GPU implementation of epidemiological behaviour in large social networks

Sairam K M Menon*§, P K Baruah§, Matija Sosic‡
§Sri Sathya Sai Institute of Higher Learning, Prashanthi Nilayam, India.
‡Faculty of Electrical Engineering and Computing, Zagreb, Croatia.
Email: sairamkmmenon@gmail.com, pkbaruah@sssihl.edu.in, matija.sosic@fer.hr

***Abstract*—In a social network, epidemic spread could be a spread of an infection, opinions, trends, fads, diseases or worm propagation in network. Epidemic spread computation on such huge and ever growing social networks is incredibly challenging. High-performance computing using GPUs has become an important tool to solve computationally intensive problems. This paper presents a GPU based implementation(GPU_OPT) of Susceptible-Infected-Recovered (SIR) model. GPU_OPT performs 1.8x-3.9x faster than an existing CUDA SIR implementation across various types of networks studied. CUDA SIR is 10x faster than FastSIR(a single core CPU implementation) in the worst and so GPU_OPT is effectively about 30x faster when compared to FastSIR on an average case. This implementation was tested on social networks of varied types like Condense Matter Physics collaboration network, friendship network, who-trust-whom relationship network and a Email communication network.**

***Index Terms*—Social Networks, SIR, GPU, Epidemiological behaviour, CUDA.**

I. Introduction

A social network is a patterned arrangement in a society where the actions of individuals are inter-related in some way or other. But over time this social structure of a social network applies to various walks of life. A social network can exist with different population levels of its heterogeneous constituents with variant complexity and interaction levels. This makes social network a field that compels heavy interdisciplinary academic interventions at various stages of its study and analysis. The heterogeneous constituents in its dynamic nature, i.e. , when active in the social network could be considered as contagions. Now, the contagions could be opinions, trends, fads, diseases, worm propagation in network and so on to name a few. But it is quite noticeable that the above examples involve a diverse phenomenon on different types of networks. Networks with complicated dependencies leading to high communication cost, synchronization problems and load balancing are issues in the computations cost involved in the simulations. Therefore, modelling a situation of social network involves a lot of computation.

Graphics Processor Unit (GPU) is a high performance computing device that exploits the data level parallelism. GPU was mainly used only for display purposes. It has become attractive for general purpose application because of its cost-effective approach to accelerate data intensive and compute intensive application. The GPU environment is a heterogeneous architecture which consists of CPU and GPU. Compute intensive problems like Epidemic simulations for huge social networks need to use this high computational power of GPUs. Compute Unified Device Architecture (CUDA), developed by Nvidia is an architecture for parallel computation. Developers of software in CUDA find a lot of functions provided by the CUDA libraries which makes programming applications of interest easier. In this work we attempt to effeciently implement the Susceptible Infected Recovered (SIR) in the CUDA platform. In the SIR model, the following represent the three compartments:

- *Susceptible*: Consists of host that probably can get infected.
- *Infected*: Consists of hosts that are infected and

*Student Author

are potential to infect the neighbouring hosts.
- *Recovered*: Consists of hosts that are immune as they have recovered from the infected phase.

To simulate a real life scenario using SIR model, we need to study large social networks. Processing huge networks with millions of nodes and edges would take a lot of time if done sequentially. So we need to reduce the simulation time by employing the GPUs. Some of other widely used epidemic simulations in the study of public health are Epifast [3], EpiSimdemics [4], InterSim [5].

A social network can be represented as a graph where the interactions among the host become edges and the nodes represent the hosts itself. In this study we try to implement the idea proposed in [1] in an efficient manner using CUDA. Here we consider static, non-weighted and undirected social networks. Parameters that decide the course of the simulations with a given starting node are:

- The probability of an infected node going to recovered state (q).
- The probability of a susceptible node getting infected from another infected node (p).

Also, each node during the simulation can be in only one of the three states of the SIR model and so do not consider a case where the state could be fuzzy. The simulation completes when all the infected nodes become either recovered or immune. Some of the contributions of this work are:

- A faster GPU based version of the SIR model.
- A scalable version of the SIR model.
- Study of the epidemiological behaviour on a wide variety of network data sets.

