Synthetic Traffic Model of the Graph500 Communications

Pablo Fuentes[1], Enrique Vallejo[1], José Luis Bosque[1], Ramón Beivide[1], Andreea Anghel[2], Germán Rodríguez[3], Mitch Gusat[2], and Cyriel Minkenberg[3] *

[1] University of Cantabria
{fuentesp, vallejoe, bosquejl, beividej}@unican.es
[2] IBM Zurich Research Laboratory
{aan, mig}@zurich.ibm.com
[3] Rockley Photonics
german.rodriguez.herrera@gmail.com,
cyriel.minkenberg@rockleyphotonics.com

Abstract. As BigData applications have gained momentum over the last years, the Graph500 benchmark has appeared in an attempt to steer the design of HPC systems to maximize the performance under memory-constricted application workloads. A realistic simulation of such benchmarks for architectural research is challenging due to size and detail limitations, and synthetic traffic workloads constitute one of the least resource-consuming methods to evaluate the performance. In this work, we propose a synthetic traffic model that emulates the behavior of the Graph500 communications. Our model is empirically obtained through a characterization of several executions of the benchmark with different input parameters. We verify the validity of our model against a characterization of the execution of the benchmark with different parameters. Our model is well-suited for implementation in an architectural simulator.

1 Introduction

BigData applications have become ubiquitous and gather the interest of system architects and designers. The Graph500 benchmark [1] appeared in 2010 with the aim of influencing the design of new systems, so they better adjust to the memory- and IO-bounded requirements of data intensive applications. Based on the execution of a BFS within a graph, it is currently one of the most known BigData-focused benchmarks [3].

* The authors would like to thank the European HiPEAC Network of Excellence for partially funding this work through a Collaboration Grant, as well as Cristóbal Camarero for his help. This work has been supported by the Spanish Ministry of Education, FPU grant FPU13/00337, the Spanish Science and Technology Commission (CICYT) under contract TIN2013-46957-C2-2-P, and by the Mont-Blanc project. The Mont-Blanc project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 671697.

Therefore, it is strongly useful for the evaluation and design of parallel computers, specially their memory and network subsystems.
Network simulators constitute a useful tool for network architects in the design and evaluation of new systems. Synthetic traffic models have smaller computational and memory requirements than full system and trace-driven simulation alternatives while retaining the core characteristics of the workload they represent. However, synthetic traffic models have traditionally consisted of permutations to determine the destination or set of destinations for the messages from a given node, which isn't a fitting scheme for the behavior of BigData applications.
The objective of this work is to establish a network traffic model of the Graph500 benchmark workload. Our model replicates the staged structure of the benchmark with large batches of uniformly distributed messages ending in a collective allreduce operation. Total amount of messages per stage is defined as a function of message aggregation and the number of explored edges, obtained empirically from a characterization of several benchmark executions with different input parameters. A similar procedure can be followed to adapt our model to emulate a BFS execution upon graphs of a different nature. This model can be handily integrated in a network simulator to forecast the network impact on system performance upon the Graph500 benchmark workload. It achieves good scalability both in size of the simulated network and maximum graph size explored in the benchmark, without requiring intensive computation.

2 Analysis of the Benchmark Communications

In [10] Suzumura *et al.* offer a thorough description of the different implementations of the Graph500 benchmark, and works from Anghel *et al.* [2] and Fuentes *et al.* [6] provide a more thorough characterization of the communications in the *simple* implementation of the benchmark.
Communications along the benchmark execution are structured into multiple batches of point-to-point messages succeeded by a point-to-point notification to all other processes which signals the end of a tree level. These batches are separated by a phase of synchronization through an all-reduce comprising a reduction and a broadcast, and correspond to the levels of the tree obtained from the BFS execution. The amount of messages is close to evenly distributed across each stage but varies significantly between tree levels, with two big levels typically representing $\sim 80\%$ of the total. The impact of the point-to-point messages which signal the end of a tree level is negligible. The likewise sporadic all-reduce is only important because it synchronizes the generation of new messages. Therefore, from a network usage perspective the major communications are the point-to-point exchanges during each tree level. These communications are highly homogeneous spatially due to the even distribution of the graph across processes. Graph500 exploits message aggregation to reduce network traffic, with every message grouping multiple queries up to a value named *coalescing size.*
Graphs in the benchmark are generated through a Kronecker matrix product similar to the Recursive MATrix (R-MAT) scale-free graph generation algorithm [4]. Graphs generated by R-MAT emulate the behavior

