

Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.

Scalable motif search in graphs using distributed computing

Andrew Esler

A thesis presented in partial fulfilment of the requirements for the degree of
a Masters in Computer Science

Massey University
Turitea
New Zealand

2012

Acknowledgments

I would like to thank my supervisor Jens Dietrich for the funding and motivation for this project. Patrick Rynhart provided extensive help with Massey's computing infrastructure. Thanks also to those who were sounding boards for ideas and pointed out my follies.

Abstract

Motif detection allows software engineers to detect antipatterns in software. By decreasing the number of antipattern instances in a piece of software, the overall quality of the software is improved. Current methods to find these antipatterns are slow and return results only when all antipatterns have been found. The GUERY framework is able to perform motif detection using multiple cores and deliver results as they are generated. By scaling GUERY to run on multiple machines, it was hoped that research requiring many queries on a graph could be performed significantly faster than is currently possible.

The objective of this thesis was to research and prototype mechanisms whereby GUERY could be run using a cluster of computers and results delivered as a stream to interested systems. A system capable of running on a cluster of machines and delivering a stream of results as they are computed was developed.

Contents

List of Figures	vii
List of Tables	ix
List of Listings	xi
1 Introduction	1
1.1 Introduction	1
1.2 Project Objective and Scope	5
1.2.1 Validation	5
1.2.2 Changes in the field	6
1.3 Spikes	6
1.4 Overview of thesis	7
2 Background	9
2.1 Graphs and Motifs	9
2.2 Graph Query Languages	14
2.3 GUERY	14
2.3.1 GUERY Queries	17
2.4 Graph Partitioning	17
2.4.1 Implementation	19
2.5 Cloud Computing	20
2.6 Buffering	21
3 Requirements and Validation Metrics	25
3.1 Requirements	25
3.1.1 Scalability	27
3.1.2 Result Streaming	27

3.1.3	Outside Project Scope	28
3.2	Validation Metrics	29
4	Existing Platforms and Frameworks	31
4.1	BSP	32
4.2	Pregel	33
4.3	Hadoop	34
4.3.1	MapReduce	35
4.3.2	Apache Hama	37
4.3.3	Giraph	37
4.3.4	GoldenOrb	38
4.4	Graph Databases	38
4.4.1	Neo4j	38
4.4.2	InfiniteGraph	38
4.5	Terracotta	39
4.6	HipG	39
4.7	Other Approaches	39
4.7.1	Custom GUERY clustering system	39
4.8	Summary	40
4.8.1	Comparison Table	40
4.8.2	Why Hadoop?	41
5	Hadoop GUERY Runner	45
5.1	Design	45
5.1.1	Input to HGR	46
5.1.2	Result Streaming	47
5.2	Constraints and Limitations	50
5.3	Local vs Hosted infrastructure	50
5.3.1	Hardware	52
5.3.2	Amazon Simple Storage Service	53
5.3.3	Problems Encountered when moving to Amazon	53
5.4	Experimental Evaluation	54
5.5	Requirements Validation	58
5.5.1	Set up complexity	59
5.5.2	Customization required	59
5.5.3	Find first	59
5.5.4	Find first 10	60
5.5.5	Find all	60

5.5.6	Synchronization overhead	60
5.5.7	Memory usage	60
5.5.8	Worker idle time	60
5.6	Weaknesses	61
5.6.1	Result Streaming Fault Tolerance	61
5.6.2	Amazon Infrastructure	62
5.7	Conclusions	62
6	Distributed Graph Processor	67
6.1	Why a messaging based system?	68
6.2	Messaging System Selection	69
6.3	RabbitMQ	71
6.3.1	Availability	73
6.4	Architecture	73
6.5	Message Queues	74
6.6	Components	78
6.6.1	Messaging	78
6.6.2	Coordinator	79
6.6.3	Worker	80
6.6.4	Partitioner	80
6.6.5	ResultProcessor	82
6.6.6	LiveLogViewer (LLV)	82
6.7	Scheduling and Task Management	83
6.8	Graph Partitioning and Partial Solutions	85
6.9	Job Execution Workflow	86
6.10	Scalability	87
6.11	Weaknesses	87
6.12	Fault Tolerance in Streaming Systems	88
6.13	Current State of DGPROC	89
6.14	Conclusions	91
7	Conclusions	93
7.1	Contributions	93
7.2	Future Work	94
	Bibliography	95
.1	Appendix GUERY Query Grammar	97
.2	Appendix HGR	104

.3	Appendix DGPROC	105
----	---------------------------	-----

List of Figures

1.1	<i>The definition of the subtype knowledge anti-pattern.</i>	3
1.2	<i>An instance of the subtype knowledge anti-pattern found in JRE</i>	
1.7.		4
2.1	<i>A graph with three distinct vertex roles and three distinct edge roles.</i>	11
2.2	<i>An example motif with three roles.</i>	11
2.3	<i>An example motif with two roles and an edge constraint.</i>	12
2.4	<i>Cloud service layers.</i>	22
4.1	<i>A diagram showing the layers of abstraction with respect to GUERY.</i>	32
5.1	<i>An overview of the HGR system showing how data flows.</i>	46
5.2	<i>The path a result takes once it is computed.</i>	49
5.3	<i>A graph of job execution time per instance type vs worker count for all three instance types.</i>	55
5.4	<i>A graph of job execution time per instance type vs worker count for m1.large and c1.medium instance types</i>	56
6.1	<i>The basic architecture of RabbitMQ.</i>	72
6.2	<i>The interaction between DGPROC components.</i>	75
6.3	<i>A diagram of the result subscriber notification process.</i>	76
6.4	<i>A diagram of how partitions are created and used during job processing.</i>	81

List of Tables

4.1	A comparison of systems	41
-----	-----------------------------------	----

Listings

2.1	ResultListener Interface	16
2.2	Subtype knowledge	17
2.3	Circular Dependency	17
2.4	Abstraction without Decoupling	18
4.1	Word Count Example map function	36
4.2	Word Count Example reduce function	36
5.1	Example of a vertex serialised in JSON	48
5.2	MotifTransformer interface	48
5.3	MotifProcessingWorker interface	49
5.4	A possible serialisation abstraction interface	64
6.1	Multithreaded GQL engine instantiation	67
6.2	DGPROC GQL engine instantiation	67
1	The ANTLR grammar file for GUERY queries.	97

