Análise de Desempenho de Plataformas de Processamento de Grafos

Daniel N. R. da Silva,[1] **Klaus Wehmuth,**[1]
Carla Osthoff,[1] **Ana Paula Appel,**[2] **e Artur Ziviani**[1]

[1] Laboratório Nacional de Computação Científica (LNCC)

[2]IBM Research Brazil

{dramos,klaus,osthoff,ziviani}@lncc.br, apappel@br.ibm.com

Resumo. *A análise de redes complexas, representadas por grafos, se aplica a diversas áreas do conhecimento. Este artigo analisa o desempenho de plataformas de processamento de grafos representativas de abordagens diversas ao problema. Também são considerados na análise realizada algoritmos de análise de grande interesse da comunidade (conectividade, centralidade e caminho), além de um conjunto de redes sintéticas e reais com características topológicas e dimensões diversas. Os resultados experimentais obtidos contribuem com diretivas para que os interessados possam melhor elencar a plataforma de processamento de grafos mais eficiente a seus interesses de análise.*

Abstract. *The analysis of complex networks, represented by graphs, finds applications in several areas of knowledge. In this paper, we analyze the performance of graph processing platforms, which represent different approaches to the problem. In our evaluation, we also consider analysis algorithms that are of great interest of the community (connectivity, centrality, and path), as well as a set of real and synthetic networks with diverse topological characteristics and dimensions. The obtained experimental results contribute with guidelines for those interest to better select the most efficient graph processing platform to their analytics interests.*

1. Introdução

Um grande número de estruturas e fenômenos em diversas áreas do conhecimento pode ser descrito através de redes complexas [Newman, 2010]. O processamento de redes complexas de grande volume apresenta novos desafios a academia e indústria, trazendo à tona a necessidade do uso de computação tanto paralela quanto distribuída. Entretanto, a eficiência no uso de paralelismo computacional assim como na distribuição dos dados sofre com características inerentes aos grafos, tais como computação dirigida a dados, problemas não estruturados, localidade pobre e razão alta entre acesso aos dados e computação [Lumsdaine et al., 2007]. Desse modo, aplicar a computação distribuída e paralela ao processamento de grandes grafos é um tópico de pesquisa relevante, havendo grande interesse pelo desempenho de plataformas de processamento de grafos.

Os tipos de análise, algoritmos e métricas influenciam o desempenho computacional do processamento de grafos de larga escala [Doekemeijer e Varbanescu, 2014, Pires et al., 2015]. De modo macroscópico, a análise de grafos pode ser dividida nas categorias: caminho, conectividade, comunidade e centralidade.[1] Já os algoritmos para pro-

[1] http://www.ibmbigdatahub.com/blog/what-graph-analytics

cessamento de grafos podem ser divididos nas classes: estatística, travessia, componentes conexos, detecção de comunidades, evolução e outros [Chakrabarti e Faloutsos, 2006]. Há uma diferença entre os termos análise e algoritmo: o primeiro se refere a tarefas que podem englobar mais de um algoritmo, até mesmo, algoritmos de classes diferentes.[2] Além disso, as métricas referem-se à abrangência da informação utilizada: local, global e combinação entre local e global [Silva e Zhao, 2016]. Por exemplo, o cálculo do grau de cada vértice é local, enquanto o grau médio (conectividade) da rede é global. Portanto, é pertinente conhecer a relação entre o conjunto algorítmico (análise, algoritmos e métricas) e a eficiência computacional no processamento de grafos de larga escala.

Logo, o desafio de compreender o processamento de grafos de larga escala envolve entender as relações e implicações de quatro aspectos: (i) **redes**: quais características (p.ex. grau de agrupamento, distribuição do grau dos vértices ou diâmetro) afetam substancialmente o desempenho computacional; (ii) **arquitetura computacional**: quais características do sistema computacional (p.ex. distribuição de memória, rede, ou uso de aceleradores) afetam o desempenho do processamento; (iii) **plataforma**: quais as consequências do uso de determinada plataforma de *software* (p.ex. modelo de programação, otimizações, *check pointing*, tolerância a falhas); e (iv) **conjunto algorítmico**: qual o desempenho dos tipos de análises, algoritmos e métricas nos outros três aspectos. Esses aspectos são claramente inter-relacionados e sua interdependência impacta diretamente no desempenho final do processamento de grafos em larga escala.

Neste artigo, é realizada uma avaliação de desempenho de plataformas de processamento de grafos. A principal contribuição do artigo reside na comparação experimental do desempenho computacional de um conjunto não presente na literatura de 3 plataformas para processamento de grafos, em um ambiente de memória compartilhada, para 3 algoritmos de categorias e análises distintas, e 23 grafos — 18 redes sintéticas *Barabási-Albert* (BA) e *Erdős Rényi* (ER), 1 grafo aleatório e 4 traços de redes reais — de até 1 bilhão de arestas. A biblioteca para análise de redes $\mathrm{NetworKit}$ ($\approx 54\%$ dos casos) e o *framework* específico $\mathrm{PowerGraph}$ ($\approx 42\%$ dos casos) foram as plataformas mais eficientes nos experimentos. A análise realizada investiga os fatores distintos que impactam diferentemente o desempenho dessas plataformas. Portanto, os resultados experimentais obtidos contribuem com diretivas para que os interessados possam melhor elencar a plataforma de processamento de grafos mais eficiente a seus interesses de análise.

O restante desse artigo está organizado como a seguir. Na Seção 2 são apresentados os trabalhos relacionados. Na Seção 3 discute-se a metodologia dos experimentos: algoritmos, plataformas e redes. Na Seção 4 são analisados os resultados. Finalmente, na Seção 5 são apresentadas as conclusões e propostas de trabalhos futuros.