of small-world phenomena, where a small fraction of the nodes (or *vertices*) have a significantly large number of direct connections (or *edges*) with other nodes, and a large proportion of the vertices have a low number of edges (or *vertex degree*). Previous works such as [4] and [8] characterize the distribution of the vertex degree in such graphs as power-law or lognormal. Groër *et al.* [7] provide a more accurate model as a series expansion from normal distributions, and Seshadhri *et al.* [9] simplify it as a combination of a lognormal and an exponential tail distribution.
In our model we consider a lognormal distribution of the vertex degree to make it simpler to implement yet sufficiently accurate. This distribution is used to select the degree of the root vertex, and can be readily replaced by the series expansion of Groër [7] or the formula of Seshadhri [9] should a more precise distribution be needed.

3 Graph500 Network Traffic Model

Our model consists of a traffic generator structured in multiple stages: for each stage, a given number of messages is dispatched from every process. Once a process has sent all its messages, it sends one message to every other process (representing the end-of-phase notification) and it enters into a collective all-reduce operation. This operation consists of a reduce operation (where all the process send to the same destination) and a broadcast (in which the destination spreads the result from the data gathering). Within each stage, the traffic consists of point-to-point messages uniformly distributed temporally and spatially.
Message generation rate depends fundamentally on the node capabilities of the system, and we introduce it as an input parameter to the model that is constant across any execution. Message destination is randomly selected among all the other processes in the application. Each process sends in every level a fixed amount of messages defined through a set of equations before entering the all-reduce operation. These equations have been obtained through a statistical analysis of the average from a set of executions of the benchmark for different input parameters.
This analysis has been conducted by establishing a linear combination that fits the observed values through a model fitting tool based on the function described by Chambers in [5]. The coefficients of said linear combinations have then been generalized to follow the variations with the input parameters. The measurements are oblivious to the infrastructure employed, so they can be extrapolated to any other system. In each stage the amount of messages delivered per process is expressed as a function of the number of edges explored per tree level and the amount of message aggregation that is being performed, represented by a *coalescing size* parameter that is by default constant in the benchmark implementation.
Table 1 presents a summary of the abbreviations employed in the model equations, with a brief description of each parameter. The number of messages sent in the point-to-point communications is determined by Eq. 1, considering the *coalescing size* as a parameter of the model.

$$m_{proc} = \frac{1}{c_s} \cdot \frac{E_l}{p} \cdot \frac{p-1}{p} \tag{1}$$

Table 1: List of abbreviations employed in the equations.

Abbr.	**Parameter**	**Description**
m_{proc}	Messages per process	Number of messages sent from each process.
c_s	Coalescing size	Amount of explored edges aggregated per message.
p	Number of processes	Number of processes employed in the benchmark execution.
E_l	Edges per tree level	Total number of edges explored within each stage of the BFS.
d_r	Degree of the root	Number of edges connected to the root vertex.
s	Scale	Base 2 log of the number of vertices in the graph.
f_e	Edgefactor	Half of the average vertex degree.
l	Tree level	Stage of the tree in the BFS execution, starting at 0.

