Computer Science, Texas A&M University

Towards Faster External-Memory Graph Computing on Small Neighborhoods

Di Xiao, Yi Cui, and Dmitri Loguinov

Internet Research Lab (IRL)
Department of Computer Science and Engineering
Texas A&M University, College Station, TX, USA 77843

Agenda

- Introduction
- Taxonomy
- Decomposable Wedge Computing (DWC)
- Experiments

Introduction

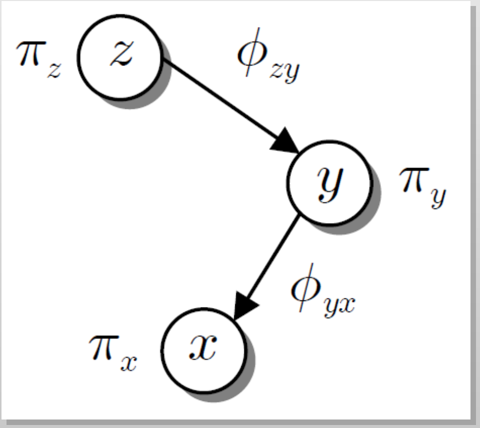
- Our focus is on applications that operate on *large-scale* graphs
  - Datasets that do not fit in RAM and require external-memory (disk-based) algorithms
  - Used in such fields as information retrieval, data analytics, social networks, databases, recommendation systems
- Assume $G = (V, E)$ is a graph stored on disk
  - Consists of n nodes and m edges
  - Depending on the application, the graph may be weighted

Introduction

- Computation can often be abstracted as applying some function f to node/edge weights within close vicinity of each node
- Classical problems use only 1st neighborhoods
  - Information flows between directly-connected nodes
  - Can be reduced to scans over adjacency lists in G
- Examples:
  - Graph inversion and PageRank
- This can be solved efficiently by MapReduce
  - Each iteration has linear cost in the number of edges m

Introduction

- More complex computation uses 2nd neighborhoods
  - Define a wedge P_{xyz} to be a simple path of length 2 that ends at x, including all relevant weights
  - Let wedge neighborhood P_x be a set of all wedges at x
  - Then, wedge computing is a process that evaluates $f(P_x)$ for all $x \in V$



- Examples
  - Supporter-based ranking: count unique nodes at distance 2 from each node along in-edges
  - Motif discovery: enumerate all triangles (3-cycles) or quandrangles (4-cycles)

Introduction

- Similarity ranking, sparse matrix-matrix multiplication, etc.

- Main caveat is that P_x is not available while scanning G
  - Worse yet, P_x is orders of magnitude larger than degree of x
  - IRLbot domain out-graph: 1.8B edges and 3.1T wedges
- Classical solutions (MapReduce, graph libraries, database joins) use streaming/sequential I/O
  - In the worst case, require sorting all T_n wedges on disk, where T_n can be as high as m^2
  - Supporter count on 7-GB IRLbot graph: prior work executes between 31 and 328 TB of I/O

Introduction

- Alternatively, sets P_x can be built using random access
  - This requires m seeks and loading of $m+T_n$ nodes from disk
  - For the IRLbot domain graph: 1.8B seeks, 12 TB of I/O
  - With spinning disks, this results in ~6 years spent in seeking
  - With SSDs, the seek cost is much smaller, but 12 TB of I/O is still relatively expensive
- Our goals
  - Examine how to solve wedge-computing problems with better asymptotic I/O cost than the existing methods
  - Do not assume that seeks are fast and keep the solution general, i.e., applicable to non-SSD drives

Agenda

- Introduction
- Taxonomy
- Decomposable Wedge Computing (DWC)
- Experiments

Taxonomy

- We partition wedge computing into 3 groups
  - Category I: admits 2D graph decomposition
  - Category II: allows 1D graph decomposition
  - Category III: no decomposition
- Category I (triangles)
  - Has efficient solutions already
- Category II (4-cycles, similarity, matrix multiplication)
  - Open problem that we tackle here
- Category III (4-cliques)
  - Left for future work

