NEW DEVELOPMENTS IN 2016
https://bitbucket.org/nsakharnykh/hpgmg-cuda

Updated to include 4\textsuperscript{th} order implementation on GPU

Optimized setup phase by porting remaining routines to GPU

Updated levels allocation to improve Unified Memory performance

Better multi-GPU scaling using CUDA-aware MPI with GPUDirect P2P

GPU memory oversubscription study (Parallel Forall blog post is pending)

GPUDirect Async implementation (https://github.com/e-ago/hpgmg-cuda-async)
HYBRID IMPLEMENTATION
Data sharing between CPU and GPU

Level N

Level N+1

Smoother

Residual

Restriction

Memory

GPU kernels

CPU functions

Level N+1 (small) is shared between CPU and GPU

To avoid frequent migrations allocate N+1 in zero-copy memory
PERFORMANCE ON TESLA P100

Application throughput (MDOF/s)

- **default**
- **zero-copy opt**

<table>
<thead>
<tr>
<th>CPU Configuration</th>
<th>2xBroadwell</th>
<th>Tesla K40</th>
<th>Tesla P100</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
<td>2x Intel E5-2697 v4 @ 2.3GHz (Broadwell-EP)</td>
<td>3.6GHz Turbo (Broadwell-EP)</td>
<td>HT off (36 cores)</td>
</tr>
<tr>
<td><strong>GPU</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Tesla K40</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Tesla P100</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
HPG MG AMR PROXY

Data locality and reuse of AMR levels
OVERSUBSCRIPTION RESULTS

- x86
- K40
- P100 (x86 PCI-e)
- P100 (P8 NVLINK)

All 5 levels fit in GPU memory

P100 memory size (16GB)

- Only 2 levels fit
- Only 1 level fits

x86 CPU: Intel E5-2630 v3, 2 sockets of 10 cores each with HT on (40 threads)
Prefetch next level while performing computations on current level

Use `cudaMemPrefetchAsync` with non-blocking stream to overlap with default stream

<table>
<thead>
<tr>
<th>compute</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>4</th>
<th>3</th>
<th>4</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefetch</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>eviction</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
RESULTS WITH USER HINTS

- x86
- K40
- P100 (x86 PCI-e)
- P100 + hints (x86 PCI-e)
- P100 (P8 NVLINK)
- P100 + hints (P8 NVLINK)

Application working set (GB):
- 1.4
- 4.7
- 8.6
- 28.9
- 58.6

Application throughput (MDOF/s):
- All 5 levels fit in GPU memory
- Only 2 levels fit
- Only 1 level fits

P100 memory size (16GB)

x86 CPU: Intel E5-2630 v3, 2 sockets of 10 cores each with HT on (40 threads)
MULTI-GPU PERFORMANCE

MPI buffers can be allocated with cudaMalloc, cudaMallocHost, cudaMallocManaged

CUDA-aware MPI can stage managed buffers through system or device memory

Preliminary results on 8xP100:
5.2x scaling using host buffers
6.8x scaling with NVLINK