2. Trabalhos relacionados

Atualmente, há mais de oitenta plataformas destinadas ao processamento de grafos [Doekemeijer e Varbanescu, 2014]. Dentre essas, este artigo destaca e classifica um subconjunto não exclusivo, o qual tem ganho notoriedade nos últimos anos na academia e indústria (ver Figura 1). Esse grande número de plataformas disponíveis para proces-

[2] Medir a relevância de um vértice através da razão entre sua importância em uma comunidade e na rede. A análise é de centralidade, mas pode englobar algoritmos de centralidade baseados em travessia como *Closeness Centrality* e de detecção de comunidade como *Community Detection by Label Propagation*.

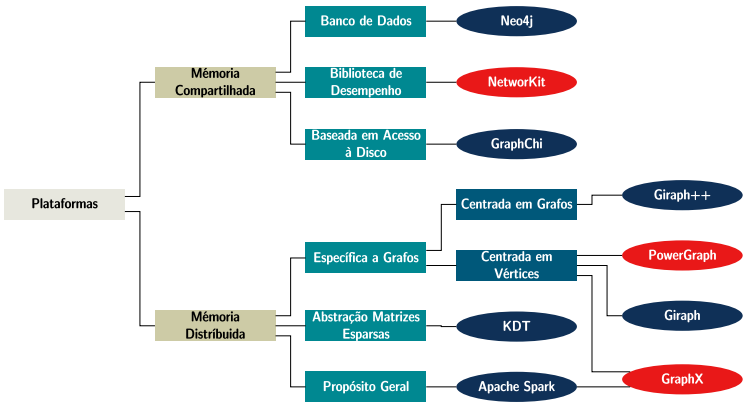


Figura 1. Principais plataformas para processamento de grafos.

samento de grafos de larga escala tem também aumentado o interesse em experimentos comparativos dessas ferramentas (ver Tabela 1).

Um ponto importante para a avaliação comparativa entre plataformas de grafos é garantir uma diversidade em relação aos aspectos chave citados na Seção 1 (redes, arquitetura computacional, plataforma e conjunto algorítmico). Graph500,[3] no momento adotado como *benchmark* estado da arte, avalia o desempenho de somente um tipo de rede sintética (grafos de *Kronecker* [Leskovec et al., 2010]) e apenas um algoritmo (busca em largura). Isso reflete a importância dos estudos comparativos recentes entre plataformas, algoritmos, arquiteturas e grafos (ver Tabela 1). Nesse conjunto, destaca-se [Guo et al., 2014], no qual é proposta uma metodologia de avaliação que contempla algoritmos, plataformas e arquitetura computacional que culminou no recente *benchmark LDBC Graphalytics* [Iosup et al., 2016].

Em particular, no presente trabalho avalia-se GraphX [Gonzalez et al., 2014], NetworKit [Staudt et al., 2014] e PowerGraph [Gonzalez et al., 2012]. Esse conjunto único na literatura de plataformas recentes representa uma diversidade de abordagens: (i) uma API para grafos (GraphX) construída sobre um ambiente genérico de processamento de grande volume de dados (Spark); (ii) um *framework* de propósito específico ao processamento de grafos de larga escala (PowerGraph); e (iii) uma biblioteca de desempenho para processamento e análise de redes (NetworKit). Vale notar que apesar de GraphX e PowerGraph serem ferramentas para ambientes de memória distribuída, elas são aptas a fazer uso de multiprocessadores quando executados em ambientes de memória compartilhada. Essas plataformas são brevemente apresentadas na Seção 3.1.

Além do conjunto considerado de plataformas de processamento de grafos, os experimentos apresentados neste trabalho diferenciam-se pelos seguintes aspectos dos demais (ver Tabela 1). É considerada uma maior quantidade de redes, geradas sinteticamente com características distintas, além de traços de redes reais, permitindo melhor comparabilidade entre as plataformas estudadas. Deve-se notar que avaliar modelos de rede com características distintas (Subseção 3.3) contrasta com os demais trabalhos, os quais avaliam redes com características topológicas similares (traços de redes reais seguindo lei de potência na distribuição de grau dos vértices). Também é adotado um conjunto de algoritmos que agrega diversidade por representar análises de grande interesse

[3] http://www.graph500.org/

Tabela 1. Características de trabalhos relacionados.

Estudo	Plataformas	Quantidade de grafos ou redes	Menor e maior redes $\mathcal{O}(n, m)$*	Algoritmos
A [Elser e Montresor, 2013]	2, 5, 14 e 26	7	$(10^4, 10^5)$ e $(10^6, 10^8)$	xiii
B [Jouili e Vansteenberghe, 2013]	10, 18, 21 e 27	1	$(10^6, 10^6)$	xii, xiv, xvi e xx
C [Guo et al., 2014]	2, 4, 7, 14, 18 e 26	7	$(10^5, 10^6)$ e $(10^8, 10^9)$	ii, v, vi, ix e xxi
D [Han et al., 2014]	2, 12, 14 e 17	5	$(10^6, 10^7)$ e $(10^8, 10^9)$	viii, xvii, xviii e xxiii
E [Lu et al., 2014]	2, 13, 14 e 24	6	$(10^7, 10^8)$ e $(10^8, 10^9)$	i, vii, x, xi, xvii, xix e xviii
F [Satish et al., 2014]	2, 8, 9, 11, 14 e 25	9	$(10^6, 10^7)$ e $(10^8, 10^{10})$	ii, iv, xvii e xxii
G [Zhao et al., 2014]	2, 6, 12, e 23	3	$(10^5, 10^6)$ e $(10^7, 10^8)$	xvii, xviii e xxii
H [Koch et al., 2016]	1, 2, 3, 14 e 19	9	$(10^5, 10^6)$ e $(10^7, 10^9)$	iii, v, vi e xvii
I [Iosup et al., 2016]	2, 15, 16, 20, 22 e 23	16	$(10^6, 10^7)$ e $(10^8, 10^9)$	ii, v, xv, xvii, xviii e xxiii
J Este artigo	**16, 19 e 23**	**24**	$(10^4, 10^5)$ **e** $(10^8, 10^9)$	**vi, xvii e xviii**