The rest of the paper is as follows: Methods section describes the basic idea of the algorithm [1] that is used in this implementation. Followed by the section Applications and Results which tabulates the timings of the implementation on various publically available social networks [8] . We finally conclude with discussion of some of the possible extensions to this work that we wish to do in future.

II. Methodology

In previously existing algorithm FastSIR algorithm [2], the time to recover is speedy when compared to Naive SIR algorithm [2] due to the employment of probability distribution of the number of infected nodes. It was noted that FastSIR does not follow the epidemic dynamics.

The simulation begins after the graph G is loaded from the input file and after the random generators are initialised. The input file which represents the network is stored as an adjacency list in a single array for ease of access in GPUs. Initially except the starting node all other nodes are set to the susceptible state. The start state that is arbitrarily chosen is in the infected state and the simulation proceeds with the infected nodes trying to infect its neighbouring nodes with a given probability p. Nodes in the infected state tries to recover with given probability q. The simulation finishes when all the infected nodes recover and goes to the recovered state.

In the heart of the algorithm followed is CUDA SIR [1] algorithm which uses CUDA BFS [6] . CUDA BFS caters well for SIR simulations as SIR simulation is a kind of graph traversal but with certain probabilities during traversal. Initially all the nodes are divided into equal sized groups which are assigned to a warp which avoids large number of threads being ideal. Parallelisation is achieved at the infection and the recovery stages of the simulation. In CUDA SIR at end of each step the appropriate functions are called by the host (CPU). Shared memory which is faster than the global memory is employed in the GPU kernel implementations of GPU_OPT. The GPU alternates between the two phases:

- Single Instruction Single Data (SISD): Here all the threads execute the same instruction on the same data in the shared memory.
- Single Instruction Multiple Data (SIMD): each thread in the same warp executes same instruction but on different data.

In SISD phase all the threads in a warp loop sequentially through the assigned nodes. If the cur-

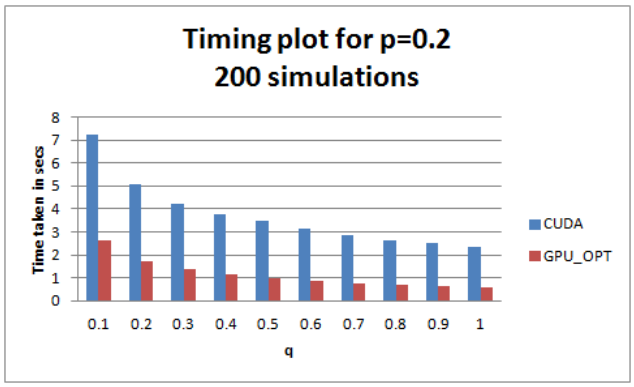

Fig. 1. Timing in secs for 200 runs of Ca-CondMat.txt

TABLE I
TIMINGS FOR 200 SIMULATIONS OF CA-CONDMAT

p	p=0.2		p=0.5		p=0.8	
q	CUDA	GPU_OPT	CUDA	GPU_OPT	CUDA	GPU_OPT
0.1	7.25	2.64	5.24	1.77	4.52	1.39
0.2	5.05	1.73	3.41	1.28	2.69	1.04
0.3	4.21	1.35	2.78	1.03	2.18	0.87
0.4	3.75	1.13	2.42	0.88	1.93	0.76
0.5	3.49	0.97	2.29	0.78	1.68	0.68
0.6	3.13	0.86	2.06	0.71	1.57	0.62
0.7	2.87	0.77	1.96	0.66	1.51	0.58
0.8	2.63	0.70	1.91	0.61	1.42	0.54
0.9	2.52	0.65	1.86	0.57	1.39	0.51
1.0	2.35	0.60	1.74	0.54	1.37	0.49

