CATEGORY: **DEVELOPER - PERFORMANCE OPTIMIZATION - D002**

POSTER **P5230**

CONTACT NAME
Da Li: da.li@mail.missouri.edu

GPU TECHNOLOGY CONFERENCE

Designing Code Variants for Applications with Nested Parallelism on GPUs

Da Li, Michela Becchi

University of Missouri - Columbia

Abstract

The effective parallelization of applications exhibiting irregular nested parallelism is still an open problem. In particular, a naïve mapping of irregular codes to the GPU hardware may lead to resource underutilization and, thereby, limited performance. In this work, we focus on two computational patterns exhibiting nested parallelism: irregular nested loops and recursive algorithms operating on tree and graph data structures. We propose different parallelization templates aimed at increasing the GPU utilization of these codes. Specifically, we investigate different mechanisms to effectively distribute irregular work to streaming multiprocessors and GPU cores. We target the Fermi and the Kepler architecture; in the latter case, we also study parallelization templates relying on dynamic parallelism and propose mechanisms to maximize the work performed by nested kernels and minimize the overhead due to their launch. Our results show that our parallelization templates can achieve 2~6x speedup as compared to baseline code variants based on simple parallelization templates.

Recursive Algorithms

Example Application: Tree descendants

• **flat parallelism**

```
flat_kernel (graph g) {
  thread-mapped-loop(node n ∈ g.nodes) {
    for (node p = g.parent[n]; p≠λ; p=g.parent[p])
      atomic{g.descendants[p]+=1;}
}}
```

• **recursive parallelism – naïve approach**

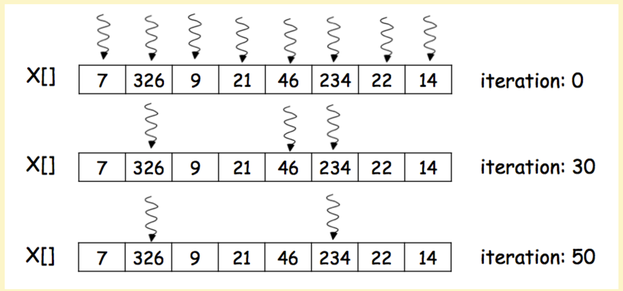
```
naive_rec_kernel (graph g, node n) {
  thread-mapped-loop(node c ∈ g.children(n)){
    if (!leaf(c)) naive_rec_kernel<1,block_SIZE>(g,c);
      atomic{g.descendants[n]+=g.descendants[c];
}}
```

• **recursive parallelism – naïve approach**

```
hier_rec_kernel (graph g, node n) {
  block-mapped-loop (node c ∈ g.children(n)) {
    bool recurse_SHMEM=false;
    if (!leaf(c)) {
      thread-mapped-loop(node gc ∈ g.children(c)){
        if (!leaf(gc)) recurse_SM=true; }
      if (recurse_SHMEM)
        hier_rec_kernel<grid_SIZE,block_SIZE>(g,c);
      else
        g.descendants[c]+=g.num_children(c); }
    atomic{g.descendants[n]+=g.descendants[c];} }}
```

Irregular Nested Loops

Irregular nested loop sizes lead to hardware underutilization




```
set_low = ( i :: x[i] <= TH)
set_high = ( i :: x[i] >= TH)
thread-mapped_kernel(set_low)
block-mapped_kernel(set_high)
```

dual-queue

```
thread-mapped-loop(i) {
  if (x[i]<=TH)
     for ( j=1 to x[i] ) computation(i,j)
  else buffer.add(i)
} blk-mapped-exec(buffer)
```

delayed-buffer

```
thread-mapped-loop(i) {
  if (x[i]<=TH)
     for ( j=1 to x[i] )
computation(i,j)
  else
     blk-mapped_nested-kernel(i)
}
```

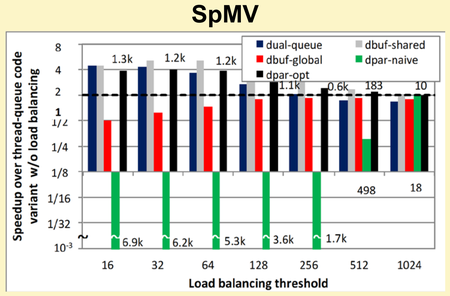
dynamic parallelism - naive

```
thread-mapped-loop(i) {
  if (x[i]<=TH)
     for ( j=1 to x[i] ) computation(i,j)
  else
     buffer.add(i)
} blk-mapped_nested-kernel(buffer)
```

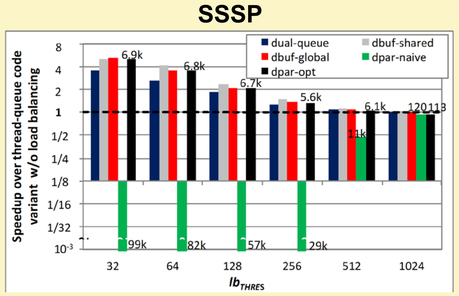
dynamic parallelism - optimized

Experiments

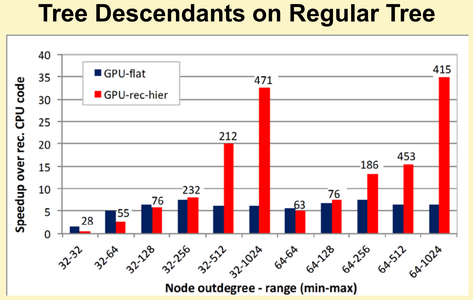
SpMV

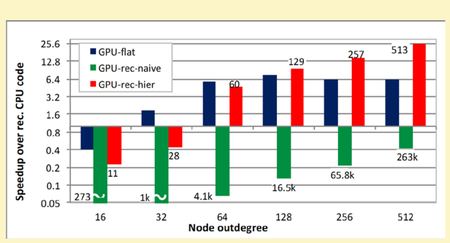


SSSP



Tree Descendants on Regular Tree





Tree Descendants on Irregular Tree

- GPU platform: NVIDIA K20 GPU, 13 x 192 CUDA cores, 4,800 Mbytes global memory
- CPU platform: Intel Xeon E5620, 15MB L1 Cache
- CUDA kernel configuration of our implementation: 192 threads per block

References

[1] P. Harish, P. J. Narayanan, “Accelerating Large Graph Algorithms on the GPU Using CUDA”, High Performance Computing (HiPC), 2007.
[2] Duane Merrill, Michael Garland, Andrew Grimshaw, “High Performance and Scalable GPU Graph Traversal”, Technical Report UVA CS-2011-05, 2011.
[3] Da Li, Michela Becchi, “Deloying Graph Algorithms on GPUs: an Adaptive Solution”, Parallel & Distributed Processing Symposium (IPDPS), 2013.
[4] Mikhail Chernoskutov, “Method of Workload Balancing in GPU Implementation of Breadth-First Search”, High Performance Computing & Simulation (HPCS), 2014.
[5] Da Li, Michela Becchi, “GRapid: a Compilation and Runtime Framework for Rapid Prototyping of Graph Applications on Many-core Processors”, Parallel and Distributed Systems (ICPADS), 2014.

Contact us: da.li@mail.missouri.edu