Levantamento realizado pelos autores.
* n e m identificam a ordem do número de vértices e arestas respectivamente.
Algoritmos: **i**) Bipartite Maximal Matching; **ii**) BFS; **iii**) Cluster. Coefficients; **iv**) Col. Filtering; **v**) Com. Detection; **vi**) Connected Components; **vii**) Diameter Estimation; **viii**) Dist. Minimum Spanning Tree; **ix**) Forest Fire Model; **x**) Graph Coloring; **xi**) HashMin; **xii**) Intensive workload; **xiii**) KCore; **xiv**) Loading workload; **xv**) Local Cluster. Coefficients; **xvi**) Neighborhood Exploration workload; **xvii**) Page Rank; **xviii**) SSSP; **xix**) Shiloach-Vishkins; **xx**) Shortest Path workload; **xxi**) Stats; **xxii**) Triangle Counting; e **xxiii**) Weakly Connected Components
Configuração: **A)** 32 nós AWS EC2 m1.medium, cada um com 2 EC2 e 3.75 GB RAM e 1 nó com 16 CPUs e 10 GB RAM; **B)** Servidor com 1 Intel dualcore 2.5GHz e 8 GB RAM; **C)** 50 nós, cada um com 2 Intel Xeon e 24GB RAM; **D)** 128 nós AWS EC2 m1.xlarge , cada um com 8 Intel Xeon 1.7GHz, 15GB RAM; **E)** 15 nós, cada um com 2 Intel Xeon 2.0GHz e 48 GB RAM; **F)** 64 nós, cada um com 24 Intel Xeon 2.7 GHz e 64 GB RAM; **G)** 8 nós, cada um 2 Intel Xeon e 24 GB RAM; **H)** 8 nós, cada um com 2 Intel Xeon e 24 GB RAM; **I)** 6 *clusters*, cada um com 200 nós contendo 2 Intel Xeon 2.40GHz e 64GB RAM; e **J)** Servidor com 2 Intel Xeon 2.27GHz e 47 GB RAM
Plataformas: **1)** Apache Flink; **2)** Apache Giraph; **3)** Apache Giraph++; **4)** Apache Hadoop; **5)** Apache Hama; **6)** Apache Spark; **7)** Apache Yarn; **8)** Código Nativo; **9)** CombBLAS; **10)** DEX; **11)** Galois; **12)** GPS; **13)** GraphChi; **14)** GraphLab; **15)** GraphMat; **16)** GraphX; **17)** Mizan; **18)** Neo4j; **19)** NetworKit; **20)** OpenG; **21)** OrientDB; **22)** PGX.D; **23)** PowerGraph; **24)** Pregel+; **25)** SociaLite; **26)** Stratosphere; e **27)** Titan

da comunidade (conectividade, centralidade e caminho). Por fim, os experimentos são realizados em um ambiente modesto, o que traz informações quanto à aplicabilidade desse em análises de grafos de larga escala. Essas escolhas são detalhadas na Seção 3.

3. Metodologia

Nesta seção é apresentada a metodologia para a escolha de algoritmos, modelos de rede e plataformas de processamento de grafos para realização dos experimentos.

3.1. Plataformas

Foram selecionadas três plataformas: GraphX [Gonzalez et al., 2014], NetworKit [Staudt et al., 2014] e PowerGraph [Gonzalez et al., 2012] para execução dos experimentos, como ressaltado e justificado na Seção 2. Para melhor compreensão, primeiro são apresentados dois modelos de programação de duas delas.

O modelo de programação centrado em vértice *Pregel* [Malewicz et al., 2010] e suas variações têm sido adotado por vários *frameworks* voltados ao processamento de grafos, por exemplo, GraphX e PowerGraph. Baseando-se no modelo *Bulk Synchronous Parallel* (BSP), o processamento é dividido em iterações (*Supersteps*).

Cada vértice (análogo a um processo no BSP) executa uma função *compute* definida pelo usuário, além de enviar e receber mensagens de seus vizinhos. Dentro de um *Superstep*, a função *compute* é executada em paralelo. As mensagens enviadas em uma iteração, só estarão disponíveis ao destinatário após a barreira de sincronização entre todos os vértices antes do início da próxima iteração. Os algoritmos terminam quando ou não houver mais troca de mensagens ou um critério de convergência global for atingido.

O modelo *GAS* não se baseia em troca de mensagens, mas em um abstração de memória compartilhada na qual os vértices se comunicam com seus vizinhos. O modelo provê três funções definidas pelos usuário: (i) *Gather*: vértices coletam informação sobre seus vizinhos através de uma soma generalizada Σ (associativa e comutativa); (ii) *Apply*: vértices atualizam seu valor baseando-se em Σ; e (iii) *Scatter*: vértices atualizam as arestas adjacentes com seus novos valores. No modo síncrono, há a realização do ciclo *Gather-Apply-Scatter (GAS)* de modo paralelo em cada iteração de um BSP.

Com isso, apresenta-se brevemente as plataformas consideradas neste estudo:

- **GraphX** é uma API para processamento de grafos disponível no Apache Spark [Zaharia et al., 2012] que provê ao desenvolvedor dois modelos de programação: *GAS* e uma variante otimizada do *Pregel* que esse trabalho utiliza em seus algoritmos. O diferencial dessa plataforma é a possibilidade de análise conjunta entre grafos e outros tipos de representação de dados [Gonzalez et al., 2014]. Entretanto, o foco deste trabalho é somente no processamento de grafos.
- **NetworKit** é uma biblioteca destinada à análise de redes que implementa de forma eficiente algoritmos para grafos, sendo muito deles implementados de forma paralela [Koch et al., 2016]. Entretanto, no presente experimento são utilizadas apenas versões seriais para uma comparação de base. Além disso, o NetworKit provê componentes para análise de redes, geradores de grafos sintéticos e ferramentas de visualização.
- **PowerGraph** é uma plataforma de processamento de grafos que explora o modelo de computação *GAS*. O diferencial da plataforma consiste em como se divide a computação entre os processos. Em síntese, ao se distribuir arestas uniformemente pelo recurso computacional, obtém-se um maior equilíbrio de carga em grafos naturais, como aqueles cuja distribuição de grau obedece a uma lei de potência. Além disso, a plataforma é escrita em C++ a fim de se obter melhor desempenho computacional.

