Implementation and experimental study of GLL algorithm with Neo4j graph database

Vlada Pogozhelskaya

JetBrains Research, Programming Languages and Tools Lab
St Petersburg State University

December 17, 2021

Introduction

- Graph data model
  - Basic entities — graph vertices
  - Relationships between entities are stored in the graph model itself
- Graph databases
  - The most popular is Neo4j
  - Only regular queries are partially supported
- Context-free constraints
  - Strictly more expressive than the regular one
  - Widely used in bioinformatics, RDF file analysis, static code analysis

Context-free path querying problems

All-paths CFPQ problem and reachability CFPQ problem

Let be:

- *Context-free grammar* $\mathbb{G} = \langle N, \Sigma, P, S \rangle$
- *Directed graph* $\mathbb{D} = \langle V, E, T \rangle$
- *Set of start vertices* $V_S \subseteq V$ *and set of final vertices* $V_F \subseteq V$

All-paths problem:

- *Find all paths* $\pi = (e_0, e_1, \cdots, e_{n-1}, e_n),\ e_k = (v_{k-1}, t_k, v_k)$ *in graph* $\mathbb{D}$, *such as* $l(\pi) = t_1 t_2 \cdots t_n \in L(\mathbb{G})$ *and* $v_0 \in V_S,\ v_n \in V_F$

Reachability problem:

- *Find all pairs* $\{(v_i, v_j) \mid \exists\ l(\pi) \in L(\mathbb{G})$ *и* $v_0 \in V_S,\ v_n \in V_F\}$

Motivation

- The problem of poor performance of CFPQ algorithms was formulated by Jochem Kuijpers as a result of an attempt to extend Neo4j
- Later, the matrix-based CFPQ algorithm showed high performance on real-world data

Overview

Generalized LL algorithm (GLL)

- Supports the entire class of context-free languages
- To reconstruct the paths the Shared Packed Parse Forest (SPPF) is supported

Proposed solution

- Based on GLL algorithm implementation in Iguana[1] project
- The ability to return as a result both a set of pairs of reachable vertices and the constructed SPPF
- Neo4j graph database is used as a graph storage
- The solution was integrated with Neo4j using Native Java API

[1] Repository of Iguana project: `https://github.com/iguana-parser/iguana`

Experimental study of proposed solution

Data

- **RDF Graphs**
  - *Grammars*

$$\begin{aligned} S \to & \overline{subClassOf}\ \ S\ subClassOf \mid \overline{type}\ \ S\ type \\ & \mid \overline{subClassOf}\ \ subClassOf \mid \overline{type}\ type \end{aligned} \tag{1}$$

$$S \to \overline{subClassOf}\ \ S\ subClassOf \mid subClassOf \tag{2}$$

$$\begin{aligned} S \to & broaderTransitive\ \ S\ \overline{broaderTransitive} \\ & \mid broaderTransitive\ \ \overline{broaderTransitive} \end{aligned} \tag{3}$$

- **Program analysis graphs**
  - *Grammar*

$$\begin{aligned} M &\to \overline{d}\ \ V\ \ d \\ V &\to (M?\ \overline{a})^*\ \ M?\ (a\ \ M?)^* \end{aligned} \tag{4}$$

All pairs results

	Graph name	$\|V\|$	$\|E\|$	#subClassOf	#type	#broaderTransitive	#a	#d
RDF	Go hierarchy	45 007	490 109	490 109	0	0	–	–
	Enzyme	48 815	86 543	8 163	14 989	8 156	–	–
	Eclass_514en	239 111	360 248	90 962	72 517	0	–	–
	Geospecies	450 609	2 201 532	0	89 065	20 867	–	–
	Go	582 929	1 437 437	94 514	226 481	0	–	–
	Taxonomy	5 728 398	14 922 125	2 112 637	2 508 635	0	–	–
Program analysis	Init	2 446 224	2 112 809	–	–	–	481 994	1 630 815
	Drivers	4 273 803	3 707 769	–	–	–	858 568	2 849 201
	Kernel	11 254 434	9 484 213	–	–	–	1 981 258	7 502 955

