LDBC benchmarks: three aspects of graph processing

Gábor Szárnyas
szarnyas@mit.bme.hu

11th TUC meeting
Austin, TX

Mission statement

LDBC is a non-profit organization dedicated to establishing benchmarks, benchmark practices and benchmark results for graph data management SW.

LDBC's Social Network Benchmark is an industrial and academic initiative, formed by principal actors in the field of graph-like data management.

Graph processing landscape

Three key aspects

Graph processing landscape

OLTP	local queries
OLAP	global queries
analytics	global computations

Graph processing landscape

OLTP	local queries

Example: “Friends’ recent likes”

```
MATCH
  (u:User {id: $uID})-[:FRIEND]-(f:User)-[l:LIKES]->(p:Post)
RETURN f, p
ORDER BY l.timestamp DESC
LIMIT 10
```

OLAP	global queries
analytics	global computations

Graph processing landscape

OLTP	local queries	limited data	frequent up.


The LDBC Social Network Benchmark: Interactive Workload

Orri Erling et al.,
The LDBC Social Network Benchmark: Interactive Workload,
SIGMOD 2015

14 complex reads, 7 simple reads, 8 updates

Queries explore the graph around a given node

OLAP	global queries
analytics	global computations

Graph processing landscape

LDBC

OLTP	local queries	limited data	frequent up.
OLAP	global queries		

Example: “One-sided friendships”

```
MATCH (u1:User)-[:FRIEND]-(u2:User)-[l:LIKES]->(p:Post),
      (u1)-[:AUTHOR_OF]->(p)
WITH u1, u2, count(l) AS likes
WHERE likes > 10
  AND NOT (u1)-[:LIKES]->(:Post)<-[:AUTHOR_OF]-(u2)
RETURN u1, u2
```

analytics	global computations

Graph processing landscape

OLTP	local queries	limited data	frequent up.
OLAP	global queries	lots of data	infrequent up.

Gábor Szárnyas et al.,
An early look at the LDBC Social Network Benchmark's Business Intelligence Workload,
GRADES-NDA 2018

25 queries with infrequent executions

Queries explore a large portion of the graph

analytics	global computations

Graph processing landscape

LDBC

OLTP	local queries	limited data	frequent up.
OLAP	global queries	lots of data	infrequent up.
analytics	global computations		

Example: “Find the most central individuals.”

BFS breadth-first search

PR PageRank

CDLP community detection by label propagation

WCC weakly connected components

LCC local clustering coefficient

SSSP single-source shortest path

Graph processing landscape

OLTP	local queries	limited data	frequent up.
OLAP	global queries	lots of data	infrequent up.
analytics	global computations	all data	no updates


LDBC Graphalytics: A Benchmark for Large-Scale Graph Analysis on Parallel and Distributed Platforms

Alexandru Iosup et al.,
LDBC Graphalytics: A Benchmark for Large-Scale Graph Analysis on Parallel and Distributed Platforms,
VLDB 2016

One-time execution

No updates

Graph processing landscape

OLTP	local queries	limited data	frequent up.
OLAP	global queries	lots of data	infrequent up.
analytics	global computations	all data	no updates

Established solutions for relational data:

- Indexing
- Materialized views
- Column stores
- Data warehouses

Challenges

What makes graph queries difficult?

Choke points

- Choke point: a challenging aspect of query processing [QOPT/QEXE]
- Allows systematic benchmark design

CP-2.1: [QOPT] Rich join order optimization `TPC-H 2.3`

This choke-point tests the ability of the query optimizer to find optimal join orders. A graph can be traversed in different ways. In the relational model, this is equivalent as different join orders. The execution time of these orders may differ by orders of magnitude. Therefore, finding an efficient join (traversal) order is important, which in general, requires enumeration of all the possibilities. The enumeration is complicated by operators that are not freely re-orderable like semi-, anti-, and outer-joins. Because of this difficulty most join enumeration algorithms do not enumerate all possible plans, and therefore can miss the optimal join order. Therefore, these chokepoint tests the ability of the query optimizer to find optimal join (traversal) orders.

Peter Boncz, Thomas Neumann, Orri Erling,
TPC-H Analyzed: Hidden Messages and Lessons Learned from an Influential Benchmark,
TPCTC 2013

Graph processing challenges / 1

connectedness

the “curse of connectedness”

computer architecures

data structures contemporary computer architectures are good at processing are linear and simple hierarchical structures, such as *Lists*, *Stacks*, or *Trees*

caching and parallelization

a massive amount of random data access is required […] poor performance since the CPU cache is not in effect for most of the time. […] parallelism is difficult

B. Shao, Y. Li, H. Wang, H. Xia (Microsoft Research),
Trinity Graph Engine and its Applications,
IEEE Data Engineering Bulleting 2017

Graph processing challenges / 2

topology

existing graph query methods […] focus on the topological structure of graphs and few have considered attributed graphs.

attributes

applications of large graph databases would involve querying the graph data (attributes) in addition to the graph topology.

complex optimization

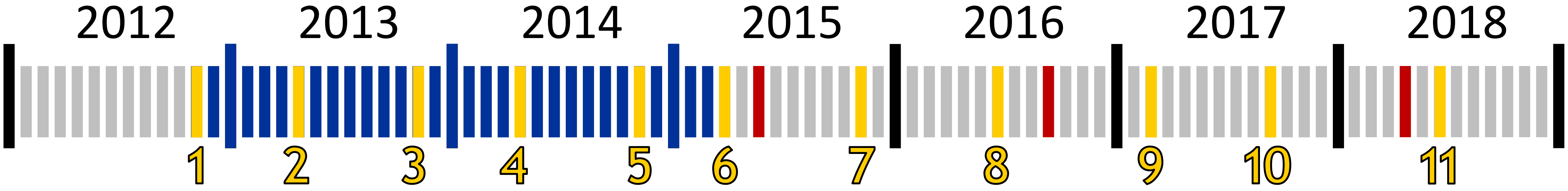
answering queries that involve predicates on the attributes of the graphs in addition to the topological structure […] makes evaluation and optimization more complex.