3.2. Algoritmos

Para se ter representantes de análises distintas, foram escolhidos os algoritmos *Connected Components* (CC), *Page Rank* (PR) e *Single Source Shortest Paths* (SSSP) destinados às avaliações de conectividade, centralidade e caminho, respectivamente. Como na maioria dos trabalhos relacionados, foram selecionados os algoritmos disponibilizados nos próprios *toolkits* das plataformas. Além disso, para cada experimento, as plataformas utilizaram os mesmos parâmetros de execução para os algoritmos, por exemplo, vértice de origem e número máximo de iterações.

Considere um grafo $G = (V, E)$, onde V é o conjunto de vértices e E o de arestas, assim como subgrafos de G denotados por G_i. No mais, $n = |V|$ e $m = |E|$. Assim, tem-se como algoritmos:

- **CC**: Deve-se encontrar a menor quantidade i de G_i de modo que exista um caminho entre quaisquer dois vértices $u, v \in G_i$. GraphX e PowerGraph utilizam algoritmos baseados na propagação de rótulo: na primeira iteração se atribui a cada vértice um rótulo distinto $r \in \mathbb{Z} \mid r \in \{1, ..., |V|\}$, propagando-o para seus vizinhos. Nas iterações posteriores, cada vértice se recebeu rótulos da iteração anterior avalia se o menor rótulo é inferior ao seu. Caso sim, atualiza a informação e propaga o rótulo. Quando a propagação acabar na rede, o algoritmo termina. Em contraste, NetworKit executa sucessivas buscas em largura (BFS) até que todos os vértices tenham sido

alcançadas, propagando através da BFS o índice de seu nó de origem. Ambos algoritmos têm desempenho no caso médio de $\mathcal{O}(n+m)$.

- **PR**: É uma métrica de centralidade projetada para avaliar a relevância de páginas (vértices) na *Web* que popularizou-se devido ao seu uso no buscador do google. Baseia-se nos conceitos de que uma página é mais importante se as páginas que a referenciam também o são e, de que um usuário em uma página pode seguir seus *links* (arestas) ou aleatoriamente ir para outra página qualquer *damping factor* df) [Page et al., 1999]. As plataformas implementam uma aproximação clássica do PR através da iteração de potência: Na primeira iteração, se atribui a cada vértice $v \in V$ o *Page Rank* $pr = 1.0$. Nas iterações i posteriores, $\forall v \in V$ se atualiza $pr = (1-df) + (df) * \sum msg((u,v)) \in E$ e propaga $pr/deg^+(v)$ pela arestas $(v,u) \in E$.[4] A convergência do algoritmo ocorre quando $|pr_{i-1} - pr_i| < \epsilon$. Todas as plataformas experimentadas utilizam $\epsilon = 10^{-7}$ e $df = 0.85$. Considerando que o número de iterações para convergência seja constante, o desempenho do algoritmo é de $\mathcal{O}(n+m)$.
- **SSSP**: Dado um vértice $v \in V$ qualquer, deve-se encontrar um dos caminhos mínimos de v a $u \in V$. As plataformas GraphX e PowerGraph utilizam algoritmos similares ao de *Bellman-Ford* [Cormen et al., 2009]. Já o NetworKit usa o algoritmo de *Dijkstra* [Cormen et al., 2009]. Como nas redes utilizadas as arestas possuem o mesmo peso, ambos algoritmos têm no caso médio complexidade de $\mathcal{O}(n+m)$.

3.3. Redes

Foram utilizadas 23 redes conexas nos experimentos (ver Tabela 2): 19 redes sintéticas (9 *Barabási Albert (BA)* [Barabási e Albert, 1999], 9 *Erdös-Rényi (ER)* [Erdős e Rényi, 1959] e 1 *Random Geometric Graph* [Penrose, 2003]) e 4 traços de redes reais. As redes BA e ER foram geradas através da biblioteca SNAP.[5] Os demais grafos foram obtidos de dois repositórios (a sigla RE os designará no decorrer do texto)[5,6]

Redes sintéticas seguindo os modelos BA e ER são maioria no conjunto. Sua adoção se baseou em dois aspectos: maior controlabilidade do experimento devido à escolha das complexidades das redes; e a distinção nas distribuições dos graus dos vértices, lei de potência (BA) e de Poisson (ER). Mais ainda, o diâmetro é menor em redes BA, enquanto o *clustering* médio é menor em redes ER. Essas características influenciam diferentemente no desempenho das plataformas. Diante disso, para o modelo BA, para cada ordem foram geradas três redes, sendo variada a quantidade de vértices $k = 10^p$, onde $p = \{1,2,3\}$, a que um nó entrante se conectará. No que tange ao modelo ER, para cada ordem n foram produzidas três redes de modo a se ter $m = 10 \times n^p$. Vale notar que quanto mais altos k e p, mais densa se torna a rede, como mostrado na Tabela 2.

De mesmo modo, foram selecionados 4 traços de redes reais (COMLJ,[5] COMORKUT,[5] SOCSINAWEIBO,[6] e SOCTWITTER2010[6]) e 1 grafo sintético (RGGN224S0[6]). COMLJ é um traço da rede da comunidade de *blogging* LiveJournal. COMORKUT é um traço da rede social Orkut. SOCSINAWEIBO é um traço da rede chinesa de *microblogging* Sina Weibo. SOCTWITTER2010 é um traço da rede social Twitter em 2010. RGGN224S0 é um grafo aleatório geométrico no qual cada vértice é um ponto de um quadrado unitário onde as arestas conectam vértices cuja distância Euclidiana é inferior a $0.55\frac{\ln n}{n}$. Deve-se notar que os traços de redes sociais apresentam

[4] $msg(x)$: valor da mensagem vinda pela aresta x e $deg^+(v)$: quantidade arestas com origem em v.
[5] https://snap.stanford.edu/ [6] http://networkrepository.com/

Tabela 2. Descrição das redes utilizadas.

