RealGraph^{Web}: A Graph Analysis Platform on the Web

Myung-Hwan Jang
Department of Computer Science
Hanyang University
Seoul, South Korea
sugichiin@hanyang.ac.kr

Yong-Yeon Jo
Department of Computer Science
Hanyang University
Seoul, South Korea
jyy0430@hanyang.ac.kr

Sang-Wook Kim*
Department of Computer Science
Hanyang University
Seoul, South Korea
wook@hanyang.ac.kr

ABSTRACT

In this demo, we present RealGraph^{Web}, a web-based platform that provides various kinds of graph analysis services. RealGraph^{Web} is based on RealGraph, a graph engine that addresses the problem of performance degradation in processing real-world big graphs, achieving great performance improvement up to 44 times over existing state-of-the-art graph engines. RealGraph^{Web} runs on a single machine with a web-based interface, thereby allowing users to easily and conveniently enjoy graph analysis services and perform various graph algorithms *anywhere on the web*. In this demo, we present how a user can analyze a graph on RealGraph^{Web} in three steps and get the analysis result quickly via a graphical user interface.

PVLDB Reference Format:
Myung-Hwan Jang, Yong-Yeon Jo, and Sang-Wook Kim. RealGraph^{Web}: A Graph Analysis Platform on the Web. PVLDB, 14(12): 2775-2778, 2021.
doi:10.14778/3476311.3476342

1 INTRODUCTION

A *graph* is a data structure widely used to model relationships among objects in various domains. Many algorithms are used to analyze graphs and discover interesting patterns from them such as node ranking, community detection, and structural analysis [10, 13, 15–17]. The size of real-world graphs are increasing rapidly, which makes an important demand to efficiently analyze such big graphs.

To help applications in analyzing large-scale graphs efficiently and conveniently, various *single-machine based graph engines* have been developed [4, 6, 8]. However, there are still some difficulties for researchers to easily analyze graphs. From the naive user's point of view (i.e., a user with limited programming skills), developing an application and its graphical interface based on the graph engine is a difficult and time-consuming task.

To address this issue, we developed RealGraph^{Web}, a *web-based* platform to provide convenient graph analysis services. It runs on a *single* machine with an interactive web-based interface, thereby allowing users to enjoy various graph analysis services *anywhere on the web*. A user simply uploads her graph to a server and chooses a graph algorithm along with its required parameters. Then, the server performs the selected algorithm on the graph and represents the analysis result *graphically* in different forms such as tables, charts, or graphs.

RealGraph^{Web} is based on our RealGraph [8], a single-machine based graph engine employed for efficient graph analysis. RealGraph addresses the performance degradation problem when processing real-world big graphs, thereby achieving significantly higher speed in compare to those of the state-of-the-art engines such as GraphChi, X-Stream, FlashGraph, and TurboGraph. The graph algorithms already implemented in RealGraph allow users to quickly analyze their graphs without programming them.

In this demo, we will present two main scenarios. The first one is for a user to perform graph analysis *easily and simply* by clicking some options through the web-based graphical user interface (GUI) implemented in RealGraph^{Web}. The second one is to make the user feel the performance improvement in the graph analysis by turning on/off our optimization options employed in RealGraph.

There could be some users who want to see her analysis result *visually* via a graphical interface or some other users who just want to get the analysis result only in a given data format quickly without visualization. To make both demands satisfied, this demo will also show simple and quick graph analysis by using the command-line interface (CLI), in addition to the GUI implemented in RealGraph^{Web}.

The organization of this paper is as follows. Section 2 briefly describes RealGraph, a single-machine based graph engine employed for efficient graph analysis in RealGraph^{Web}. Section 3 illustrates the overall demonstration scenario with RealGraph^{Web}. Finally, Section 4 summarizes and concludes this paper.

2 OVERVIEW OF REALGRAPH

2.1 Architecture

RealGraph [8] is a single-machine based graph engine to efficiently analyze real-world graphs. RealGraph, illustrated in Figure 1, consists of the four following layers: storage management, buffer management, object management, and thread management. The three bottom layers basically follow the design concepts adopted in database storage systems such as WiSS [3], EXODUS [2], and Shore [1]. The storage space is partitioned into *blocks* that are aligned with I/O units; a block contains a number of objects, each of which represents a node and its adjacency list (i.e., a node-id with its adjacent node-ids). If an object cannot fit a given block, it could be stored across multiple blocks. A thread management layer manages a pool of threads and accesses/processes graph data using thread pooling. It also manages attribute data that stores intermediate and final results of a graph algorithm, and two indicators that store the information about the nodes to be processed at the current/next iterations.