G-SPARQL: A Hybrid Engine for Querying Large Attributed Graphs

S. Sakr, S. Elnikety, Y. He (Microsoft Research),
G-SPARQL: A Hybrid Engine for Querying Large Attributed Graphs,
CIKM 2012

LDBC benchmarks

Timeline



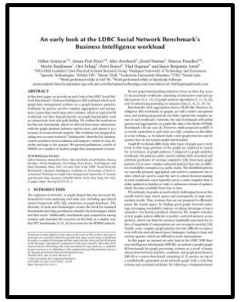
Interactive
SIGMOD 2015


The LDBC Social Network Benchmark: Interactive Workload

Graphalytics
VLDB 2016

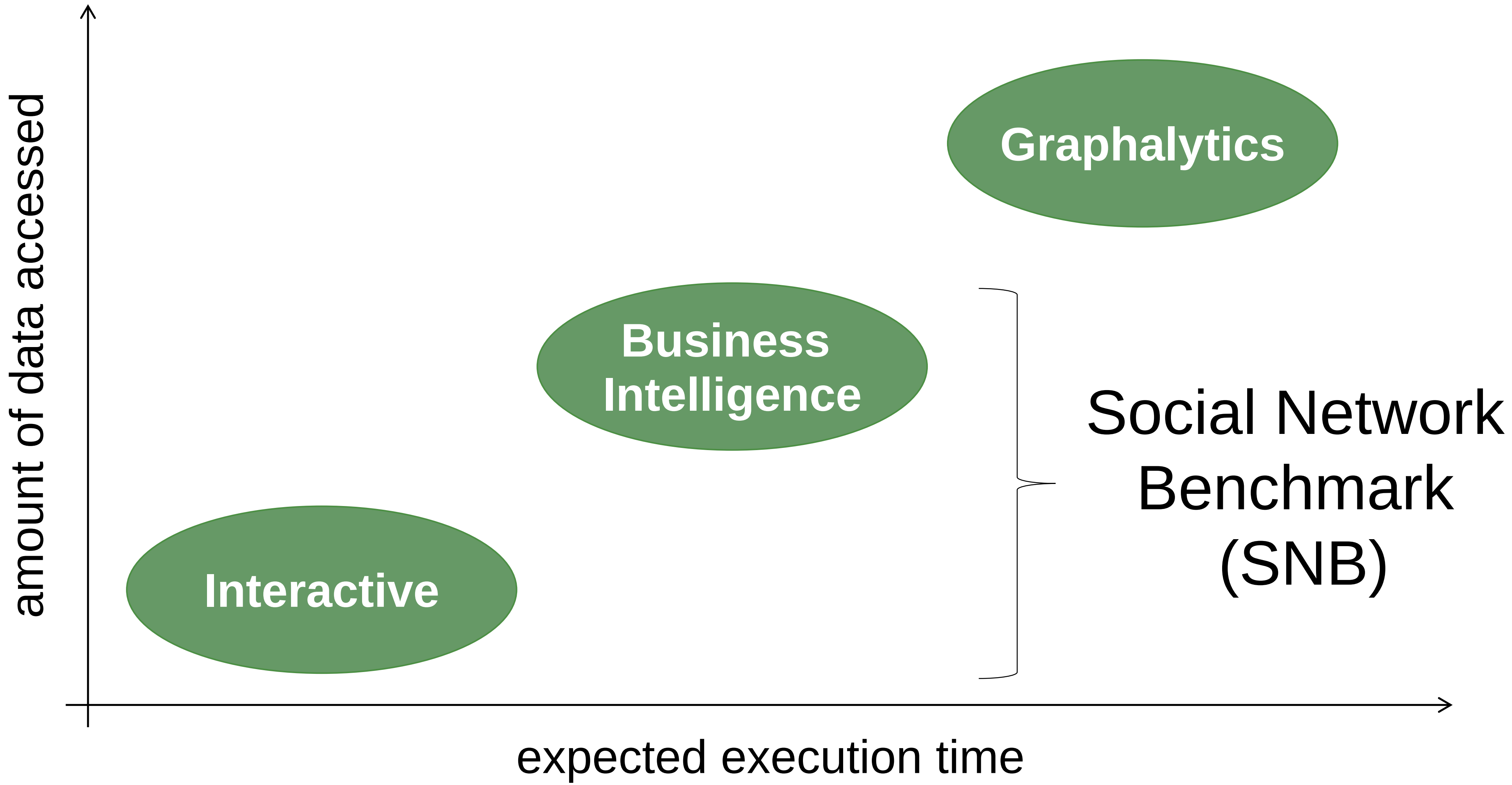

LDBC Graphalytics: A Benchmark for Large-Scale Graph Analysis on Parallel and Distributed Platforms