Nome da Rede	Ordem	Tamanho	Diâmetro aproximado	Grau Médio	Clustering Médio Aproximado	Tamanho em Disco
BA-010K-100K	$1.00 \cdot 10^4$	$9.99 \cdot 10^4$	4	20	0.00345	914 KB
BA-010K-001M	$1.00 \cdot 10^4$	$9.95 \cdot 10^5$	3	199	0.02250	9 MB
BA-010K-010M	$1.00 \cdot 10^4$	$9.50 \cdot 10^6$	2	1,900	0.13336	90 MB
BA-100K-001M	$1.00 \cdot 10^5$	$1.00 \cdot 10^6$	5	20	0.00052	11 MB
BA-100K-010M	$1.00 \cdot 10^5$	$9.99 \cdot 10^6$	3	200	0.00372	111 MB
BA-100K-100M	$1.00 \cdot 10^5$	$9.95 \cdot 10^7$	2	1,990	0.02267	1 GB
BA-001M-010M	$1.00 \cdot 10^6$	$1.00 \cdot 10^7$	6	20	0.00007	131 MB
BA-001M-100M	$1.00 \cdot 10^6$	$1.00 \cdot 10^8$	4	200	0.00057	1 GB
BA-001M-001B	$1.00 \cdot 10^6$	$9.99 \cdot 10^8$	(*)	1,999	(*)	13 GB
ER-010K-100K	$1.00 \cdot 10^4$	$1.00 \cdot 10^5$	5	20	0.00067	978 KB
ER-010K-001M	$1.00 \cdot 10^4$	$1.00 \cdot 10^6$	3	200	0.00676	10 MB
ER-010K-010M	$1.00 \cdot 10^4$	$1.00 \cdot 10^7$	2	2,000	0.07694	98 MB
ER-100K-001M	$1.00 \cdot 10^5$	$1.00 \cdot 10^6$	6	20	0.00007	12 MB
ER-100K-010M	$1.00 \cdot 10^5$	$1.00 \cdot 10^7$	3	200	0.00067	118 MB
ER-100K-100M	$1.00 \cdot 10^5$	$1.00 \cdot 10^8$	2	200	0.00676	1 GB
ER-001M-010M	$1.00 \cdot 10^6$	$1.00 \cdot 10^7$	7	20	0.00001	138 MB
ER-001M-100M	$1.00 \cdot 10^6$	$1.00 \cdot 10^8$	4	200	0.00007	1 GB
ER-001M-001B	$1.00 \cdot 10^6$	$1.00 \cdot 10^9$	(*)	2,000	(*)	14 GB
COMLJ	$4.00 \cdot 10^6$	$3.47 \cdot 10^7$	17	17	0.28430	479 MB
COMORKUT	$3.07 \cdot 10^6$	$1.17 \cdot 10^8$	9	76	0.16660	2 GB
RGGN224S0	$1.68 \cdot 10^7$	$1.33 \cdot 10^8$	3,025	16	0.58669	2 GB
SOCSINAWEIBO	$5.87 \cdot 10^7$	$2.61 \cdot 10^8$	7	9	0.02270	4 GB
SOCTWITTER2010	$2.13 \cdot 10^7$	$2.65 \cdot 10^8$	18	25	0.12680	4 GB

(*) A computação desses valores foi abortada após 120 horas de execução.

distribuição de grau seguindo uma lei de potência.

4. Resultados experimentais

Nesta seção, são apresentados e discutidos os resultados dos experimentos para as plataformas, algoritmos e redes apresentados na Seção 3. Os experimentos foram executados em um servidor Ubuntu 14.10 com 2 Intel® Xeon™ CPU E5520 @ 2.27GHz com 4 núcleos físicos cada, 47GB DDR3 DIMM e Seagate Desktop HDD ST1000DM003 1TB 64MB Cache SATA 6.0Gb/s 3.5. Foram utilizadas as versões das plataformas GraphLab PowerGraph 2.2, GraphX incluído em Spark 1.6.1 e NetworKit 4.0. Utilizou-se 8 núcleos de processamento para GraphX e PowerGraph e 1 núcleo para NetworKit.

Os resultados refletem a média do tempo total (*makespan*) de cinco execuções para cada combinação de plataforma, algoritmo e rede, sendo estabelecidos intervalos de confiança de 95% para as médias. Assim, ao comparar as alternativas pertinentes a cada experimento, se houve sobreposição dos intervalos de no mínimo de duas delas (sendo elas as mais eficientes), considerou-se que não existiu diferenciação estatística entre tais (SDE). Além disso, estabeleceu-se o tempo máximo de execução de cada experimento em 14 horas. Com isso, o conjunto completo de experimentos só pôde ser realizado para plataforma NetworKit. As plataformas GraphX e PowerGraph em alguns experimentos requisitaram mais de cinco vezes o tamanho do grafo em disco para alocar as estruturas de dados necessárias em memória, acarretando no uso de *swap* em disco, ultrapassando o tempo estabelecido, ou até mesmo na falha do experimento. Essa situação é também relatada em trabalhos relacionados [Iosup et al., 2016, Zhao et al., 2014].

Os resultados dos experimentos são analisados sob duas perspectivas complementares: (i) para cada algoritmo, qual a melhor plataforma (Seção 4.1); e (ii) para cada

