If you were plowing a field, which would you rather use?

Two oxen, or 1024 chickens?

(Attributed to S. Cray)

Abdullah Gharaibeh, Lauro Costa, Elizeu Santos-Neto and Matei Ripeanu
Netsyslab
The University of British Columbia
The ‘Field’ to Plow: Graph processing

1B users
150B friendships

PageRank
1.4B pages, 6.6B links

100B neurons
700T connections
Graph Processing Challenges

CPUs

- Poor locality
- Data-dependent memory access patterns
- Low compute-to-memory access ratio
- Large memory footprint (>256GB)
- Varying degrees of parallelism (both intra- and inter-stage)

Caches + summary data structures
The GPU Opportunity

**CPUs**
- Caches + summary data structures
- Large memory footprint >256GB
- Poor locality
- Data-dependent memory access patterns

**GPUs**
- Caches + summary data structures
- Massive hardware multithreading
- 6GB!

Varying degrees of parallelism (both intra- and inter-stage)

**Assemble a hybrid platform**
Motivating Question

Can we efficiently use hybrid systems for large-scale graph processing?

YES WE CAN! 2x speedup (4 billion edges)
Methodology

Performance Model
- Predicts speedup
- Intuitive

Totem
- A graph processing engine for hybrid systems
- Applies algorithm-agnostic optimizations

Partitioning Strategies
- Can one do better than random?

Evaluation
- Hybrid vs. Symmetric
- Partitioning strategies

A Yoke of Oxen and a Thousand Chickens for Heavy Lifting Graph Processing, PACT 2012

On Graphs, GPUs, and Blind Dating: A Workload to Processor Matchmaking Quest, IPDPS 2013
Totem: Compute Model

Bulk Synchronous Parallel

- Rounds of computation and communication phases
- Updates to remote vertices are delivered in the next round
- Partitions vote to terminate execution
Totem: Programming Model

- A user-defined kernel runs simultaneously on each partition

```c
const totem_config_t config = {
    graph,
    partitioning_strategy,
    algo_compute_func,
    msg_combine_func,
    ...
};

totem_config(&config);
totem_execute();
```
Totem: Programming Model

- A user-defined kernel runs simultaneously on each partition
  - Vertices are processed in parallel within each partition
- Messages to remote vertices are combined if a combiner function is defined

```c
#totem_config_t config = {
  graph,
  partitioning_strategy,
  algo_compute_func,
  msg_combine_func,
  ...
};
totem_config(&config);
totem_execute();
```
<table>
<thead>
<tr>
<th></th>
<th>Intel Nehalem Xeon X5650 (2x sockets)</th>
<th>Fermi GPU Tesla C2075 (2x GPUs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core Frequency</td>
<td>2.67GHz</td>
<td>1.15GHz</td>
</tr>
<tr>
<td>Num HW-threads</td>
<td>12</td>
<td>448</td>
</tr>
<tr>
<td>Last Level Cache</td>
<td>12MB</td>
<td>2MB</td>
</tr>
<tr>
<td>Main Memory</td>
<td>144GB</td>
<td>6GB</td>
</tr>
<tr>
<td>Memory Bandwidth</td>
<td>32GB/sec</td>
<td>144GB/sec</td>
</tr>
</tbody>
</table>
Use Case: Breadth-first Search

- Graph Traversal algorithm
- Very little computation per vertex
- *Sensitive to cache utilization*

Visited bit-vector

```
1 0 1 0 1 0 0 0 1 0
```
BSF Performance

Workload: Graph500 benchmark (|V| = 256m, |E| = 4B)
1S1G = 1 CPU Socket + 1 GPU

Linear improvement with respect to offloaded part

Fills GPU memory
BSF Performance

Workload: Graph500 benchmark (|V|=256m, |E|=4B)
1S1G = 1 CPU Socket + 1 GPU

Can we do better than random partitioning?
BSF Performance

Workload: Graph500 benchmark (|V|=256m, |E|=4B)
1S1G = 1 CPU Socket + 1 GPU

Computation phase dominates run time
Workload-Processor Matchmaking

Observation:
Real-world graphs have **power-law** edge degree distribution

Idea:
Place the *many low-degree* vertices on **GPU**

Place the *few high-degree* vertices on **CPU**

Log-log plot of edge degree distribution for LiveJournal social network
BFS Performance

Workload: Graph500 benchmark ($|V|=256m$, $|E|=4B$)
1S1G = 1 CPU Socket + 1 GPU

Specialization $\Rightarrow$ Superlinear speedup (~3x)
~1.8x performance gain!
BFS Performance

Offloading the many low-degree nodes improves CPU cache performance

Workload: Graph500 benchmark (|V|=256m, |E|=4B)
1S1G = 1 CPU Socket + 1 GPU
BFS Scalability

- Hybrid offers ~2x speedup vs Symmetric
- GPU memory is limited, GPU share < 1% of the edges!
- Comparable to top 100 in Graph500 challenge!
More Compute Intensive Use Case

PageRank

Hybrid offers better gains for more compute intensive algorithms
Summary

• **Problem**: large-scale graph processing is challenging
  – **Heterogeneous workload** – power-law degree distribution

• **Solution**: **Totem Framework**
  – **Harnesses hybrid systems** – Utilizes the strengths of CPUs and GPUs
  – **Partitioning** – workload to processor matchmaking
  – **Powerful abstraction** – simplifies implementing algorithms for hybrid systems
  – **Algorithms** – BFS, SSSP, PageRank, Centrality Measures
Totem

Open Source Package
code@: netsyslab.ece.ubc.ca