*Corresponding author.

Proceedings of the VLDB Endowment, Vol. 14, No. 12 ISSN 2150-8097.
doi:10.14778/3476311.3476342

Table 1: Performance comparison of RealGraph with various graph engines on Yahoo and Twitter data (Times in sec.) [8]

vs. single-machine (on Yahoo)	Graph algorithms			System environment			
	BFS	WCC	PageRank	Machine	#Cores	Mem. Size	Storage
RealGraph	19	385	269	Single PC with Intel i7	4	16G	SSD
FlashGraph	122	O.O.M	O.O.M				
TurboGraph	843	2,709	441				
GridGraph	12,415	3,673	14,078				
GraphChi	O.O.T	3,074	O.O.T				
X-Stream	O.O.T	8,016	O.O.T				
vs. distributed-system (on Twitter)	Graph algorithms			System environment			
	BFS	WCC	PageRank	Machine	#Cores	Mem. Size	Storage
RealGraph	14	34	94	Single PC with Intel i7	4	16G	SSD
PowerGraph	-	-	72	64 Amazon EC2 cc1.4xlarge	8	23G	HDD
GraphLab	-	244	249	16 Amazon EC2 m2.4xlarge	8	68G	HDD
GraphX	-	251	419	16 Amazon EC2 m2.4xlarge	8	68G	HDD
Giraph	-	200	596	16 Amazon EC2 m2.4xlarge	8	68G	HDD

2.2 Features and Processing Steps

The important features of RealGraph for providing high performance in big graph analysis are as follows:

Hierarchical indicator: RealGraph uses a hierarchical indicator composed of *multi-level* bit vectors instead of a *flat single-level* bit vector. Its lowest-level has a flat bit vector, each bit corresponding to a node. A higher-level consists of a bit vector where each bit compresses a *range* of a lower-level. Here, a bit set as 1 implies that its lower-level range needs scanning; otherwise, the range does not need scanning. Scanning with this hierarchical indicator starts from the highest-level (i.e., root) and moves down to a lower-level only if a bit in its higher-level is set as 1. As a result, most of ranges in the sparse lowest-level are skipped, avoiding unnecessary scanning.

Block-wise workload allocation: RealGraph uses block-wise workload allocation where each thread processes a *fixed-size block* instead of a *variable-size object*. While a block has multiple objects of variable sizes, the total size of objects in a block is almost identical. This idea enables to have uniform distribution of workloads over threads even when processing a big hub node in a graph.

Efficient data layout: RealGraph stores graphs by considering *data locality* [7]. The data accessed together in graph algorithms are placed in the *same* or *adjacent locations* in the storage. This idea not only reduces the number of accessed blocks but also increases the degree of sequential accesses of blocks.

RealGraph performs an algorithm on a graph as follows. First, given some bits set as 1 in the indicator, corresponding to the nodes in the graph to be processed, RealGraph identifies these nodes from the indicator in the current iteration. It examines whether each block containing the nodes is in the buffer. If the block does not exist in the buffer, then it loads the block from storage to the buffer. Then, it assigns the nodes in the loaded block to a thread for processing. Threads perform the functions and update the results by accessing the adjacency lists of the nodes and set the bits corresponding to those nodes as 1 in the indicator for the next iteration. This process is performed iteratively until no more bits set as 1 in the indicator.

2.3 Performance

Table 1 shows the performance comparison of RealGraph with single-machine based graph engines on the biggest Yahoo[1] dataset having 1,413M nodes and 6.6B edges (top-half) and distributed-system based graph engines on the Twitter[2] dataset having 61M nodes and 1.4B edges (bottom-half). In this table, "O.O.M" denotes the out-of-memory case; "O.O.T" does the case when the experiment does not finish in 24 hours; "-" does the absence of the experiment in the corresponding original work. We observe that the performance of RealGraph is an *order-of-magnitude* better than the state-of-the-art single-machine based graph engines, such as TurboGraph [6] and GridGraph [18], and does not cause out-of-memory cases like FlashGraph [4]. Also, RealGraph equipped with low computing power consistently performs better than distributed-system based engines such as PowerGraph [5] and GraphLab [11].