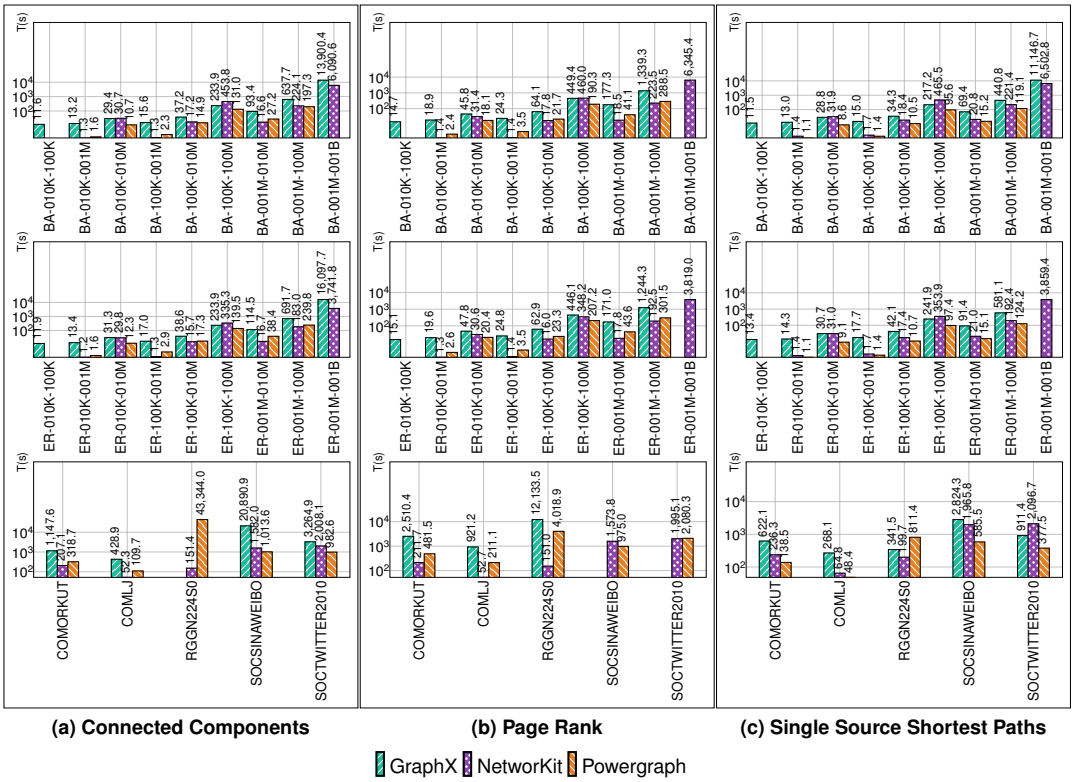


Figura 2. Comparativo do tempo de execução $T(s)$ entre as plataformas.

plataforma, em qual algoritmo esta obteve melhor desempenho (Seção 4.2). Mais ainda, nas duas perspectivas analisa-se o desempenho entre os diferentes modelos de rede.

4.1. Desempenho dos algoritmos para cada plataforma

Em relação ao algoritmo **CC**, o NetworKit obteve o melhor desempenho geral, sendo seguido pelo PowerGraph. Nota-se que a diferença de desempenho entre as duas plataformas foi mais significativa para o modelo ER. Isso se deve ao diâmetro maior desse tipo de rede, ou seja, há mais iterações para PowerGraph, enquanto, há a execução de apenas uma BFS para NetworKit, pois as redes têm só um componente conexo (Figura 2a). Já o algoritmo **Page Rank** exigiu maior quantidade de iterações do que os demais algoritmos, assim o *overhead* de sincronização nas plataformas centradas em vértice tornou-se maior, propiciando ao NetworKit também ser o mais eficiente (Figuras 2b e 4a). Por sua vez, considerando o algoritmo **SSSP**, sua menor quantidade de iterações em relação aos seus pares possibilitou ao PowerGraph ser mais eficiente (Figuras 2c e 4a). De modo geral, o diâmetro das redes teve maior impacto em **CC** e **Page Rank**, por exemplo, o desempenho para o grafo RGGN224S0 (Figuras 2a e 2b).

4.2. Desempenho das plataformas para cada algoritmo

As execuções dos algoritmos para **GraphX** são estatisticamente distintas entre si, isto é, apesar dos algoritmos terem ordem de complexidade computacional similar, a plataforma depende das especifidades dos mesmos (Figura 4b). A plataforma foi mais

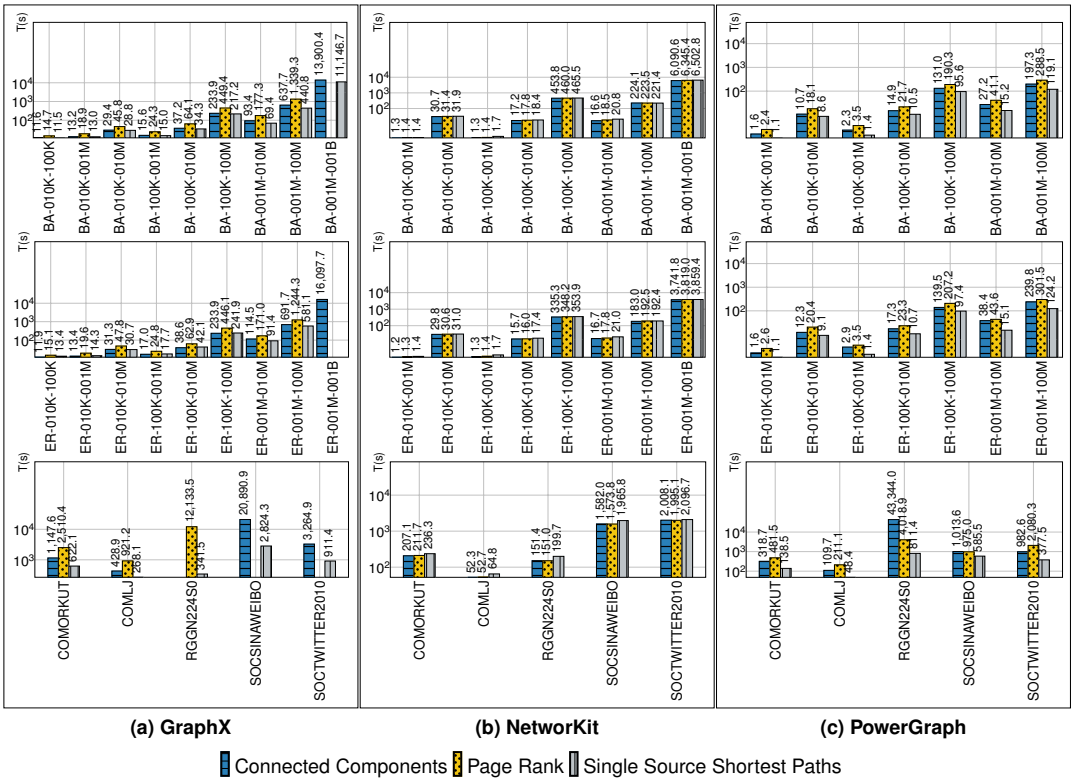
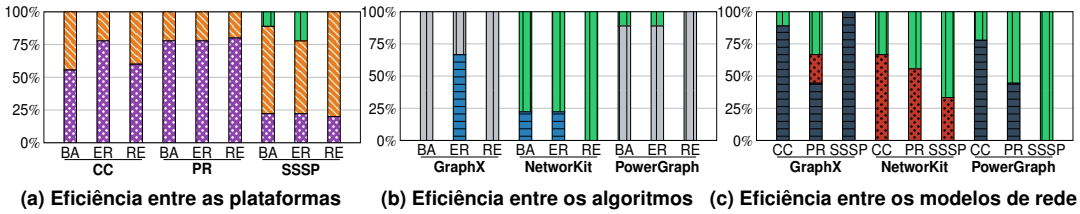