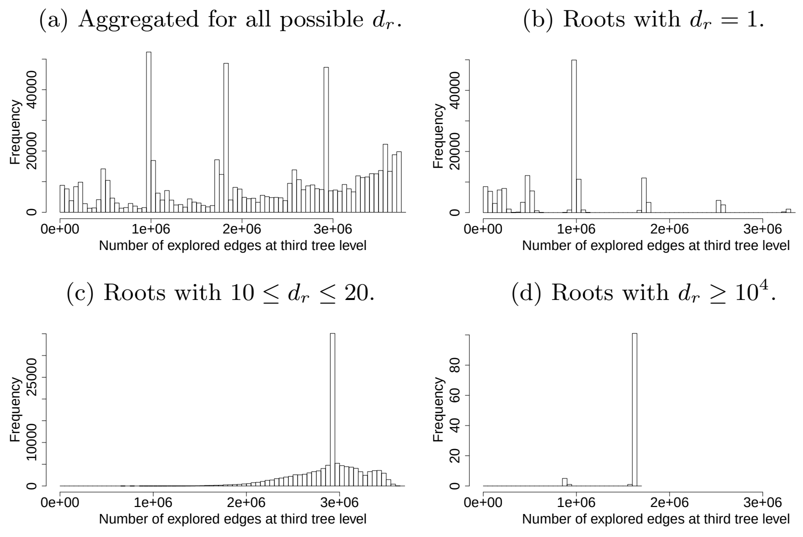

Fig. 1: Histogram of the number of explored edges in the third tree level, with different root degree d_r. Graph with scale $s = 17$ and edgefactor $f_e = 16$.

3.1 Relation between Number of Edges per Tree Level and Root Vertex Degree

The total number of explored edges across the whole BFS execution is almost constant for a pair of given *scale* and *edgefactor* parameter values, but its distribution between tree levels varies heavily. Simply averaging multiple executions with the same input parameters masks the actual behavior, as occurs in the histogram in Fig. 1a which does not seem to follow any clear distribution. This histogram corresponds to the number of new edges traversed in a graph during the third tree level, obtained by running a BFS for every possible root in the graph.
Our approach is to determine the number of explored edges per tree level as a function of the degree of the root vertex, which heavily influences the distribution. Figs. 1b-1d display the average distribution of the number of explored edges during the third tree level for the same graph in Fig. 1a, with three different ranges of root degree: roots with only one neighbour

(Fig. 1b), roots with 10 to 20 neighbors (Fig. 1c) and roots with a high root degree of 10,000 neighbors or more (Fig. 1d). It can be observed that for each range of the root degree there is only one predominant peak, as opposed to the histogram for all roots in the graph shown in Fig. 1a where there were multiple peaks of similar impact.
This heavy dependence on the root degree occurs because roots with low vertex degree will originate a low amount of communications at first tree levels, shifting the biggest part of the graph traversal to higher levels, whereas roots with high degree will rapidly explore the majority of the graph and present low (or non-existent) communication at higher levels. Our model characterizes the mean amount of visited edges per tree level as a function of the root degree and the graph parameters (*scale* and *edgefactor*). This method has the benefit of adjusting reasonably to the observed behavior while remaining low-demanding computer-wise. The first tree level is trivially determined as the root degree. The selection of the root degree will be performed randomly following a lognormal distribution, as discussed in Section 2.
Fig. 2 depicts the average number of edges upon the root degree, broken down per tree level. X-axis is displayed in logarithmic scale. Results correspond to the same data used for the histograms in Fig. 1. The three blocks circled in red correspond with the average number of explored edges in the third tree level whose distribution was presented in Figs. 1b-1d. Some values in Fig. 2 are interpolated, as not all the vertex degrees are present in a graph. Note that Y-axis in Fig.2b is also in logarithmic scale. The aggregated amount of edges remains almost constant, since the size of the graph is independent of the vertex selected as root (and consequently the amount of edges to traverse during the BFS will be similar). However, the distribution of those edges among the tree levels varies significantly, further confirming our motivation to relate the communications to the vertex degree at the tree root. In the following subsection we will present an equation linking the number of edges per tree level to the root degree.

3.2 Number of Explored Edges per Tree Level

Our model approximates the evolution of the average number of explored edges per vertex for each tree level (as shown in Fig.2b) through a polynomial of degree 2 as the one described in Eq. 2 (in the next page), where d_r is the root degree, and the A, B and C factors depend on the *scale*, *edgefactor* and tree level. The first tree level is an exception to this, with the number of explored edges being defined directly as the degree of the root vertex (and originating messages only at one process, the one hosting the root). We consider a notation for the tree level l that spans from $l = 0$.
We have approximated A, B and C in each tree level to fit the observed values for several combinations of *scales* $s = 16, 17, 18, 19, 20, 23, 25$ and *edgefactors* $f_e = 16, 20, 32, 45, 64$. For each explored combination, we have run a BFS for every vertex in the graph, and measured the average number of explored edges per level for each root vertex degree. Then we have obtained an expression that fits the evolution of each of the

(a) Stacked values. (b) Curves.

