J. Parallel Distrib. Comput. 76 (2015) 81–93

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Work efficient parallel algorithms for large graph exploration on emerging heterogeneous architectures☆

Dip Sankar Banerjee *, Ashutosh Kumar, Meher Chaitanya, Shashank Sharma, Kishore Kothapalli

International Institute of Information Technology, Hyderabad, Gachibowli, Hyderabad, 500 032, India

HIGHLIGHTS

- Processing real world graphs in an efficient manner through input pruning.
- Two different pruning strategies based on 1-degree nodes and articulation points.
- Improvements upto 35% or 1.57x over current best known results.
- Experimental evaluation of algorithms proposed on several real world graphs.
- Heterogeneous multicore implementation provides better performance efficiency.

ARTICLE INFO

Article history:
Received 25 March 2014
Received in revised form
17 November 2014
Accepted 25 November 2014
Available online 26 December 2014

Keywords:
Graph algorithms
Irregular computations
Heterogeneous computing
Input pruning

ABSTRACT

Graph algorithms play a prominent role in several fields of sciences and engineering. Notable among them are graph traversal, finding the connected components of a graph, and computing shortest paths. There are several efficient implementations of the above problems on a variety of modern multiprocessor architectures.

It can be noticed in recent times that the size of the graphs that correspond to real world datasets has been increasing. Parallelism offers only a limited succor to this situation as current parallel architectures have severe short-comings when deployed for most graph algorithms. At the same time, these graphs are also getting very sparse in nature. This calls for particular solution strategies aimed at processing large, sparse graphs on modern parallel architectures.

In this paper, we introduce graph pruning as a technique that aims to reduce the size of the graph. Certain elements of the graph can be pruned depending on the nature of the computation. Once a solution is obtained on the pruned graph, the solution is extended to the entire graph. Towards, this end we investigate pruning based on two strategies that justifies their use in current real world graphs.

We apply the above technique on three fundamental graph algorithms: breadth first search (BFS), Connected Components (CC), and All Pairs Shortest Paths (APSP). For experimentations, we use three different sources for real world graphs. To validate our technique, we implement our algorithms on a heterogeneous platform consisting of a multicore CPU and a GPU. On this platform, we achieve an average of 35% improvement compared to state-of-the-art solutions. Such an improvement has the potential to speed up other applications reliant on these algorithms.

1. Introduction

Graph algorithms find a large number of applications in engineering and scientific domains. Prominent examples include solving problems arising in VLSI layouts, phylogeny reconstructions, data mining, image processing, and the like. Some of the most commonly used graph algorithms are graph exploration algorithms such as Breadth First Search (BFS), computing components, and finding shortest paths. As the current real life problems often

☆ A part of this work has appeared previously in Proceedings of the 20th IEEE International Conference on High Performance Computing (HiPC), 2013.

* Corresponding author.
E-mail addresses: dipsankar.banerjee@reasearch.iiit.ac.in (D.S. Banerjee), ashutosh.kumar@students.iiit.ac.in (A. Kumar), meher.c@reasearch.iiit.ac.in (M. Chaitanya), shashank.sharma@students.iiit.ac.in (S. Sharma), kkishore@iiit.ac.in (K. Kothapalli).

http://dx.doi.org/10.1016/j.jpdc.2014.11.006
0743-7315/

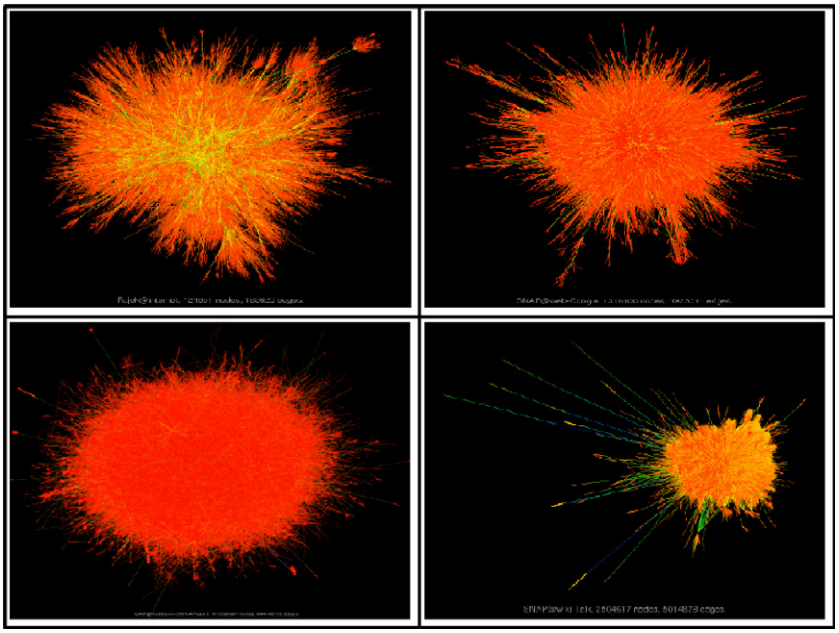

Fig. 1. A sample of four real world graphs from [45]. On the top-left corner is the graph internet, top-right is the graph web-Google, bottom left is the graph webbase_1M, and the bottom-right is the graph wiki-Talk.