TABLE II
TIMINGS IN SECS FOR 200 SIMULATIONS OF LOC-GOWALLA

p	p=0.2		p=0.5		p=0.8	
q	CUDA	GPU_OPT	CUDA	GPU_OPT	CUDA	GPU_OPT
0.1	29.44	13.84	19.03	9.89	17.21	7.64
0.2	21.86	9.46	12.80	6.81	9.75	5.55
0.3	18.80	7.48	10.92	5.57	7.85	4.58
0.4	15.58	6.29	9.96	4.85	6.88	4.09
0.5	13.74	5.43	9.12	4.42	6.49	3.76
0.6	12.30	4.82	8.56	4.07	6.11	3.53
0.7	11.14	4.27	7.94	3.79	6.05	3.34
0.8	10.19	3.90	7.78	3.54	5.92	3.20
0.9	9.25	3.58	7.05	3.33	5.40	3.12
1.0	8.76	3.34	6.73	3.15	5.14	3.01

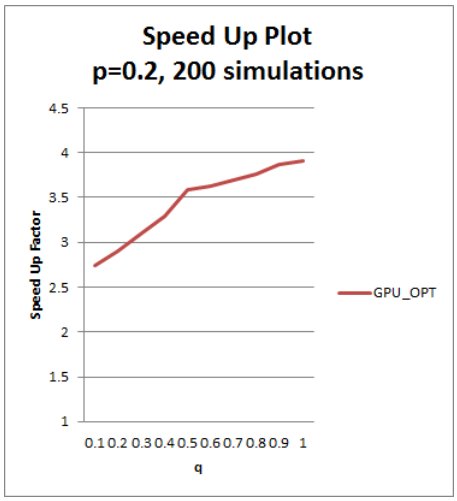

Fig. 2. SpeedUp Plot for Ca-CondMat.txt

rently inspected node is infected, SIMD phase is applied to its neighbours. When p value is much smaller than the q value the infection phase stops even before it infects every node. Whereas, if p is much larger than the q value then it would be a waste of time to check that there are no susceptible neighbours. A specialized function is implemented to take care of the above mentioned situation in GPU_OPT.

III. APPLICATIONS AND RESULTS

The tests were performed on system with a 64-bit Westmere processor Xeon Intel Hexa-Core CPU and Nvidia Tesla M2070 GPU. Matija et al.[1] show that CUDA SIR achieves a speed of 10x in the worst case when compared to CPU SIR(FastSIR) Implementation. This work shows that we achieve a speed up of 1.8x to 3.9x when compared to CUDA SIR. Therefore, when compared to FastSIR, our SIR implementation(GPU_OPT) is about 30x faster in the average case. It was noted that the performance of the SIR model implementations depends on the type of the network under study. The following subsections shows the results of the comparative study of CUDA SIR and GPU_OPT done on various networks.

A. Collaboration Network

Ca-CondMat.txt is a Condense Matter Physics collaboration network that includes scientific collaborations between authors of the papers submitted to this category. In this network if two authors have co-authored then they share an undirected edge between them. It is available in [9]. The timing for the various p and q values are noted in the Table I and in the fig 1 . Also the speed up of the implementation is represented in the fig 2. The timing plot and the speedup plots clearly shows the performance achieved by this implementation(GPU_OPT).

