Optimizing Heterogeneous Platforms for Unstructured Parallelism

Pis: Sudhakar Yalamanchili
Hyesoon Kim
Richard Vuduc
Graph Algorithms

- Web Search
- Networks
- Data Analytics
- Chip Design
- Network Security
- Bioinformatics
Coordinated algorithm, compiler and architecture design efforts to support **unstructured parallelism applications** on Heterogeneous platforms

1) Increase dynamic parallelism
2) Reduce memory divergence
3) Scalable graph algorithms
GPUs are effective for structured applications

However, for unstructured applications
- Poor workload balance -> control flow divergence
- Un-coalesced memory access -> memory divergence
- Low Compute Utilization

Wang et al. IISWC 2014
Dynamically Formed Parallelism (DFP)

- Pockets of Structured Parallelism in irregular applications
  - Locally uniform control flow
  - Locally coalesced memory access

- E.g. Graph Traversal
CDP: launch a kernel from device side, on Kepler GK110 GPU
Solution for DFP
- Launch a child kernel for detected DFP
- Launch only when sufficient parallelism

Parent Kernel

Child Kernels

Launched by t0
Launched by t2
Launched by t3
Launched by t4

Uniform control flow
More memory coalescing

Reduced workload imbalance

Wang et al. ISCA 2015
Dynamic Thread Block Launch

Kernel Distributor Entries
- PC
- Dim
- Param
- ExeBL
  - NAGEI
  - LAGEI

SMX Scheduler
- Control Registers
  - KDEI
  - AGEI
  - NextBL

Aggregated Group Table
- AggDim
- Param
- Next
- ExeBL

DTBL Scheduling

Microarchitecture Extension

Kernel Management Unit
- FCFS Controller
- Kernel Distributor

Aggregated Group Information
- PC
- Dim
- Param
- Next
- DRAM

SMX
- Thread Block Control Registers
  - KDEI
  - AGEI
  - BLKID

Wang et al. ISCA 2015
Memory Problem

- Memory divergence problem
- Memory latency problem

- Prefetch a **data-dependent** memory access pattern **found commonly** in graph algorithms
- Use **spare registers** to make prefetches more **effective**
SRAP – Inserting Prefetches

- Typical graph traverse patterns in GPUs
  - Loop with identified access pattern
    - Two loads into r2 and r3, second load dependent on first load
    - First load has a stride of 4
## GraphBig Benchmark

<table>
<thead>
<tr>
<th>Category</th>
<th>Workload</th>
<th>Computation Type</th>
<th>CPU</th>
<th>GPU</th>
<th>Use Case Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph traversal</td>
<td>BFS</td>
<td>CompStruct</td>
<td>✓</td>
<td>✓</td>
<td>Recommendation for Commerce</td>
</tr>
<tr>
<td></td>
<td>DFS</td>
<td>CompStruct</td>
<td>✓</td>
<td>✓</td>
<td>Visualization for Exploration</td>
</tr>
<tr>
<td>Graph update</td>
<td>Graph construction (GCons)</td>
<td>CompDyn</td>
<td>✓</td>
<td></td>
<td>Graph Analysis for Image Processing</td>
</tr>
<tr>
<td></td>
<td>Graph update (GUp)</td>
<td>CompDyn</td>
<td>✓</td>
<td></td>
<td>Fraud Detection for Bank</td>
</tr>
<tr>
<td></td>
<td>Topology morphing (TMorph)</td>
<td>CompDyn</td>
<td>✓</td>
<td></td>
<td>Anomaly Detection at Multiple Scales</td>
</tr>
<tr>
<td>Graph analytics</td>
<td>Shortest path (SPath)</td>
<td>CompStruct</td>
<td>✓</td>
<td>✓</td>
<td>Smart Navigation</td>
</tr>
<tr>
<td></td>
<td>K-core decomposition (kCore)</td>
<td>CompStruct</td>
<td>✓</td>
<td>✓</td>
<td>Large Cloud Monitoring</td>
</tr>
<tr>
<td></td>
<td>Connected component (CComp)</td>
<td>CompStruct</td>
<td>✓</td>
<td></td>
<td>Social Media Monitoring</td>
</tr>
<tr>
<td></td>
<td>Graph coloring (GColor)</td>
<td>CompStruct</td>
<td>✓</td>
<td></td>
<td>Graph matching for genomic medicine</td>
</tr>
<tr>
<td></td>
<td>Triangle count (TC)</td>
<td>CompProp</td>
<td>✓</td>
<td>✓</td>
<td>Data Curation for Enterprise</td>
</tr>
<tr>
<td></td>
<td>Gibbs inference (GI)</td>
<td>CompProp</td>
<td>✓</td>
<td></td>
<td>Detecting Cyber Attacks</td>
</tr>
<tr>
<td>Social analysis</td>
<td>Degree centrality (DCentr)</td>
<td>CompStruct</td>
<td>✓</td>
<td>✓</td>
<td>Social Media Monitoring</td>
</tr>
<tr>
<td></td>
<td>Betweenness centrality (BCentr)</td>
<td>CompStruct</td>
<td>✓</td>
<td>✓</td>
<td>Social Network Analysis in Enterprise</td>
</tr>
</tbody>
</table>

### Data Set

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Type</th>
<th>Vertex#</th>
<th>Edge#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Twitter Graph</td>
<td>Type 1</td>
<td>120M</td>
<td>1.9B</td>
</tr>
<tr>
<td>IBM Knowledge Repo</td>
<td>Type 2</td>
<td>154K</td>
<td>1.72M</td>
</tr>
<tr>
<td>IBM Watson Gene Graph</td>
<td>Type 3</td>
<td>2M</td>
<td>12.2M</td>
</tr>
<tr>
<td>CA Road Network</td>
<td>Type 4</td>
<td>1.9M</td>
<td>2.8M</td>
</tr>
<tr>
<td>LDBC Graph</td>
<td>Synthetic</td>
<td>Any</td>
<td>Any</td>
</tr>
</tbody>
</table>

IBM System-G based benchmarks

[https://github.com/graphbig/graphBIG/wiki](https://github.com/graphbig/graphBIG/wiki)

Collaboration with IBM
Memory and Control Divergence in GraphBig

![Graph showing the relationship between Branch Divergence and Memory Divergence.](image-url)

- Branch Divergence is plotted on the y-axis.
- Memory Divergence is plotted on the x-axis.
- The graph shows a scattered distribution of data points.

**Axes:**
- **Y-axis:** Branch Divergence
- **X-axis:** Memory Divergence

**Legend:**
- Red diamonds represent the data points.
On-going work

Algorithm-level work

- Staleness aware graph computing
  - Reduce synchronization → Increase parallelism
- BFS 1.5 streaming partitioning algorithms
Project Outcomes


- Nagesh B Lakshminarayana, Hyesoon Kim, “Spare Register Aware Graph Algorithms on GPUs”, HPCA Feb 2014.


- GraphBig Benchmark [https://github.com/graphbig/graphBIG/wiki](https://github.com/graphbig/graphBIG/wiki) (Submitted, April 2015)