Figura 3. Comparativo do tempo de execução $T(s)$ entre os algoritmos.

eficiente na maioria dos experimentos para o algoritmo SSSP (Figura 3a). Além disso, o **GraphX** foi melhor em redes BA do que ER, o que se deve ao menor tamanho e diâmetro de algumas redes BA.

Ao comparar a execução dos algoritmos (Figura 4b), **NetworKit** foi a plataforma mais homogênea nos experimentos. Assim, algoritmos de ordem similar tendem a ter desempenho estatisticamente igual para a biblioteca. Ao comparar os modelos de rede (Figura 4c), a plataforma obteve melhor desempenho no modelo ER, apesar da que todas redes ER possuem a mesma ou maior quantidade de arestas que seus pares BA. Por exemplo, a rede ER de tamanho 10^9 tem cerca de 5.0×10^5 mais arestas que a rede BA de mesma ordem (Figura 3b). A execução serial da plataforma descarta problemas de balanço de carga relacionados à topologia da rede. Foram examinadas as aplicações dos algoritmos através da ferramenta gprof [Graham et al., 1982], concluindo-se que a rotina mais demandante é NetworKit::Graph::indexInOutEdgeArray, a qual exige mais tempo nas redes BA. A rotina tem por intuito dados dois nós u e v, retornar o índice de u no vetor de vizinhança de v, sendo o tamanho de tal vetor proporcional ao grau de v. Desse modo, conclui-se que a estrutura de *hubs* (nós com muitos vizinhos) em uma rede BA influencia significativamente o tempo de execução de **NetworKit**.

PowerGraph obteve melhor desempenho invariavelmente para o algoritmo *SSSP*, fato que concerne a menor quantidade de iterações observada (Figura 4b). Além disso, a plataforma foi mais eficiente no modelo BA em virtude do menor tamanho e diâmetro de



(a) Indica a fração dos casos em que uma plataforma foi a melhor para cada algoritmo e conjunto de rede. Por exemplo, NetworKit foi a plataforma mais eficiente em 75% ou mais das redes BA, ER e RE para o algoritmo *Page Rank*. **(b)** Avalia, para cada tipo de rede e plataforma, a fração de casos em que cada algoritmo teve execução mais eficiente. Por exemplo, *SSSP* foi o algoritmo mais rápido (75% ou mais) nos experimentos da plataforma PowerGraph. **(c)** Avalia, para cada tipo de algoritmo e plataforma, a fração de casos em que o modelo de rede (BA ou ER) teve execução mais rápida (SDE indica sem diferença estatística). Por exemplo, ao executar o algoritmo *Connected Components*, GraphX foi mais eficiente no modelo BA (mais de 75%).

Figura 4. Síntese dos resultados.

algumas dessas redes. Entretanto, quanto menor a quantidade de iterações dos algoritmos, mais o *framework* tendeu a desempenho similar, e.g. *SSSP* (Figura 3c).

Para grafos de repositórios, o maior diâmetro de RGGN224S0 elevou substancialmente o tempo de execução de **GraphX** e **PowerGraph**, mais ainda para o algoritmo CC que demonstrou maior duração de cada *superstep*. Além disso, **NetworKit** foi menos eficiente em SOCSINAWEIBO e SOCTWITTER2010, refletindo a influência da estrutura de *hubs* nas redes.

5. Conclusões e trabalhos futuros

A conhecimento dos autores, este é o primeiro estudo, realizado em um ambiente de memória compartilhada desse porte, avaliativo experimental de desempenho de três plataformas no processamento de análises relevantes em um grupo diversificado de grafos. Ao analisar GraphX, NetworKit e PowerGraph, considerou-se três modelos de programação e arquiteturas de sistema diferentes. Desse modo, avaliou-se experimentalmente o desempenho dessas plataformas através dos tempos de execução de *Connected Components*, *Page Rank* e *SSSP* em um *dataset* composto por um grafo aleatório, redes sintéticas *Barabási-Albert* e *Erdös-Rényi*, além de traços de redes reais.

Ao conduzir os experimentos em um servidor de tamanho modesto, com 47 GB de memória e 8 núcleos de processamento, foi demonstrada a aplicabilidade de um ambiente computacional desse porte na execução de análises de grafos de larga escala, sendo esse apto a processar redes de um bilhão de arestas em menos de 2 horas.

O NetworKit, apesar de sua execução serial, obteve o melhor desempenho no cenário geral (mais eficiente em $\approx 54\%$ dos casos). Ser o mais eficiente em 100% dos experimentos para PR e o único sistema apto a executar os experimentos para o conjunto completo de redes impactou nesse resultado. Entretanto, o PowerGraph (mais eficiente em $\approx 42\%$ dos casos) foi mais eficiente nas redes mais densas que seguiam leis de potência para os algoritmos com menor quantidade de iterações (CC e SSSP). Apesar da diferença em tempos de execução entre as plataformas GraphX e PowerGraph nos experimentos apresentados ser menos expressiva do que em [Iosup et al., 2016], o presente

trabalho também demonstra a menor eficiência do GraphX. Desse modo, os resultados experimentais apresentados podem auxiliar aos interessados em suas tarefas de melhor selecionar as plataformas para o processamento de grafos de acordo com as caraterísticas de suas redes e análises pretendidas.