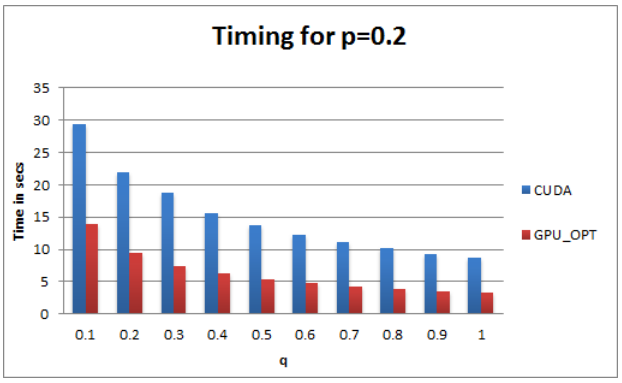

Fig. 3. Timing Plot for loc-gowalla-edges.txt

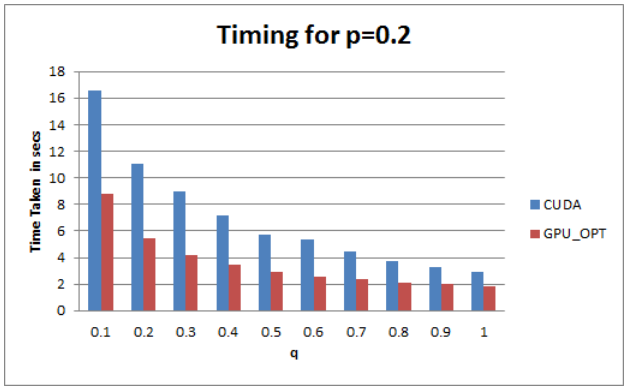

Fig. 4. Timing Plot for soc-Epinions1.txt

TABLE III
TIMINGS IN SECS FOR 200 SIMULATIONS OF SOC-EPINIONS

p	p=0.2		p=0.5		p=0.8	
q	CUDA	GPU_OPT	CUDA	GPU_OPT	CUDA	GPU_OPT
0.1	16.5	8.79	11.87	6.73	10.73	5.48
0.2	11.04	5.47	7.49	4.38	6.22	3.74
0.3	8.95	4.20	6.04	3.42	4.92	3.00
0.4	7.15	3.43	5.17	2.90	4.24	2.63
0.5	5.74	2.94	4.55	2.58	3.73	2.40
0.6	5.36	2.60	4.31	2.34	3.38	2.25
0.7	4.44	2.34	4.10	2.16	3.22	2.14
0.8	3.76	2.14	3.78	2.01	2.98	2.09
0.9	3.30	1.99	3.63	1.90	2.95	2.03
1.0	2.93	1.87	3.16	1.82	2.81	1.99

TABLE IV
TIMINGS IN SECS FOR 200 SIMULATIONS OF EMAIL-ENRON

p	p=0.2		p=0.5		p=0.8	
q	CUDA	GPU_OPT	CUDA	GPU_OPT	CUDA	GPU_OPT
0.1	6.43	5.37	6.28	4.31	6.11	3.42
0.2	3.53	2.42	3.58	2.70	3.56	2.50
0.3	2.25	1.73	2.78	2.07	2.75	2.02
0.4	1.45	1.10	2.03	1.60	2.33	1.79
0.5	1.41	0.88	1.81	1.43	1.93	1.56
0.6	1.07	0.61	1.58	1.19	1.74	1.43
0.7	1.10	0.58	1.43	0.90	1.55	1.35
0.8	0.68	0.41	1.09	0.86	1.41	1.22
0.9	0.71	0.39	0.99	0.75	1.37	1.11
1.0	0.55	0.28	1.03	0.76	1.23	1.13

B. Friendship Network

The file loc-gowalla-edges.txt is a friendship network which is based on location. Here the users share their location by checking-in to the social networking website. Data set is available at [10] . Rumour or fad spread in a social networking website based on location is simulated using the SIR model where the infection spread is the rumour spread. The timings are compared in the Table II and in the fig 3.

C. Who-Trust-Whom Network

The network in soc-Epinions1.txt is a directed graph that represents who-trust-whom relationship in a social networking site of consumer review site named Epinions.com. While running the SIR model on this network we assume that if a user trusts another user then there exists an edge between them. Hence SIR can simulate the spread about a news among the user network. This data set is available at [11]. The timings are compared in the Table III and in the fig 4.

D. Email Network

Data set in email-Enron.txt represent a communication via email within a dataset of half a million of emails made publicly available during an investigation by the Federal Energy Regulatory Commission. In this network an edge exists between two email addresses if there was at least one mail exchanged between them. SIR model simulates the possible way that information can propagate in such a network. Data set available at [12] . The timings are compared in the Table IV and in the fig 5.