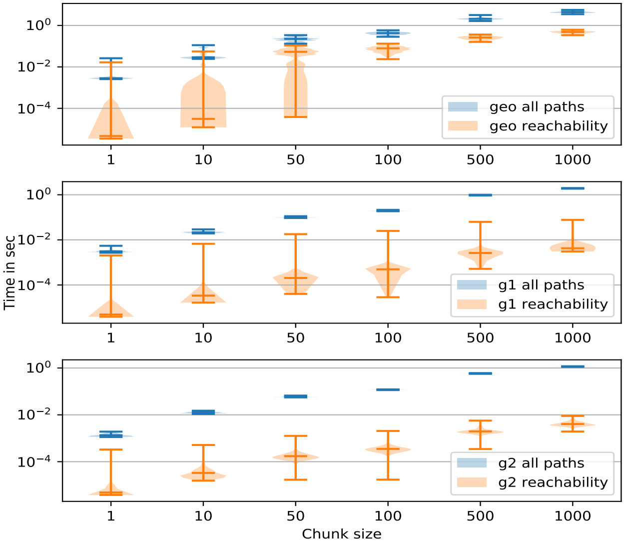
Graph name	G_1		G_2		*Geo*		*PointsTo*	
	time (sec)	#answer	time (sec)	#answer	time (sec)	#answer	time (sec)	#answer
Go hierarchy	564,72	588 976	2813,50	738 937	–	–	–	–
Enzyme	0,19	396	0,17	8163	8,54	14 267 542	–	–
Eclass_514en	295,06	90 994	279,80	96 163	–	–	–	–
Geospecies	2,64	85	2,00	0	256,86	226 669 749	–	–
Go	11,18	640 316	10,00	659 501	–	–	–	–
Taxonomy	43,72	151 706	29,58	2 112 637	–	–	–	–
Init	–	–	–	–	–	–	113,35	3 783 769
Drivers	–	–	–	–	–	–	736,81	18 825 025
Kernel	–	–	–	–	–	–	850,46	16 747 731

All pairs results

	Graph name	$\|V\|$	$\|E\|$	#subClassOf	#type	#broaderTransitive	#a	#d
RDF	Go hierarchy	45 007	490 109	490 109	0	0	–	–
	Enzyme	48 815	86 543	8 163	14 989	8 156	–	–
	Eclass_514en	239 111	360 248	90 962	72 517	0	–	–
	Geospecies	450 609	2 201 532	0	89 065	20 867	–	–
	Go	582 929	1 437 437	94 514	226 481	0	–	–
	Taxonomy	5 728 398	14 922 125	2 112 637	2 508 635	0	–	–
Program analysis	Init	2 446 224	2 112 809	–	–	–	481 994	1 630 815
	Drivers	4 273 803	3 707 769	–	–	–	858 568	2 849 201
	Kernel	11 254 434	9 484 213	–	–	–	1 981 258	7 502 955

Graph name	G_1		G_2		*Geo*		*PointsTo*	
	time (sec)	#answer	time (sec)	#answer	time (sec)	#answer	time (sec)	#answer
Go hierarchy	564,72	588 976	2813,50	738 937	–	–	–	–
Enzyme	0,19	396	0,17	8163	8,54	14 267 542	–	–
Eclass_514en	295,06	90 994	279,80	96 163	–	–	–	–
Geospecies	2,64	85	2,00	0	256,86	226 669 749	–	–
Go	11,18	640 316	10,00	659 501	–	–	–	–
Taxonomy	43,72	151 706	29,58	2 112 637	–	–	–	–
Init	–	–	–	–	–	–	113,35	3 783 769
Drivers	–	–	–	–	–	–	736,81	18 825 025
Kernel	–	–	–	–	–	–	850,46	16 747 731

Multiple-source results



Results

- Not only the matrix-based algorithm is applicable to real-world data
- Results are submitted to EDBT conference
- Discussion with Neo4j team about a deeper algorithm integration is in progress
- The main direction of further research is to find a way to effectively parallelize the GLL algorithm