GRADES-NDA 2019

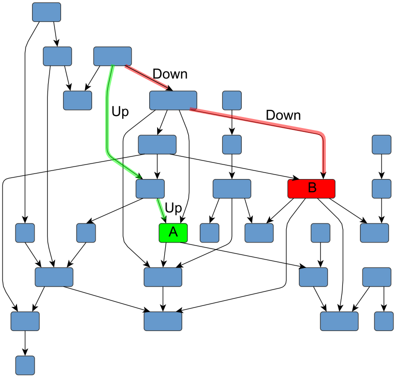
Evaluation of the Context-Free Path Querying Algorithm Based on Matrix Multiplication

Nikita Mishin, Iaroslav Sokolov, Egor Spirin, Vladimir Kutuev, Egor Nemchinov, Sergey Gorbatyuk, **Semyon Grigorev**

JetBrains Research, Programming Languages and Tools Lab
Saint Petersburg University

June 30, 2019

Context-Free Path Querying



Navigation through a graph

- Are nodes A and B on the same level of hierarchy?
- Is there a path of form **Up**^{n} **Down**^{n}?
- Find all paths of form **Up**^{n} **Down**^{n} which start from the node A

Context-Free Path Querying: Relational Query Semantics

- $\mathbb{G} = (\Sigma, N, P)$ — context-free grammar in normal form
  - $A \to BC$, where $A, B, C \in N$
  - $A \to x$, where $A \in N, x \in \Sigma$
  - $L(\mathbb{G}, A) = \{\omega \mid A \to^* \omega\}$
- $G = (V, E, L)$ — directed graph
  - $v \xrightarrow{l} u \in E$
  - $L \subseteq \Sigma$
- $\omega(\pi) = \omega(v_0 \xrightarrow{l_0} v_1 \xrightarrow{l_1} \cdots \xrightarrow{l_{n-2}} v_{n-1} \xrightarrow{l_{n-1}} v_n) = l_0 l_1 \cdots l_{n-1}$
- $R_A = \{(n, m) \mid \exists n\pi m, \text{ such that } \omega(\pi) \in L(\mathbb{G}, A)\}$

Matrix-based algorithm

Algorithm Context-free recognizer for graphs

function $\textsc{contextFreePathQuerying}$(D, G)

  $n \leftarrow$ the number of nodes in D

  $E \leftarrow$ the directed edge-relation from D

  $P \leftarrow$ the set of production rules in G

  $T \leftarrow$ the matrix $n \times n$ in which each element is $\varnothing$

  for all $(i, x, j) \in E$ **do** ▷ Matrix initialization

    $T_{i,j} \leftarrow T_{i,j} \cup \{A \mid (A \rightarrow x) \in P\}$

  while matrix T is changing **do**

    $T \leftarrow T \cup (T \times T)$ ▷ Transitive closure T^{cf} calculation

  return T

Questions for Evaluation

- Can GPGPUs utilization for CFPQ improve performance in comparison with the CPU version?

Questions for Evaluation

- Can GPGPUs utilization for CFPQ improve performance in comparison with the CPU version?
- Is it possible to achieve higher performance by means of existing libraries for matrices operations or do we need to create our own solution to get more control?

Questions for Evaluation

- Can GPGPUs utilization for CFPQ improve performance in comparison with the CPU version?
- Is it possible to achieve higher performance by means of existing libraries for matrices operations or do we need to create our own solution to get more control?
- Can we achieve a high-performance solution with high-level languages such as Python?

Questions for Evaluation

- Can GPGPUs utilization for CFPQ improve performance in comparison with the CPU version?
- Is it possible to achieve higher performance by means of existing libraries for matrices operations or do we need to create our own solution to get more control?
- Can we achieve a high-performance solution with high-level languages such as Python?
- Can we gain more performance improvement by using sparse matrix representation for CFPQ?

CPU-based implementations

[Scipy] Sparse matrices multiplication by using **Scipy** in **Python** programming language

CPU-based implementations

[Scipy] Sparse matrices multiplication by using **Scipy** in **Python** programming language

[M4RI] Dense matrices multiplication by using **m4ri2** library which implements the Method of Four Russians in **C** language

GPGPU-based implementations

[GPU4R] Our own implementation of the Method of Four Russians in **CUDA C**

GPGPU-based implementations

[GPU4R] Our own implementation of the Method of Four Russians in **CUDA C**

[GPU_N] Our own implementation of the naïve boolean matrix multiplication in **CUDA C**

GPGPU-based implementations

[GPU4R] Our own implementation of the Method of Four Russians in **CUDA C**

[GPU_N] Our own implementation of the naïve boolean matrix multiplication in **CUDA C**

[GPU_Py] Manual implementation of naïve boolean matrix multiplication in **Python** by using **numba** compiler

Dataset

[RDF]

- The set of the real-world RDF files (ontologies)
- Queries:
  $G_4 : s \to SCOR\ s\ SCO \mid TR\ s\ T \mid SCOR\ SCO \mid TR\ T$,
  $G_5 : s \to SCOR\ s\ SCO \mid SCO$

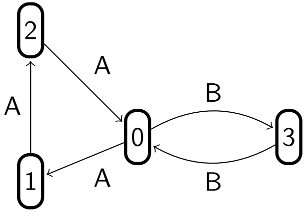
Dataset

[RDF]

- The set of the real-world RDF files (ontologies)
- Queries:
  $G_4 : s \to SCOR\ s\ SCO \mid TR\ s\ T \mid SCOR\ SCO \mid TR\ T$,
  $G_5 : s \to SCOR\ s\ SCO \mid SCO$

[Worst]

- The input graph is a two cycles of coprime lengths with a single common vertex



- Queries: $G_1 : s \to A\ s\ B \mid A\ B$

Dataset

[Full]

