Computing Betweenness Centrality for Small World Networks on a GPU

Pushkar R. Pande
Georgia Institute of Technology

David A. Bader
Georgia Institute of Technology

Introduction

- **Betweenness Centrality of a vertex v is defined as,**

$$BC(v) = \sum_{S \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

σ_{st} **: Number of shortest paths between vertices s and t**

$\sigma_{st}(v)$ **: Number of shortest paths between vertices s and t through v**

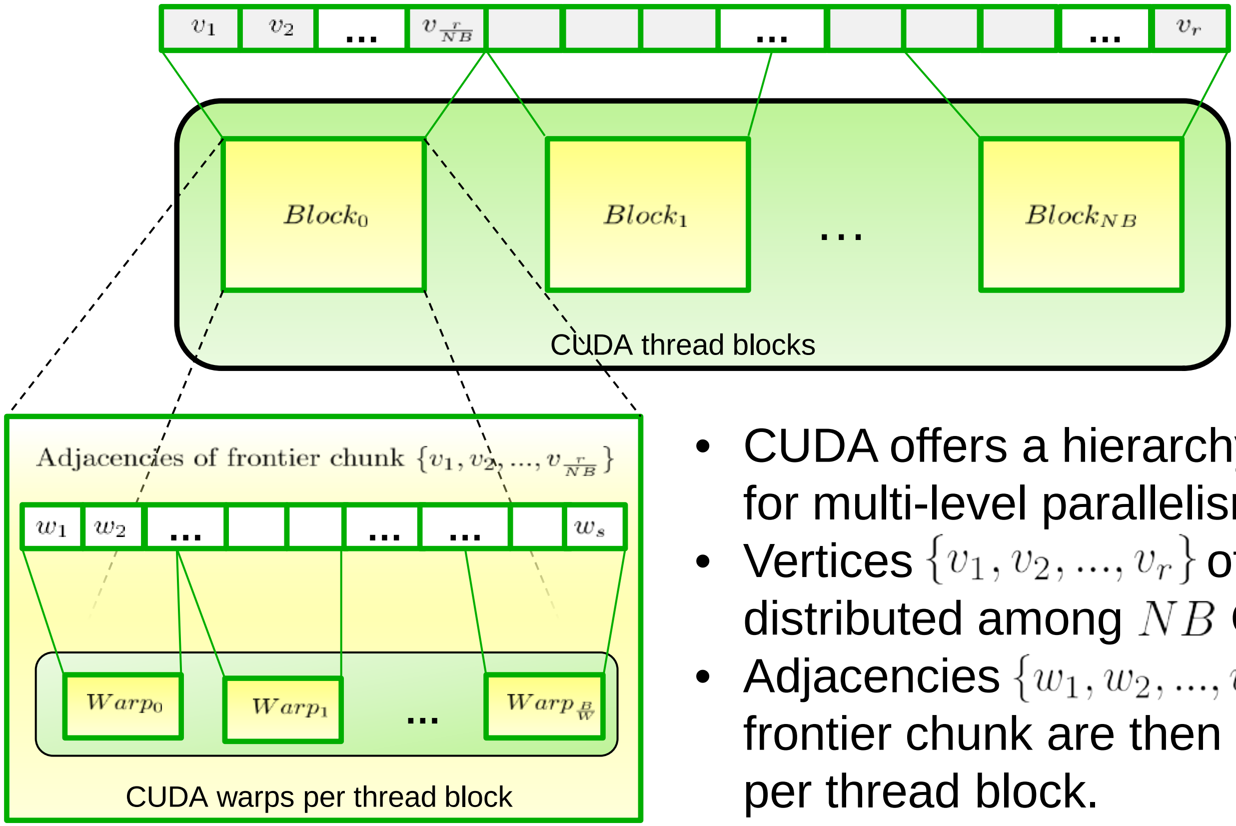
- **For a graph with 1 million vertices and 10 million edges, betweenness computation on GPU is accelerated by ~20$\times$ compared to single thread CPU performance.**
- **Our algorithm for computing betweenness is based on the sequential algorithm [Brandes 2001],**

  For each source vertex, perform:

  1. **Breadth-First traversal for enumerating shortest paths.**
  2. **Accumulate dependencies by back propagation.**

- **Bader and Madduri (ICPP 2006) gave the first parallel implementation. This targeted the highly multithreaded supercomputer, the Cray XMT.**

Our GPU Implementation using CUDA



NB – Number of thread blocks

B – Block size

W – Warp size

- CUDA offers a hierarchy of threads that allows for multi-level parallelism
- Vertices $\{v_1, v_2, ..., v_r\}$ of i_{th} frontier are distributed among NB CUDA thread blocks.
- Adjacencies $\{w_1, w_2, ..., w_s\}$ of the corresponding frontier chunk are then distributed to $\frac{B}{W}$ warps per thread block.

Performance and Optimizations

- **Performance is improved by,**
  - **Using fast shared memory for buffering writes to global memory.**
    - **Coalesces global memory access**
    - **Reduces number of atomic operations on global data structures.**
  - **Frontier pre-processing for balanced load.**
  - **Accumulating dependencies by assigning a warp per vertex.**
  - **Improved work assignment per warp using virtualized warp size.**
- **GPU performance for betweenness compared to CPU with number of threads $nt = \{1, 16\}$ for a subset of 500 source vertices,**

Graph	#Vertices	#Edges	CPU Time (s)		GPU Time (s)		Speed up on Fermi	
			$nt = 1$	$nt = 16$	Tesla	Fermi	$nt = 1$	$nt = 16$
syn1.gr	262,144	2,097,152	24.19	5.47	5.65	2.64	9.16	2.07
syn2.gr	524,288	4,194,304	80.72	15.78	13.48	6.33	12.75	2.49
syn3.gr	1,048,576	8,388,608	184.11	32.68	23.31	11.31	16.28	2.89
syn4.gr	1,000,000	10,000,000	93.63	17.35	13.77	6.11	15.32	2.84
syn5.gr	1,000,000	10,000,000	232.79	33.87	19.98	11.83	19.68	2.86

GPU: NVIDIA C1060 (Tesla), 1.3 GHz, 30×8 cores
NVIDIA M2070 (Fermi), 1.15 GHz, 14×32 cores
CPU: 2 quad-core Intel Xeon E5530, 2.4 GHz, hyper-threading enabled