3 DEMONSTRATION OF REALGRAPH^{WEB}

RealGraph^{Web} is a web-based platform that provides an interactive web-based interface, allowing users to access various graph analysis services by performing graph algorithms easily and conveniently anywhere on the web. Graph analysis services using RealGraph^{Web} are performed in three steps: graph upload, algorithm execution, and result visualization & download steps (Refer to Figure 1). In this section, we present how a user analyzes a graph on RealGraph^{Web} via these three steps.

3.1 Graph Upload

A user *uploads* her target graph to a RealGraph^{Web} server in the *graph upload* step. RealGraph^{Web} provides the user with a sample input graph; this allows the user to upload her own graph in the correct format. Then, it converts the received graph to its own structure and stores it in the storage of RealGraph.

[1] Yahoo: https://webscope.sandbox.yahoo.com
[2] Twitter: https://snap.stanford.edu/data/twitter-2010.html

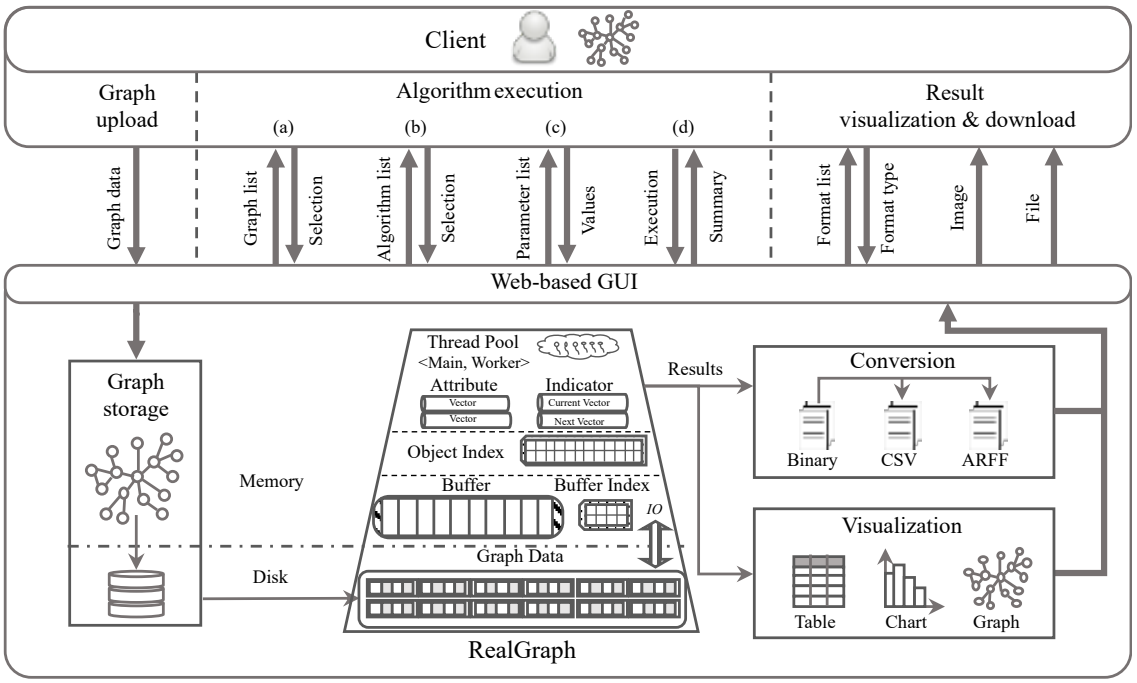


Figure 1: An overview of RealGraph^{Web}.

3.2 Algorithm Execution

Figures 1-(a)~(d) show the substeps of an *algorithm execution* step. The server provides a list of graphs already stored in RealGraph^{Web}, among which the user selects her target for analysis (Figure 1-(a)). The user sees the characteristics of the graph, such as numbers of nodes and edges. The server provides various graph algorithms commonly implemented in many graph engines, such as Outdegree/Indegree distribution, breadth first search (BFS) [14], PageRank [13], and weakly connected component (WCC) [15]. It also provides additional graph algorithms, such as betweenness centrality (BC) [12], hypertext induced topic selection (HITS) [9], random walk with restart (RWR) [17], and community detection (CD) [10], among which a user selects one for her analysis (Figure 1-(b)). Each algorithm has its own parameters: for instance, BFS requires the starting node while PageRank does the number of iterations. A user sets parameters as the values she wants (Figure 1-(c)). Then, she requests the execution of the algorithm on the graph (Figure 1-(d)). The server stores the result and the elapsed time of algorithm execution in a binary file and prints them in text format after the execution is completed.