BI
GRADES-NDA @SIGMOD 2018

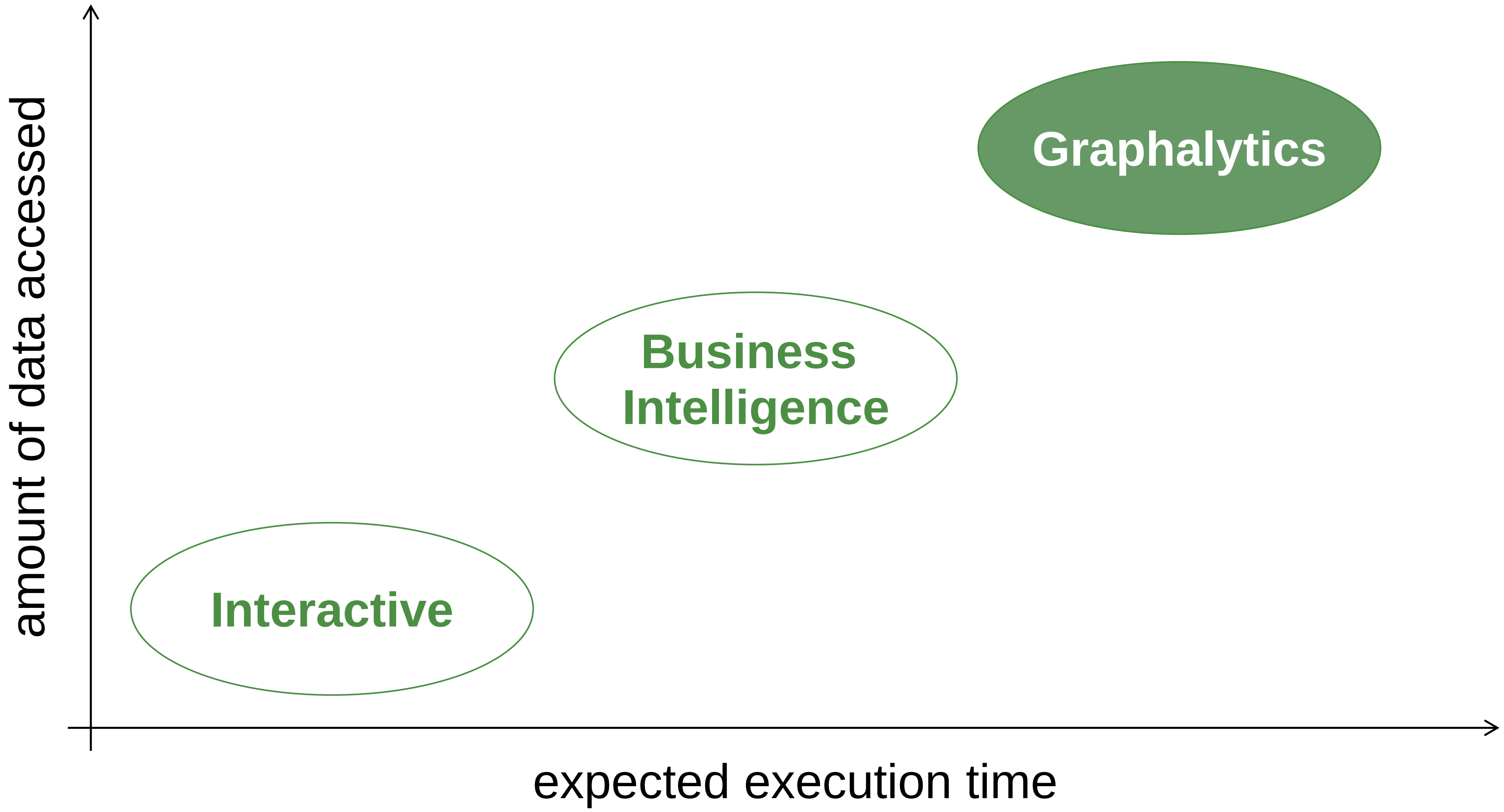

An early look at the LDBC Social Network Benchmark's Business Intelligence workload

EU FP7 project
TUC meetings
Benchmark papers

LDBC benchmarks at a glance



LDBC benchmarks at a glance



Graphalytics workload

Alexandru Iosup et al.

Graphalytics

- An LDBC benchmark
- Advanced benchmarking harness
- Many classes of algorithms used in practice
- Diverse real and synthetic datasets
- Diverse set of experiments representative for practice
- Renewal process to keep the workload relevant
- Extended toolset for manual choke-point analysis
- Enables comparison of many platforms, community-driven and industrial

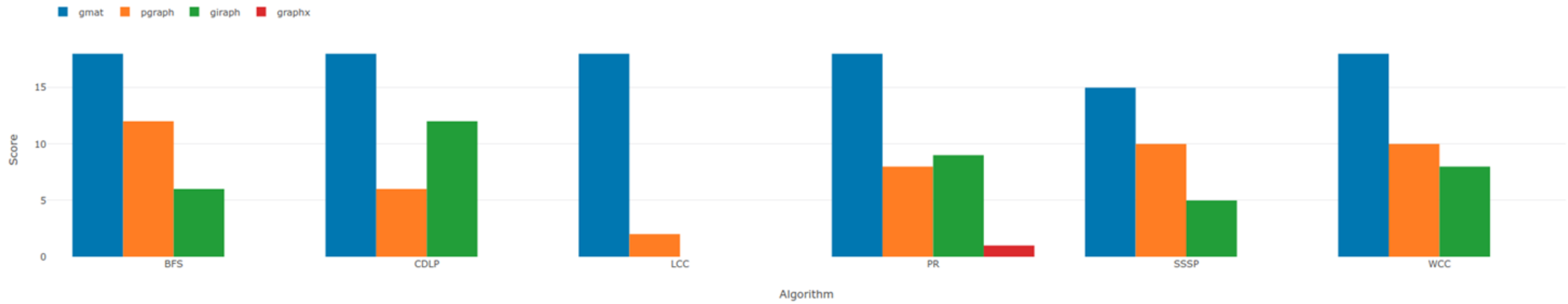
[Iosup et al., VLDB'16] [Guo et al., CCGRID'15] [Guo et al., IPDPS'14]

graphalytics.org

ldbcouncil.org/ldbc-graphalytics

Graphalytics Global Competition

- Systematic and periodic comparison of Graph processing systems.
- Register & submit benchmark results at graphalytics.org



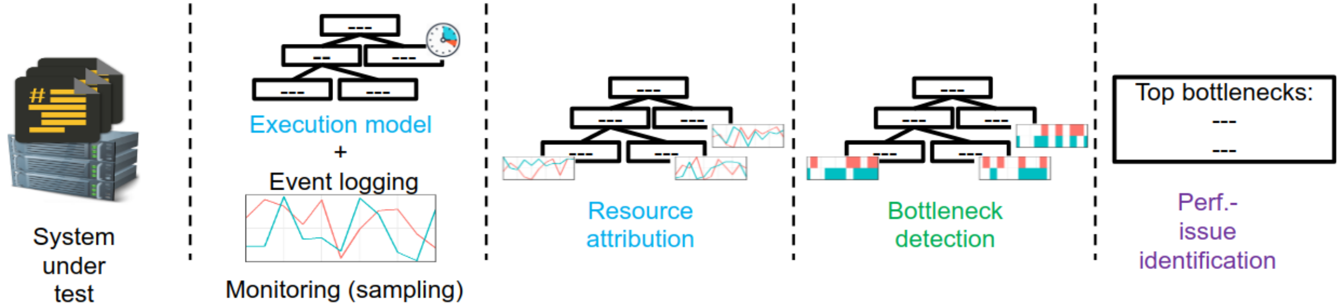
Rank	System name	Total score	BFS	CDLP	LCC	PR	SSSP	WCC
No. 1	gmat	105	18	18	18	18	15	18
No. 2	pgraph	48	12	6	2	8	10	10
No. 3	giraph	40	6	12	0	9	5	8
No. 4	graphx	1	0	0	0	1	0	0

