap of at most N strips (N is a input parameter)