3.3 Result Visualization & Download

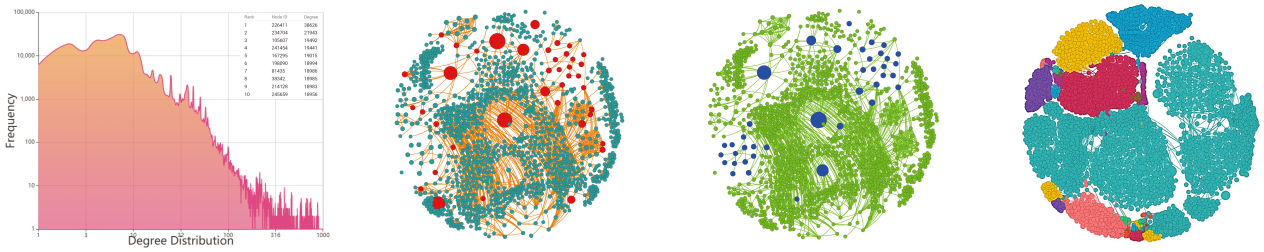
The server provides visualization and download functions in a *result visualization & download* step. It visualizes the execution result in a binary file in terms of a table, a chart, or a graph by using the libraries such as Matplotlib, Echarts, and Shingle.js. The server allows the user to *download* the image of a visualized result and also the whole binary file that contains the execution result. It provides an option to transform a binary file to a CSV file or an ARFF file *for compatibility* because some tools such as Matlab, R, and Weka require those formats.

Figure 2 shows some examples of visualizing the results of different graph algorithms. For example, in the case of degree distribution, the frequencies of nodes having different degrees are presented in a chart (Figure 2-(a)). In the case of PageRank, it visualizes the graph in such a way that the size of a node is proportional to its PageRank score and the nodes with top-N PageRank scores are highlighted with different colors (Figure 2-(b)); in the case of HITS, where there exist two kinds of scores (e.g., hub score and authority score), graph visualization is similar to the previous PageRank case, but having the two kinds of result graphs for the two scores (Figure 2-(c)); in the case of CD, it visualizes the graph in such a way that the nodes belonging to the *same* community have the *same* color and the nodes belonging to *different* communities have *different* colors (Figure 2-(d)).

3.4 Demo Scenarios

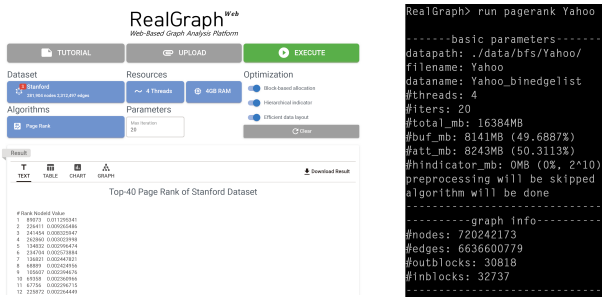
We will present two main scenarios in this demo. The first scenario will show how a naive user performs graph analysis just by clicking some options through the web-based GUI of RealGraph^{Web}. A user will upload graphs, analyze them, and visualize/download results, simply following the three steps provided by RealGraph^{Web}. The second scenario will make the user feel efficiency of RealGraph^{Web}; specifically, during the graph analysis, she will realize the effect of performance improvement obtained by the optimization techniques employed in RealGraph through turning on/off the provided optimization options.

In addition to the main scenarios, users will enjoy performing various algorithms on different graphs provided in RealGraph^{Web}.



(a) Degree distribution (b) PageRank (c) HITS (hub scores) (d) Community detection

Figure 2: Visualization examples.



(a) Graphical user interface (b) Command-line interface

Figure 3: RealGraph^{Web} interface.

Sometimes, users just want to get the analysis result quickly without visualization because the visualization takes very long time in the case of analyzing big graphs. For this case, we will use a simple interactive command-line interface (CLI) implemented in RealGraph^{Web}. This CLI allows users to perform fast graph analysis and produce the result in various data formats supported by RealGraph^{Web}. Figure 3 shows the GUI and CLI implemented in RealGraph^{Web}.

4 CONCLUSIONS

In this demo, we presented RealGraph^{Web}, a web-based platform that allows a user to utilize various graph analysis services without requiring programming. With two demo scenarios, we showed that the web-based GUI of RealGraph^{Web} makes it easy and convenient for users to experience those services anywhere and anytime on the web and that the graph analysis performed in RealGraph^{Web} is very efficient, thanks to the optimization techniques employed in RealGraph, our single-machine based graph engine.