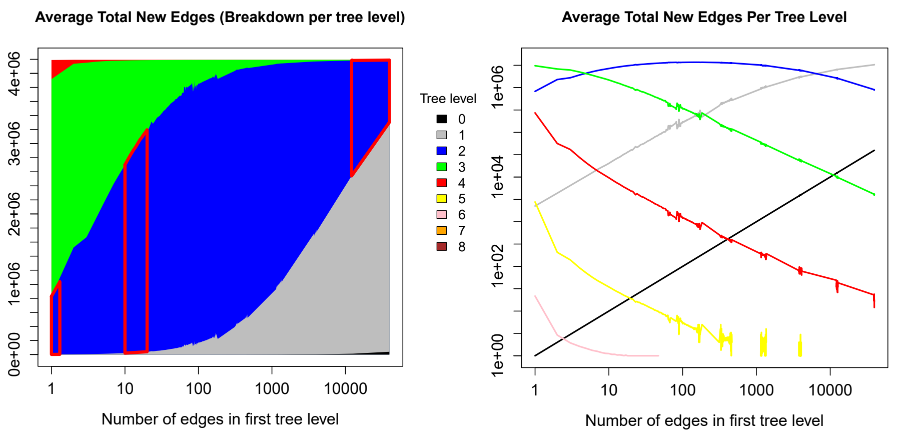

Fig. 2: Number of explored edges per root degree, broken down per tree level.

parameters in Eq. 2 with the *scale* and *edgefactor* parameters as well as the tree level, presented in Eqs. 3 - 5. Additionally, we know that the maximum number of edges that can be traversed is twice the number of edges in the graph, because each edge is traversed in both ways. We limit the total number of edges traversed across all tree levels below that limit; when none the processes can send any more messages, the execution of the model is ended.

$$\ln\left(\bar{E}_l\right) \approx A \cdot \ln^2\left(d_r\right) + B \cdot \ln\left(d_r\right) + C, \qquad l \geq 1 \tag{2}$$

$$A = -0.133 + 0.0046 \cdot s + e^{l + 0.01257 \cdot f_e - 0.1829 \cdot s - 3.6554} \tag{3}$$

$$B = 2 - l \cdot (0.91 + 0.002 \cdot f_e - 0.012 \cdot s) \tag{4}$$

$$C = e^{1 + (1 + 0.004 \cdot f_e) \cdot e^{-\left(0.0011 \cdot s^2 - 0.0451 \cdot s + 2.09\right)} \cdot l \cdot (4.5 + 0.078 \cdot s - l)} \tag{5}$$

4 Model Validation

To validate our model, we have employed measured values from a set of Graph500 executions with parameters different than those employed to obtain the model. Since the impact of message aggregation is clearly defined by Eq. 1, we focus the validation in a crosscheck between the prediction from our model and the values obtained through the measurement of an actual run of the benchmark. Figure 3 displays the average number of explored edges $\bar{E}_l$ for all possible root vertices in a graph of *scale* $s = 22$ and *edgefactor* $f_e = 16$. Points correspond to the measured values, whereas lines are the fittings obtained through our model. The fitting curves approximate clearly the observed behavior, following the same trends as the measurements for every stage of the execution. The

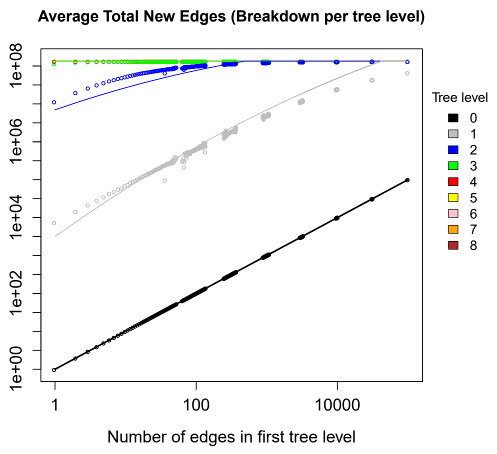

Fig. 3: Validation of the model. Points correspond to measured average values from a real execution, lines correspond to the fittings from the model.