Table 1
The SNAP [43] graphs used for experimentations and their properties. The column heading r in the last column indicates the number of iterations required to remove all pendant vertices.

SNAP Graphs					
Graph	Description	Nodes	Edges	Pendant vertices	**r**
amazon0601	Amazon product co-purchasing network [28]	403,394	3,200,490	38,121 (**9.45**%)	3
email-Enron	Email communication network from Enron [26]	36,692	367,662	2,069 (**5.64**%)	2
ca-Condmat	Collaboration n/w of Condensed Matter [29]	23,133	186,936	3,338 (**14.43**%)	2
Roadnet-TX	Road network of Texas [32]	1,393,383	3,843,320	170,271 (**12.22**%)	4
Web-Stanford	Web graph of Stanford.edu [32]	281,903	2,312,497	21,819 (**7.74**%)	3
Web-Berkstan	Web graph of Berkeley and Stanford [32]	685,230	7,600,595	60,506 (**8.83**%)	3
Web-Notredam	Web graph of Notre Dam [32]	325,729	1,497,134	33,322 (**10.23**%)	2
p2p-Gnutella	Gnutella peer to peer network [27]	62,586	147,892	9,738 (**15.56**%)	2
LiveJ	Links in Live Journal[4]	4,848,571	68,993,773	403,401 (**8.3**%)	4
Flickr	Connection among Flickr users [35]	2,302,925	33,140,018	488,450 (**21.2**%)	3
Baidu	Links in Baidu Chinese online encyclopedia [37]	2,141,300	17,794,839	266,592 (**12.4**%)	5
Wiki	Links in English wikipedia [3]	15,172,740	131,166,252	1,195,612 (**7.8**%)	6
Orkut	Connection of Orkut users [50]	3,072,627	11,718,583	464,274 (**15.1**%)	4
Patents	Citations among US patents [30]	3,774,768	16,518,948	691,160 (**18.3**%)	2
Roadnet-CA	Road network of California [31]	1,965,206	5,533,214	228,357 (**11.6**%)	3

involve the analysis of massive graphs, it is often seen that parallel solutions provide an acceptable recourse.

Parallel computing on graphs however is often very challenging because of their irregular nature of memory accesses. This irregular nature of memory access stresses the I/O systems of most modern parallel architectures. It is therefore not surprising that most of the recent progress in scalable parallel graph algorithms is aimed at addressing these challenges via innovative use of data structures, memory layouts, and SIMD optimizations [36,20,39]. Recent results have been able to make efficient use of modern parallel architectures such as the Cell BE [39], GPUs [36,21,20], Intel multi-core architectures [12,49,1] and the like. Algorithms running of GPUs have shown standout performance amongst these because of its massive parallelism.

Another recent development in parallel computing is the design and engineering of heterogeneous algorithms that are aimed at heterogeneous computing platforms. Heterogeneous computing platforms consist of tightly coupled heterogeneous devices including CPUs and accelerator(s). One such example is a collection of a CPU coupled with a graphics accelerator (GPU). Heterogeneous algorithms for CPU+GPU based computational platforms have been designed for also graph breadth-first exploration [21,36,18]. All of the above-cited works show an average of 2x improvement over pure GPU algorithms.

Most of the above works in general aim at data structure and memory layout optimizations but largely run classical algorithms on the entire input graph. These algorithms are designed for general graphs whereas the current generation graphs possess markedly distinguishable features such as being large, sparse, and large deviation in the vertex degrees. In Fig. 1, we show some of the real-world graphs taken from [45]. As can be seen from Fig. 1, these graphs have several vertices of very low degree, often as low as 1. For instance, in the case of the graph web-Google, 14% of the vertices have degree 1. Table 1 lists other properties of a few real world graphs from [45].

Current parallel algorithms and their implementations [36,18,39,21,47] do not take advantage of the above properties. For instance, in a typical implementation of the breadth-first search algorithm, one uses a queue to store the vertices that have to be explored next. But, a vertex v of degree 1 that is in the queue will not lead to the discovery of any yet undiscovered vertices. So, the