BFS

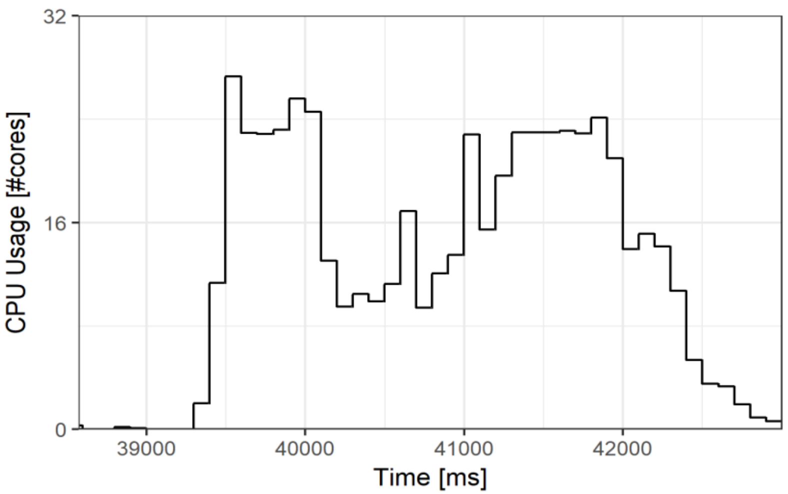
System name	Total score (EVPS)	Datagen-8_5-Fb	Datagen-8_6-Fb	Datagen-8_7-Zf	Graph500-25	Datagen-8_8-Zf	Datagen-8_9-Fb
gmat	18	2,185,887 kEVPS +3	2,170,844 kEVPS +3	438,309 kEVPS +3	1,930,948 kEVPS +3	461,637 kEVPS +3	2,549,718 kEVPS +3
pgraph	12	92,709 kEVPS +2	95,225 kEVPS +2	14,768 kEVPS +2	79,172 kEVPS +2	17,197 kEVPS +2	107,126 kEVPS +2
giraph	6	35,876 kEVPS +1	38,133 kEVPS +1	8,455 kEVPS +1	38,291 kEVPS +1	9,853 kEVPS +1	46,299 kEVPS +1
graphx	0	5,722 kEVPS +0	5,423 kEVPS +0	2,389 kEVPS +0	3,499 kEVPS +0	2,806 kEVPS +0	5,402 kEVPS +0

Grade10

Automated Bottleneck Detection and Performance Issue Identification



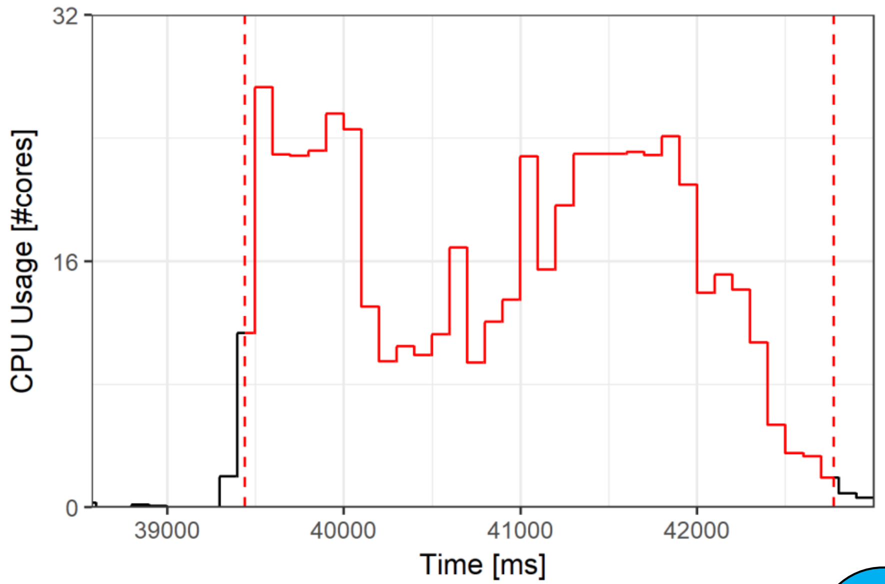
Without Grade10:

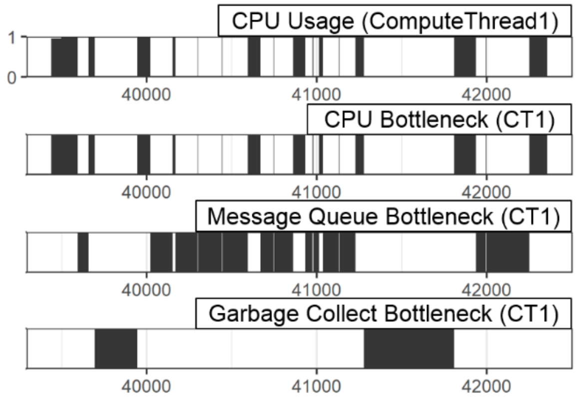


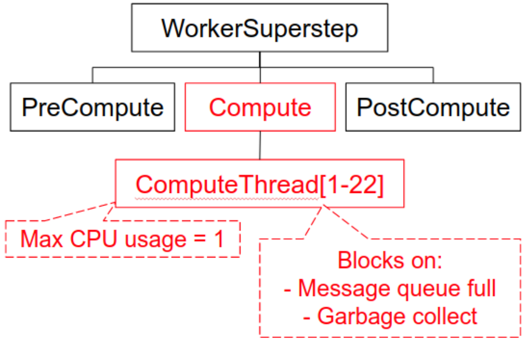
WorkerSuperstep

CPU usage < 32 cores (100%)
No bottleneck visible.. yet

With Grade10:







Average time bottlenecked for Compute/ComputeThread:

- **None: 0 ms (always bottlenecked)**
- **Message queue full: 1768 ms**
- **Garbage collect: 781 ms**
- **CPU: 748 ms**

Social Network Benchmark

SNB workloads

SNB task force

Arnau Prat
Sparsity / DAMA-UPC
(Task Force Leader)

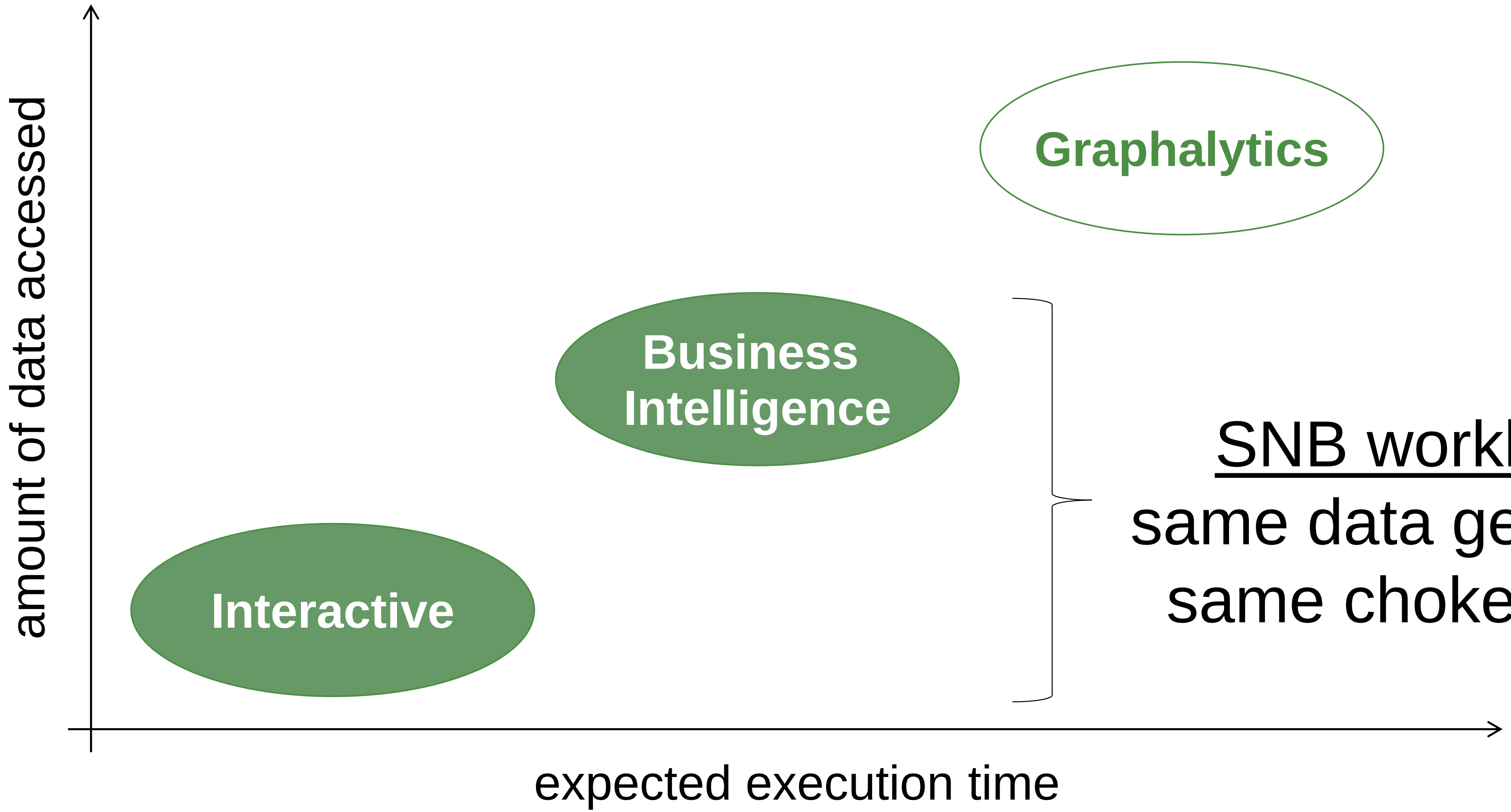
Alex Averbuch
Neo4j

Gábor Szárnyas
BME / MTA-BME

Vlad Haprian
Oracle Labs

Marcus Paradies
DLR

LDBC benchmarks at a glance



<u>SNB workloads</u>
same data generator,
same choke points

Data generator

github.com/ldbc/ldbc_snb_datagen

Social network graph

Realistic generator:

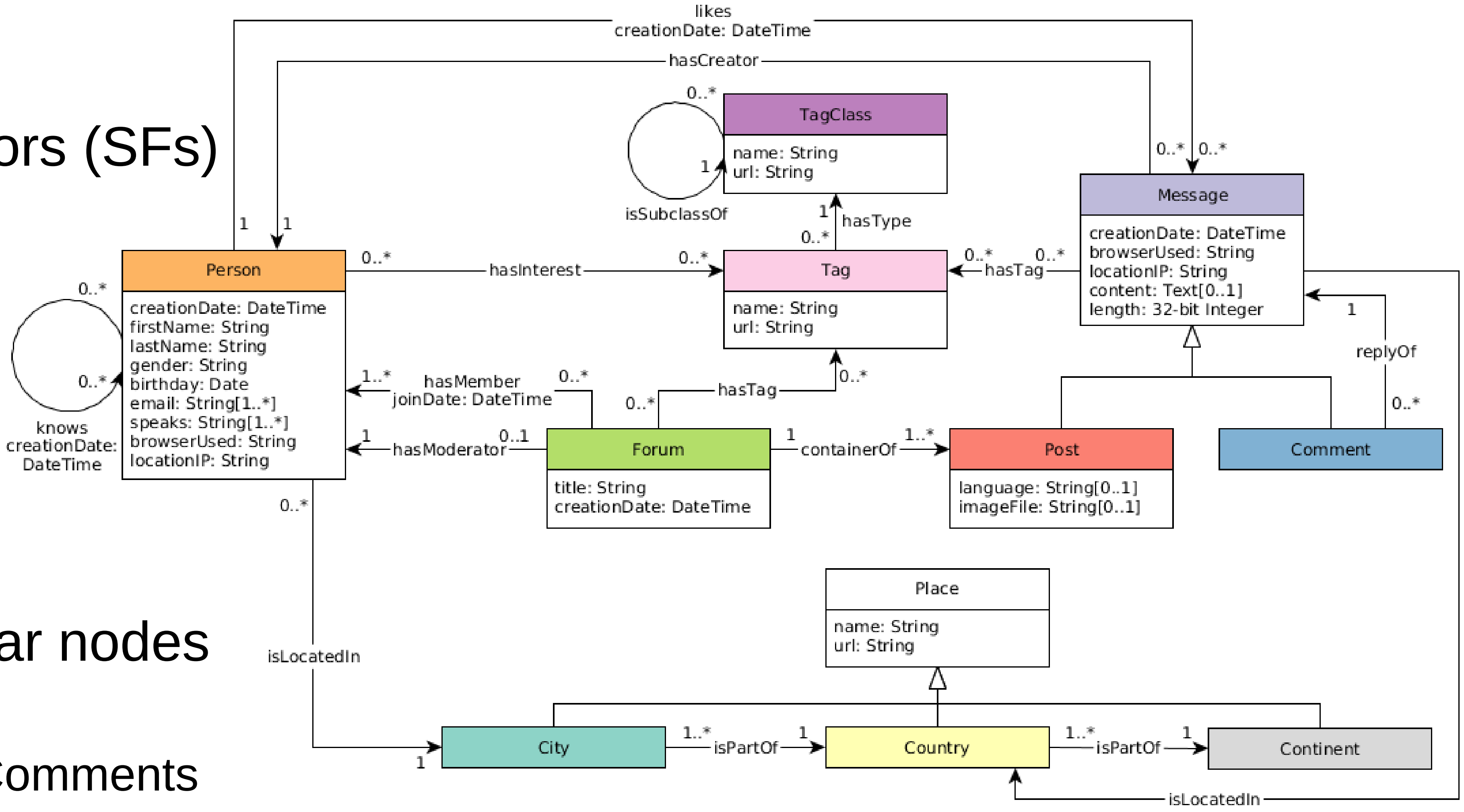
- DATAGEN
- Increasing scale factors (SFs)

Nodes:

- Collection attributes
- Type inheritance

Edges:

- Attributes
- Edges between similar nodes
  - Network of Persons
  - Reply tree of Posts/Comments



Workload specifications

github.com/ldbc/ldbc_snb_docs

Choke points [execution]

- Graph-specific challenges:
  - Cache-unfriendliness, difficult to index, difficult to parallelize

CP-3.3: [QEXE] Scattered index access patterns

This choke-point tests the performance of indexes when scattered accesses are performed. The efficiency of index lookup is very different depending on the locality of keys coming to the indexed access. Techniques like vectoring non-local index accesses by simply missing the cache in parallel on multiple lookups vectored on the same thread may have high impact. Also detecting absence of locality should turn off any locality dependent optimizations if these are costly when there is no locality. A graph neighborhood traversal is an example of an operation with random access without predictable locality.

Queries

BI 4 BI 5 BI 7 BI 8 BI 15 BI 16 BI 19 BI 21 BI 22 BI 23 BI 25 IC 5 IC 7 IC 8 IC 9 IC 10 IC 11 IC 12 IC 13 IC 14

Choke points [language]

New choke points to cover *language features*

- CP-8.1: Complex patterns
- CP-8.2: Complex aggregations
- CP-8.3: Ranking-style queries
  - “arg min”-style queries, `OVER` and `rank()` in PostgreSQL
- CP-8.4: Query composition
  - Focal point of G-CORE
- CP-8.5: Dates and times
  - Recent advancement in openCypher and Neo4j
- CP-8.6: Handling paths
  - Focal point of G-CORE

Choke points [language]: Paths

1. Path unwinding
   - Higher-order queries
   - e.g. for a given path, calculate a score for each edge and summarize them
2. Matching semantics ~ walks vs. trails vs. simple paths
   - Homomorphism-based
   - Isomorphism-based
     - No-repeated-anything
     - No-repeated-node semantics
     - No-repeated-edge semantics
3. Regular path queries (RPQs)