model reproduces the staged behavior and replicates the dependence on the root degree, observing a similar proportion between the impact of each stage in the total amount of explored edges. From the third level $l = 2$ we observe that the model result is truncated for large root degrees when the maximum number of edges given by the *scale* and *edgefactor* are explored.
The dynamic range of the observed values is very large due to its logarithmic nature; this implies that any deviation in the prediction will incur in a very large absolute error. Still, the relative error of our model for this second tree level is lower than 18% in more than 90% of the cases. For the third tree level, which amounts the largest amount of communications for most root degrees, the model is still able to reproduce the same behavior with an average relative error of 12.5%. The total number of explored edges across the graph presents a relative error below 12%, which is corrected when the maximum value is reached and the edges in the last levels are truncated. A similar analysis has been conducted upon graphs of *scale* $s = 18$ and *edgefactor* $f_e = 40$, with analogous results.

5 Conclusions

Current evaluations of BigData workloads consist mostly of full-system simulations of the real applications, or rely on the use of traces. Both options limit severely the size and detail of the network that can be investigated via simulation - which confers observability otherwise unattainable. Here we have introduced a novel computationally non-intensive synthetic traffic model of the most scalable implementation of the Graph500 benchmark. We have analyzed the distribution of the benchmark communications in stages and its relation with the number of explored edges

per tree level. Furthermore, we have identified a strong connection to the degree of the vertex selected as root of the tree.
We have modeled the benchmark behavior as a set of stages of point-to-point messages separated by all-reduce collective operations. The number of messages is defined as a linear model of the benchmark parameters (*scale*, *edgefactor*) for each stage within the BFS computation (*tree level*). Using an empirical characterization of actual benchmark executions for different graph parameters, we have defined a set of equations to compute the average number of edges per tree level for any given tree root degree; the degree of the root vertex is decided randomly following a lognormal distribution.
As next steps, we intend to expand our model using a distribution for the characterization, and to implement it in a network simulator.

References

1. Graph500 benchmark (May 2016), `http://www.graph500.org/`
2. Anghel, A., Rodriguez, G., Prisacari, B.: The importance and characteristics of communication in high performance data analytics. In: Workload Characterization (IISWC), 2014 IEEE International Symposium on. pp. 80–81. IEEE (2014)
3. Beamer, S., Asanovic, K., Patterson, D.: Locality exists in graph processing: Workload characterization on an ivy bridge server. In: Workload Characterization (IISWC), 2015 IEEE International Symposium on. pp. 56–65 (Oct 2015)
4. Chakrabarti, D., Zhan, Y., Faloutsos, C.: R-mat: A recursive model for graph mining. In: Proceedings of the 2004 SIAM International Conference on Data Mining. pp. 442–446
5. Chambers, J., Hastie, T.: Statistical Models in S. Wadsworth & Brooks/Cole (1992)
6. Fuentes, P., Bosque, J.L., Beivide, R., Valero, M., Minkenberg, C.: Characterizing the communication demands of the graph500 benchmark on a commodity cluster. In: Proceedings of the 2014 IEEE/ACM Int. Symposium on Big Data Computing. pp. 83–89
7. Groër, C., Sullivan, B.D., Poole, S.: A mathematical analysis of the r-mat random graph generator. Networks 58(3), 159–170 (2011)
8. Kim, M., Leskovec, J.: Multiplicative attribute graph model of real-world networks. In: Algorithms and Models for the Web-Graph, pp. 62–73. Springer (2010)
9. Seshadhri, C., Pinar, A., Kolda, T.G.: An in-depth study of stochastic kronecker graphs. In: Data Mining (ICDM), 2011 IEEE 11th International Conference on. pp. 587–596. IEEE (2011)
10. Suzumura, T., Ueno, K., Sato, H., Fujisawa, K., Matsuoka, S.: Performance characteristics of Graph500 on large-scale distributed environment. In: Workload Characterization (IISWC), 2011 IEEE International Symposium on. pp. 149–158. IEEE (2011)