Pretende-se em trabalhos futuros: (i) aumentar a abrangência do experimento ao incluir mais plataformas da taxonomia apresentada; (ii) realizar experimentos em ambiente de memória distribuída com redes de maior escala; (iii) avaliar métricas além de tempo de execução, como escalabilidade horizontal e vertical, consumo de recursos computacionais (disco, memória e rede) e *overhead* de plataforma; (iv) estabelecer um conjunto de algoritmos mais diversificado em relação as categorias e tipos de análises adotados neste estudo, cobrindo assim mais métricas de interesse.

Agradecimentos

Os autores agradecem ao CNPq e FINEP por financiarem em parte este trabalho.

Referências

Barabási, A. e Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(5439):509–512.

Chakrabarti, D. e Faloutsos, C. (2006). Graph mining: Laws, generators, and algorithms. *ACM Computing Surveys*, 38(1):2.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., e Stein, C. (2009). *Introduction to algorithms*. MIT Press.

Doekemeijer, N. e Varbanescu, A. (2014). A survey of parallel graph processing frameworks. Technical report, Delft University of Technology.

Elser, B. e Montresor, A. (2013). An evaluation study of big data frameworks for graph processing. In *Proc. IEEE Int. Conf. on Big Data*, pages 60–67.

Erdős, P. e Rényi, A. (1959). On random graphs i. *Publ. Mathematicae Debrecen*, 6:290–297.

Gonzalez, J., Low, Y., Gu, H., Bickson, D., e Guestrin, C. (2012). PowerGraph: Distributed graph-parallel computation on natural graphs. In *Proc. USENIX Symp. on Operating Systems Design and Implementation (OSDI)*, pages 17–30.

Gonzalez, J., Xin, R., Dave, A., Crankshaw, D., Franklin, M., e Stoica, I. (2014). GraphX: Graph processing in a distributed dataflow framework. In *Proc. USENIX Symp. on Operating Systems Design and Implementation (OSDI)*, pages 599–613.

Graham, S., Kessler, P., e Mckusick, M. (1982). Gprof: A call graph execution profiler. In *Proc. ACM SIGPLAN Notices*, pages 120–126.

Guo, Y., Biczak, M., Varbanescu, A., Iosup, A., Martella, C., e Willke, T. (2014). How well do graph-processing platforms perform? an empirical performance evaluation and analysis. In *Proc. IEEE Int. Parallel and Distributed Processing Symp. (IPDS)*, pages 395–404.

Han, M., Daudjee, K., Ammar, K., Özsu, M., Wang, X., e Jin, T. (2014). An experimental comparison of pregel-like graph processing systems. In *Proc. VLDB Endowment*, pages 1047–1058.

Iosup, A., Hegeman, T., Ngai, W., Heldens, S., Pérez, A., Manhardt, T., Chafi, H., Capota, M., Sundaram, N., Anderson, M., et al. (2016). LDBC graphalytics: A benchmark for large-scale graph analysis on parallel and distributed platforms. Technical report, Delft University of Technology.

Jouili, S. e Vansteenberghe, V. (2013). An empirical comparison of graph databases. In *Proc. ASE/IEEE Int. Conf. on Social Computing (SocialCom)*, pages 708–715.

Koch, J., Staudt, C., Vogel, M., e Meyerhenke, H. (2016). An empirical comparison of big graph frameworks in the context of network analysis. *ArXiv e-prints*, abs/1601.00289.

Leskovec, J., Chakrabarti, D., Kleinberg, J., Faloutsos, C., e Ghahramani, Z. (2010). Kronecker graphs: An approach to modeling networks. *Journal of Machine Learning Research*, 11:985–1042.

Lu, Y., Cheng, J., Yan, D., e Wu, H. (2014). Large-scale distributed graph computing systems: An experimental evaluation. In *Proc. VLDB Endowment*, pages 281–292.

Lumsdaine, A., Gregor, D., Hendrickson, B., e Berry, J. (2007). Challenges in parallel graph processing. *Parallel Processing Letters*, 17(1):5–20.

Malewicz, G., Austern, M., e Bik, A. (2010). Pregel: A system for large-scale graph processing. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 135–145.

Newman, M. (2010). *Networks: An introduction*. Oxford University Press.

Page, L., Brin, S., Motwani, R., e Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Penrose, M. (2003). *Random geometric graphs*. Oxford University Press.

Pires, R., Rêgo, L., Macêdo, J., e Vidal, V. (2015). Processamento de grafos em big data. In Hara, C. S., Porto, F., e Ogasawawara, E., editors, *Tópicos em Gerenciamento de Dados e Informações – SBBD 2015*, chapter 1, pages 8–38. SBC.

Satish, N., Sundaram, N., Patwary, M., Seo, J., Park, J., Hassaan, M., Sengupta, S., Yin, Z., e Dubey, P. (2014). Navigating the maze of graph analytics frameworks using massive graph datasets. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 979–990.

Silva, T. e Zhao, L. (2016). *Machine learning in complex networks*. Springer.

Staudt, C., Sazonovs, A., e Meyerhenke, H. (2014). NetworKit: An interactive tool suite for high-performance network analysis. *ArXiv e-prints*, abs/1403.3005.

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M. J., Shenker, S., e Stoica, I. (2012). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proc. USENIX Conf. on Networked Systems Design and Implementation (NSDI)*, pages 15–28.

Zhao, Y., Yoshigoe, K., Xie, M., Zhou, S., Seker, R., e Bian, J. (2014). Evaluation and analysis of distributed graph-parallel processing frameworks. *Journal of Cyber Security and Mobility*, 3(3):289–316.