Foundations of Modern Query Languages for Graph Databases

R. Angles et al.,
Foundations of Modern Query Languages for Graph Databases,
ACM Computing Surveys, 2017

Choke points [language]: Paths

CP-8.6: [LANG] Handling paths

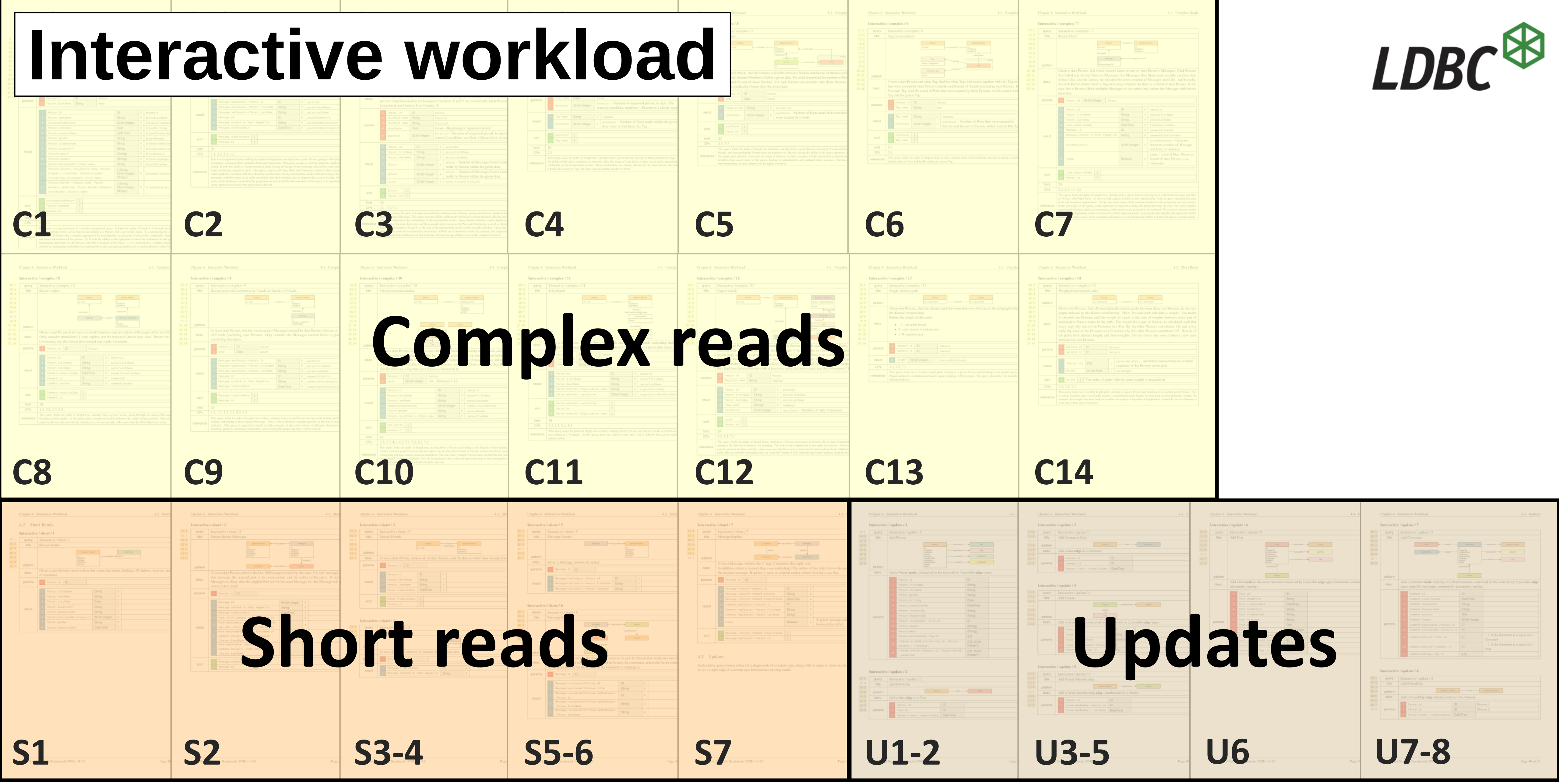
Handling paths as first-class citizens is one of the key distinguishing features of graph database systems [3]. Hence, additionally to reachability-style checks, a language should be able to perform *path unwinding* [1], i.e. express queries that operate on elements of a path such as calculating a score for each edge of a path. Also, some use cases specify uniqueness constraints on paths, e.g. that a certain path must not have repeated nodes (referred to as "walks" in graph theory) or not have repeated edges ("trails" in graph theory). Following the definitions of paper [1], *homomorphism-based semantics* (no constraints on repetitions) and multiple flavours of *isomorphism-based semantics* (no-repeated-node, no-repeated-edge, and no-repeated-anything).

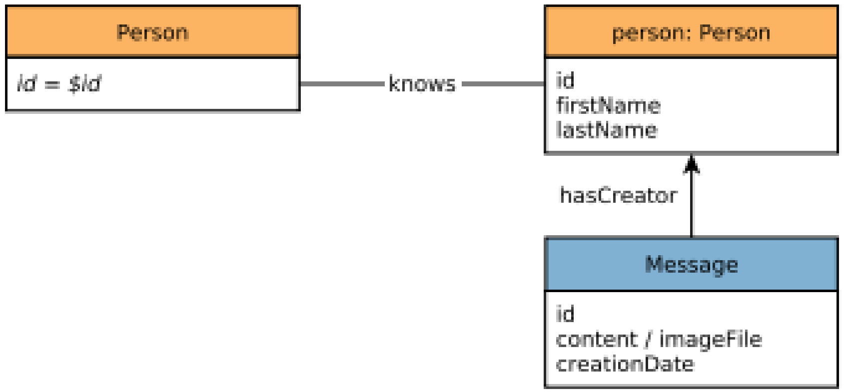
Cypher. Cypher uses *no-repeated-edge matching semantics* (in return, this semantics is sometimes dubbed as *cyphermorphism*). Configurable matching semantics (e.g. `MATCH ALL WALKS`) were proposed in the openCypher language. RPQs are also proposed in the openCypher language as *path patterns*.

G-CORE. G-CORE treats paths as *first-order citizens*: its *path property graph data model* can store paths in the graph model itself. However, the language only supports shortest path semantics (for tractability reasons) and does not allow enumeration of all paths. G-CORE uses *homomorphism-based matching semantics*.

SPARQL. SPARQL uses *homomorphism-based matching semantics* and supports RPQs as *property paths*. Isomorphism-based matching semantics can be expressed by introducing custom filtering condition on predicates, e.g. `FILTER ( ?e1 != ?e2 )`.