ACKNOWLEDGMENTS

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) and the National Research Foundation of Korea (NRF) grants funded by the Korea government (Ministry of Science and ICT, MSIT) (No. 2020-0-01373, No. NRF-2020R1A2B5B03001960, and No. 2018R1A5A7059549), and supported by Samsung Electronics Co., Ltd (Project No. IO201209-07876-01).

REFERENCES

[1] Michael J Carey, David J DeWitt, Michael J Franklin, Nancy E Hall, Mark L McAuliffe, Jeffrey F Naughton, Daniel T Schuh, Marvin H Solomon, CK Tan, Odysseas G Tsatalos, Seth J White, and Michael J Zwilling. 1994. Shoring up persistent applications. In *Proceedings of the ACM International Conference on Management of Data*. ACM, 383–394.

[2] Michael J Carey, David J DeWitt, Joel E Richardson, and Eugene J Shekita. 1986. *Object and file management in the EXODUS extensible database system*. University of Wisconsin-Madison. Computer Sciences Department.

[3] H-T Chou, David J Dewitt, Randy H Katz, and Anthony C Klug. 1985. Design and implementation of the Wisconsin storage system. *Software: Practice and Experience* 15, 10 (1985), 943–962.

[4] Zheng Da, Disa Mhembere, Randal Burns, Joshua Vogelstein, Carey E Priebe, and Alexander S Szalay. 2015. FlashGraph: Processing billion-node graphs on an array of commodity SSDs. In *Proceedings of the USENIX International Conference on File and Storage Technologies*. 45–58.

[5] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin. 2012. Powergraph: Distributed graph-parallel computation on natural graphs. In *Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation*. 17–30.

[6] Wook-Shin Han, Sangyeon Lee, Kyungyeol Park, Jeong-Hoon Lee, Min-Soo Kim, Jinha Kim, and Hwanjo Yu. 2013. TurboGraph: A fast parallel graph engine handling billion-scale graphs in a single PC. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*. ACM, 77–85.

[7] Yong-Yeon Jo, Jiwon Hong, Myung-Hwan Jang, Jae-Geun Bang, and Sang-Wook Kim. 2016. Data locality in graph engines: Implications and preliminary experimental results. In *Proceedings of the ACM International Conference on Information and Knowledge Management*. ACM, 1885–1888.

[8] Yong-Yeon Jo, Myung-Hwan Jang, Sang-Wook Kim, and Sunju Park. 2019. RealGraph: A graph engine leveraging the power-law distribution of real-world graphs. In *Proceedings of the World Wide Web Conference*. ACM, 807–817.

[9] Jon M Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *J. ACM* 46, 5 (1999), 604–632.

[10] Xin Liu and Tsuyoshi Murata. 2010. Advanced modularity-specialized label propagation algorithm for detecting communities in networks. *Physica A: Statistical Mechanics and its Applications* 389, 7 (2010), 1493–1500.

[11] Yucheng Low, Danny Bickson, Joseph Gonzalez, Carlos Guestrin, Aapo Kyrola, and Joseph M Hellerstein. 2012. Distributed GraphLab: A framework for machine learning and data mining in the cloud. *Proceedings of the Very Large Databases Endowment* 5, 8 (2012), 716–727.

[12] Mark EJ Newman. 2005. A measure of betweenness centrality based on random walks. *Social Networks* 27, 1 (2005), 39–54.

[13] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web*. Technical Report. Stanford InfoLab.

[14] Robert Sedgewick and Kevin Wayne. 2011. *Algorithms*. Addison-Wesley professional.

[15] Kenji Suzuki, Isao Horiba, and Noboru Sugie. 2003. Linear-time connected-component labeling based on sequential local operations. *Computer Vision and Image Understanding* 89, 1 (2003), 1–23.

[16] Gábor Takács and Domonkos Tikk. 2012. Alternating least squares for personalized ranking. In *Proceedings of the ACM Conference on Recommender Systems*. ACM, 83–90.

[17] Hilmi Yildirim and Mukkai S Krishnamoorthy. 2008. A random walk method for alleviating the sparsity problem in collaborative filtering. In *Proceedings of the ACM Conference on Recommender Systems*. ACM, 131–138.

[18] Xiaowei Zhu, Wentao Han, and Wenguang Chen. 2015. Gridgraph: Large-scale graph processing on a single machine using 2-level hierarchical partitioning. In *USENIX Annual Technical Conference*. 375–386.