IV. CONCLUSION AND FUTURE WORK

In this paper we present a GPU based implementation of SIR model and compare its performance with respect to CUDA SIR. We also checked

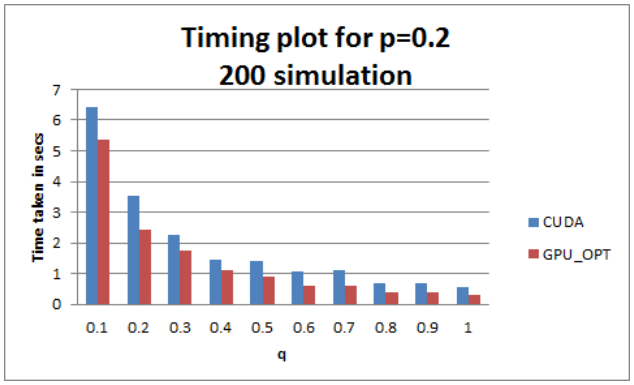

Fig. 5. Timing Plot for email-Enron.txt

the performance on varied network data sets and hence showing the possibility of SIR model application onto various domains. We have shown that GPU_OPT performs 1.8x-3.9x faster than CUDA SIR and effectively about 30x faster when compared to FastSIR on an average case. As a future work, we would like to implement the SIR on multiple GPUs with more efficient parallel random generators and graph traversal techniques. Also, study on dynamic networks with interventions would be an interesting work that simulates the network more realistically but at the cost of higher complexity in computation.

ACKNOWLEDGMENT

Special thanks to Mr. Nino Antulov-Fantulin of Division of Electronics, Laboratory for Information Systems, Rudjer Boskovic Institute, Zagreb, Croatia for helping in understanding some of the concepts of the SIR model proposed in the paper [2]. This work was partially supported by a Nvidia grant under Professor partnership program, a Defence Research and Development Organization (DRDO) grant under Extramural Research and Intellectual Property rights and the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number OCI-1053575.

REFERENCES

[1] Matija Sosic, Mile Sikic, CUDA implementation of the algorithm for simulating the epidemic spreading over large networks. MIPRO, 2012 Proceedings of the 35th International Convention.

[2] Nino Antulov-Fantulin, Alen Lancic, Mile Sikic, FastSIR Algorithm: A Fast Algorithm for simulation of epidemic spread in large networks by using SIR compartment model, arXiv:1202.1639v1[cs.DS], 2012.

[3] G. K.R. Bisset, J. Chen, X. Feng, V.A. Kumar,M.V.Marathe, Epifast: a fast algorithm for large scale realistic epidemic simulations on distributed memory systems, in: Proceedings of the 23rd international conference on Supercomputing, ICS 09, ACM, New York, NY, USA, 2009, pp. 430439.

[4] C.L. Barrett, K.R. Bisset, S.G. Eubank, X. Feng, M.V. Marathe, EpiSimdemics: an efficient algorithm for simulating the spread of infectious disease over large realistic social networks, in: Proceedings of the 2008 ACM/IEEE conference on Supercomputing, SC 08, IEEE Press, Piscataway, NJ, USA, 2008.

[5] Chris J. Kuhlman, V. S. Anil Kumar, Madhav V. Marathe, Henning S. Mortveit, Samarth Swarup, Gaurav Tuli, S. S. Ravi, Daniel J. Rosenkrantz, A GENERAL-PURPOSE GRAPH DYNAMICAL SYSTEM MODELING FRAMEWORK, in:Proceedings of the 2011 Winter Simulation Conference.

[6] Sungpack Hong, Sang Kyun Kim, Tayo Oguntebi, Kunle Olukotun, Accelerating CUDA Graph Algorithms at Maximum Warp, 2011.

[7] NVIDIA CUDA C Programming Guide, version 4.2

[8] Stanford large network dataset collection, http://snap.stanford.edu/data/index.html, 2012.

[9] http://snap.stanford.edu/data/ca-CondMat.html, 2012.

[10] http://snap.stanford.edu/data/loc-gowalla.html, 2012.

[11] http://snap.stanford.edu/data/soc-Epinions1.html, 2012.

[12] http://snap.stanford.edu/data/email-Enron.html, 2012.