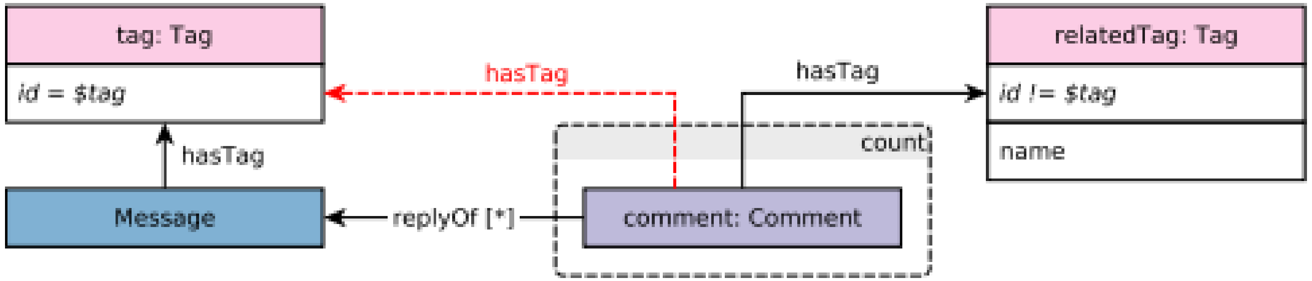
Interactive workload



<table>
<tr><td>query</td><td>Interactive / complex / 2</td></tr>
<tr><td>title</td><td>Recent posts and comments by your friends</td></tr>
<tr><td>pattern</td><td>
</td></tr>
<tr><td>desc.</td><td>Given a start Person, find (most recent) Messages from all of that Person's friends, that were created before (and including) a given date.</td></tr>
<tr><td>params</td><td><table>
<tr><td>1</td><td><code>Person.id</code></td><td>ID</td><td></td></tr>
<tr><td>2</td><td><code>date</code></td><td>DateTime</td><td></td></tr>
</table></td></tr>
<tr><td>result</td><td><table>
<tr><td>1</td><td><code>Message-hasCreator->Person.id</code></td><td>ID</td><td>R</td><td></td></tr>
<tr><td>2</td><td><code>Message-hasCreator->Person.firstName</code></td><td>String</td><td>R</td><td></td></tr>
<tr><td>3</td><td><code>Message-hasCreator->Person.lastName</code></td><td>String</td><td>R</td><td></td></tr>
<tr><td>4</td><td><code>Message.id</code></td><td>ID</td><td>R</td><td></td></tr>
<tr><td>5</td><td><code>Message.content or Post.imageFile</code></td><td>String</td><td>R</td><td></td></tr>
<tr><td>6</td><td><code>Message.creationDate</code></td><td>DateTime</td><td>R</td><td></td></tr>
</table></td></tr>
</table>

LDBC

BI workload

query	BI / read / 8
title	Related topics
pattern	
desc.	Find all Messages that have a given Tag. Find the related Tags attached to replies of these Messages (direct relation not transitive). but only of those replies that do not have the given Tag. Group the Tags by name, and get the count of replies in each group.
params	1 tag 32-bit Integer
result	1 relatedTag.name String R 2 count 32-bit Integer R
sort	1 count ↓ 2 relatedTag.name ↑
limit	100
CPs	1.6, 3.3, 5.2

Driver and implementations

github.com/ldbc/ldbc_snb_driver

github.com/ldbc/ldbc_snb_implementations

Implementing an SNB workload

1. Get / generate data set
2. Implement loader
3. Implement queries and driver adapter

<u>Validation</u>

1. Get / generate validation data sets
2. Cross-validate for multiple SFs
3. If required, fix issues and go to 2.

Validation is very time consuming, but…

- Even after 2 validated tools, there were bugs in *both* implementations
- Even after 3 validated tools, there were ambiguities in the spec

Implementations / Interactive workload

The SIGMOD 2015 paper had implementations for Virtuoso and Sparksee.

Current implementations:

- PostgreSQL
- Sparksee
- SPARQL (some fixes by students of Tomer Sagi @ University of Haifa)

Next up:

- Cypher
- ?

Implementations / BI workload

Cross-validated implementations:

• Cypher	Neo4j	25/25
• SPARQL	Stardog	24/25
• SQL	PostgreSQL	25/25
• Imperative (C++)	Sparksee	25/25
• PGQL	Oracle Labs PGX	10/25

Next up:

- Spark SQL
- Cypher for Apache Spark
- ?

Incremental View Maintenance (IVM)

LDBC BI queries helped identify challenges for IVM on graphs:

- Complex aggregations
- Nested data structures
- Higher-order queries (path unwinding)

<u>Results</u>:

- Rules to transform queries to *nested relational algebra* and to *flat RA*
- Open-source prototype (ingraph/openCypher), supports ~15/25 BI queries
- Incremental higher-order queries are an open problem

Gábor Szárnyas et al.,
Reducing Property Graph Queries to Relational Algebra for Incremental View Maintenance, arXiv preprint

Progress and roadmap

SNB progress report: 10th vs. 11th TUC

pre-10th TUC

- 54 Trello cards
- Specification
  - 180+ commits
- DATAGEN
  - 40+ commits
- “Close to publication”

10th – 11th TUC

- 67 Trello cards
- Specification
  - 250+ commits
- DATAGEN
  - 50+ commits
- Driver and implementations
  - 600+ commits

Roadmap – 10th TUC

- Implement & validate for Neo4j, PostgreSQL and Sparksee ✓
- Publish a subset of the benchmark in a workshop ✓
  - GraphQ @ EDBT (late Nov)
  - GRADES @ SIGMOD (late March) ✓
- Gather feedback & refine ✓
- Define update operations ✗

- We are recruiting! ✓

Roadmap – 11th TUC

- Social Network Benchmark workloads
  - Goal: publish the BI workload as an industry track conference paper
  - Help industry adoption
  - Define update operations: insertions and deletes (cf. GDPR)

- Graphalytics
  - Goal: establish Graphalytics 2.0
  - Run global competition

- We are still recruiting!

Acknowledgements

Gábor Szárnyas was partially supported by NSERC RGPIN-04573-16 (Canada) and the MTA-BME Lendület Cyber-Physical Systems Research Group (Hungary).

DAMA-UPC research was supported by the grant TIN2017-89244-R from MINECO (Ministerio de Economia, Industria y Competitividad) and the recognition 2017SGR-856 (MACDA) from AGAUR (Generalitat de Catalunya).

Sparsity thanks the EU H2020 for funding the Uniserver project (ICT-04-2015-688540).

Hungarian Academy of Sciences

MTA-BME Lendület
Cyber-Physical Systems Research Group

MŰEGYETEM 1782

Department of Measurement and Information Systems

McGill

Department of Electrical and Computer Engineering