Agenda

- Introduction
- Taxonomy
- Decomposable Wedge Computing (DWC)
- Experiments

DWC

- We create two 1D partitioning schemes that allow I/O-efficient processing of wedges
  - DWC_1 uses a baseline approach
  - DWC_2 is a more sophisticated method
- The paper provides details and models I/O
  - Assume M is RAM size
  - DWC_1 has I/O cost proportional to m^2/M
  - DWC_2 is linear if the expected product of in/out degree stays bounded as $n \to \infty$
- Two flavors of each method
  - DWC_1-A / DWC_2-A partition on x
  - DWC_1-B / DWC_2-B do the same on z

Agenda

- Introduction
- Taxonomy
- Decomposable Wedge Computing (DWC)
- Experiments

Experiments

- Run supporter ranking algorithm
  - Count *unique* nodes at distance 2, excluding direct neighbors
  - 8-core Intel Skylake-X CPU with M = 8 GB of RAM
  - 160TB RAID-50 system with 24 magnetic drives

Name	Graph	Nodes n	Edges m	Degree	Size (GB)	Wedges $T_n = \sum_i X_i Y_i$
$\mathcal{D}_1$	ClueWeb-domain	30,558,375	415,167,456	13.6	1.7	4,240,567,641,185
$\mathcal{D}_2$	ClueWeb-host	110,675,107	1,064,508,293	9.6	4.5	1,404,873,157,927
$\mathcal{D}_3$	ClueWeb-page	2,570,747,470	49,902,497,310	19.4	199.0	2,889,895,321,002
$\mathcal{D}_4$	IRLbot-domain	86,534,418	1,799,516,827	20.8	7.1	3,073,393,262,407
$\mathcal{D}_5$	IRLbot-host	641,982,060	6,752,615,553	10.5	27.9	2,704,210,948,405
$\mathcal{D}_6$	IRLbot-page	4,051,690,819	238,194,440,791	58.8	916.2	79,864,490,755,128

114x RAM

- Six web graphs
  - From 1.7 GB (415M edges) to 916 GB (238B edges)
  - Wedge count varies from 1.4T to 80T
  - Main case of interest is $\mathcal{D}_6$

Experiments

Shaded cells = unable to finish in 3 weeks (result extrapolated)

- I/O in TB:

Method	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_5$	$\mathcal{D}_6$
Hadoop	564	155	338	328	315	9,399
STXXL	237	75	159	170	149	4,974
GraphChi	47	17	29	31	27	808
Rstream	62	21	44	45	41	1,165
DWC_2-A	0.002	0.004	2.5	0.007	0.06	28
DWC_2-B	0.002	0.004	1.9	0.007	0.08	18

  - Hadoop fails on all 6 datasets (155TB to 9.4PB)
  - STXXL finishes the sparsest graph $\mathcal{D}_2$ using 75TB, but does not complete any of the other ones
  - GraphChi/Rstream are successful in $\mathcal{D}_1$–$\mathcal{D}_5$ with 17-62TB, but stall on $\mathcal{D}_6$ (808TB – 1.1PB)
- DWC_2-B works well in all cases
  - For graphs that fit in RAM, it uses > 4000x less I/O
  - On those that do not ($\mathcal{D}_3$, $\mathcal{D}_5$, $\mathcal{D}_6$), it needs 15-337x less I/O

Experiments

- Runtime (days):

Method	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_5$	$\mathcal{D}_6$
Hadoop	115	34	77	83	72	2,437
STXXL	41.4	11.6	26.7	28.7	24.8	1,093
GraphChi	16.6	4.8	9.4	9.9	8.8	259
Rstream	16.5	4.9	10.2	10.8	9.5	281
DWC_2-A	0.20	0.17	0.24	0.28	0.20	2.2
DWC_2-B	0.08	0.07	0.15	0.11	0.09	1.3

  - Hadoop is projected to take between a month and 6.7 years
  - STXXL 11 days to 3 years
  - GraphChi/Rstream between 5 days and 9 months

- DWC_2-B compared to the best prior work
  - For graphs that fit in RAM ($\mathcal{D}_1$, $\mathcal{D}_2$, $\mathcal{D}_4$): 68-206x faster
  - For medium scale datasets ($\mathcal{D}_3$, $\mathcal{D}_5$): 62-97x faster
  - For $\mathcal{D}_6$, 199x faster