- The input graph is sparse, but the result is a full graph
- Queries:
  $G_2 : s \to s\ s \mid A$,
  $G_3 : s \to s\ s\ s \mid A$

Dataset

[Full]

- The input graph is sparse, but the result is a full graph
- Queries:
  $G_2 : s \to s\ s \mid A$,
  $G_3 : s \to s\ s\ s \mid A$

[Sparse]

- Sparse graphs are generated by the GTgraph
- Queries: $G_1 : s \to A\ s\ B \mid A\ B$

Evaluation

- OS: Ubuntu 18.04
- CPU: Intel core i7 8700k 3,7HGz
- RAM: DDR4 32 Gb
- GPGPU: Geforce 1080Ti(11Gb RAM)

Evaluation: worst case

#V	Scipy	M4RI	GPU4R	GPU_N	GPU_Py	CuSprs
16	0.032	< 1	0.008	0.002	0.027	0.309
32	0.118	0.001	0.034	0.008	0.136	0.441
64	0.476	0.041	0.133	0.032	0.524	0.988
128	2.194	0.226	0.562	0.129	2.751	3.470
256	15.299	1.994	3.088	0.544	11.883	15.317
512	121.287	23.204	13.685	2.499	43.563	102.269
1024	1593.284	528.521	88.064	19.357	217.326	1122.055
2048	-	-	-	325.174	-	-

Evaluation: sparse

Graph	Scipy	M4RI	GPU4R	GPU_N	GPU_Py	CuSprs
G5k-0.001	10.352	0.647	0.113	0.041	0.216	5.729
G10k-0.001	37.286	2.395	0.435	0.215	1.331	35.937
G10k-0.01	97.607	1.455	0.273	0.138	0.763	47.525
G10k-0.1	601.182	1.050	0.223	0.114	0.859	395.393
G20k-0.001	150.774	11.025	1.842	1.274	6.180	-
G40k-0.001	-	97.841	11.663	8.393	37.821	-
G80k-0.001	-	1142.959	88.366	65.886	-	-

Example 2: Pseudoknot

RDF			Query G_4					
Name	#V	#E	Scipy	M4RI	GPU4R	GPU_N	GPU_Py	CuSprs
atm-prim	291	685	3	2	2	1	5	269
biomed	341	711	3	5	2	1	5	283
foaf	256	815	2	9	2	< 1	5	270
funding	778	1480	4	7	4	1	5	279
generations	129	351	3	3	2	< 1	5	273
people_pets	337	834	3	3	3	1	7	284
pizza	671	2604	6	8	3	1	6	292
skos	144	323	2	4	2	< 1	5	273
travel	131	397	3	5	2	< 1	6	268
unv-bnch	179	413	2	4	2	< 1	5	266
wine	733	2450	7	6	4	1	7	294

RDF			Query G_5					
Name	#V	#E	Scipy	M4RI	GPU4R	GPU_N	GPU_Py	CuSprs
atm-prim	291	685	1	< 1	1	< 1	2	267
biomed	341	711	4	< 1	1	< 1	5	280
foaf	256	815	1	< 1	1	< 1	2	263
funding	778	1480	2	< 1	3	< 1	4	274
generations	129	351	1	< 1	1	< 1	2	263
people_pets	337	834	1	< 1	1	< 1	3	277
pizza	671	2604	2	< 1	2	< 1	5	278
skos	144	323	< 1	< 1	1	< 1	2	265
travel	131	397	1	< 1	1	< 1	3	271
unv-bnch	179	413	1	< 1	1	< 1	3	266
wine	733	2450	1	< 1	3	< 1	3	281

Evaluation: Full

#V	Query G_2								
	Scipy	M4RI	GPU4R	GPU_N	GPU_Py	CuSprs	Scipy	M4RI	GPU4
100	0.007	0.002	0.002	< 1	0.003	0.278	0.023	0.076	0.00
200	0.040	0.003	0.002	0.001	0.004	0.279	0.105	0.098	0.00
500	0.480	0.003	0.003	0.001	0.004	0.329	1.636	0.094	0.00
1000	3.741	0.007	0.005	0.001	0.006	0.571	13.071	0.106	0.00
2000	40.309	0.063	0.019	0.003	0.017	1.949	93.676	0.108	0.03
5000	651.343	0.366	0.125	0.038	0.150	99.651	1205.421	0.851	0.19
10000	-	1.932	0.552	0.315	0.840	1029.042	-	4.690	1.05
25000	-	33.236	7.252	5.314	15.521	-	-	70.823	15.24
50000	-	360.035	58.751	44.611	129.641	-	-	775.765	130.2
80000	-	1292.817	256.579	190.343	641.260	-	-	-	531.6

Conclusion

- Dataset is published
- Impl!!!
- RDF analysis

Future research

- Detailed investigation of implemented algorithms
- Create open extensible platform for CFPQ algorithms comparison
- Evaluate other CFPQ algorithms
  - !!!
  - !!!
- Add new data and queries
- Integrate with LDBC

Contact Information

- Semyon Grigorev:
  - s.v.grigoriev@spbu.ru
  - Semen.Grigorev@jetbrains.com
- Nikita Mishin: mishinnikitam@gmail.com
- Iaroslav Sokolov: sokolov.yas@gmail.com
- Egor Spirin: egor@spirin.tech
- Vladimir Kutuev: vladimir.kutuev@gmail.com
- Egor Nemchinov: nemchegor@gmail.com
- Sergey Gorbatyuk: sergeygorbatyuk171@gmail.com

- Dataset and algorithm implementations: https://github.com/SokolovYaroslav/CFPQ-on-GPGPU

Thanks!