BENCHMARKING ESSENTIAL GRAPH QUERIES

Renzo Angles

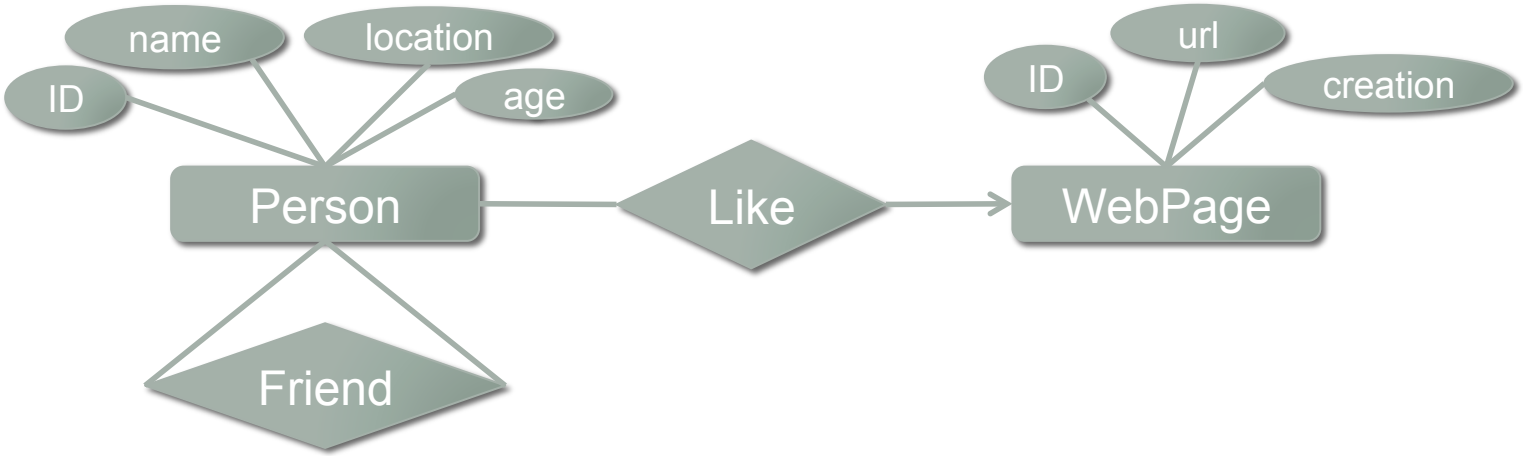
Assistant Professor, Universidad de Talca (Chile)
Post-doc, Vrije University Amsterdam

1st Workshop on Graph-based Technologies and Applications (Graph-TA)
Barcelona, February 19, 2013

Motivation

- Increasing amount of graph data
- NOSQL databases => Graph (oriented) databases
- What is the most suitable graph database?
  - Theoretical comparison (complexity and expressive power)
  - Empirical comparison (performance, usability, etc.)
  - Benchmarks (there is not a standard one)
  - The application domain is very important
- Our work
  - Empirical comparison of loading and querying data at low-scale (1K, 100K, 500K, 1M)
  - Experience on using several graph databases

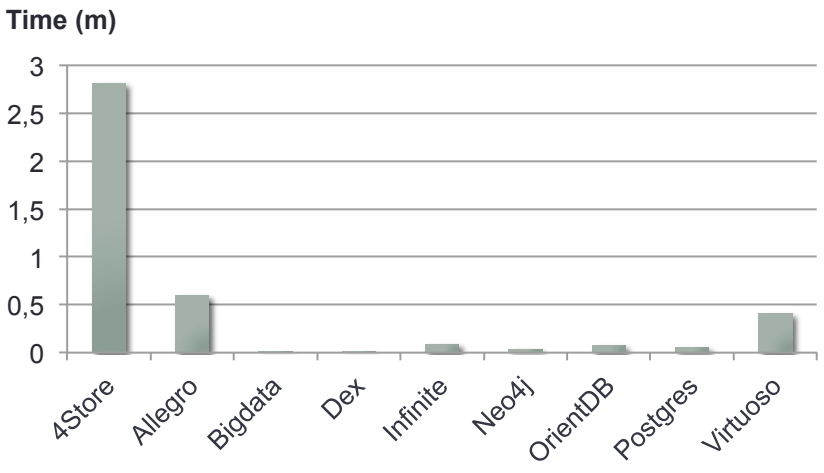
Benchmark use case



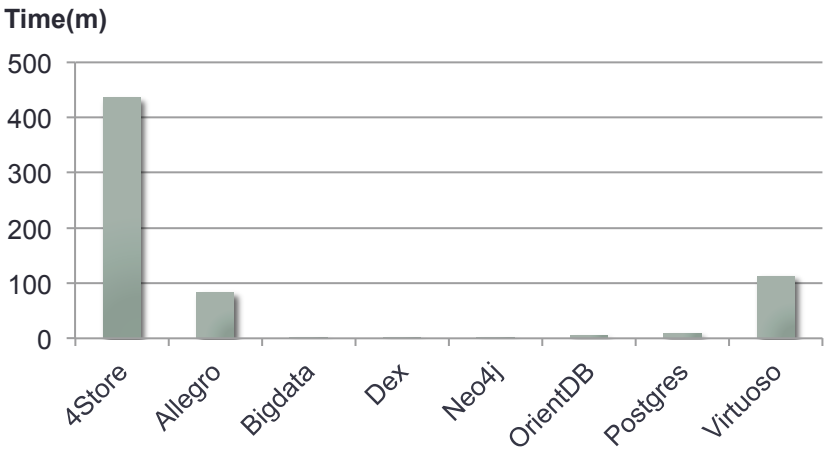
- 12 essential graph queries
  - Attribute searching (Get people with a given name)
  - Node/edge adjacency (Get people that likes a given Web page)
  - Fixed-length paths (Get the friends of the friends of a given person)
  - Reachability (Is there a “friend” connection between two people?)
  - Pattern matching (Get the common friends between two people)
  - Aggregates (Get the number of friends of a given person)

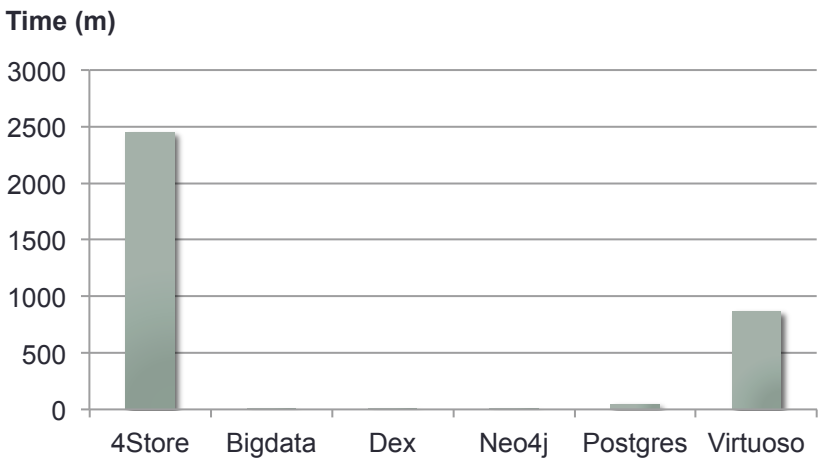
Data loading test

N=1.000 E=5.874

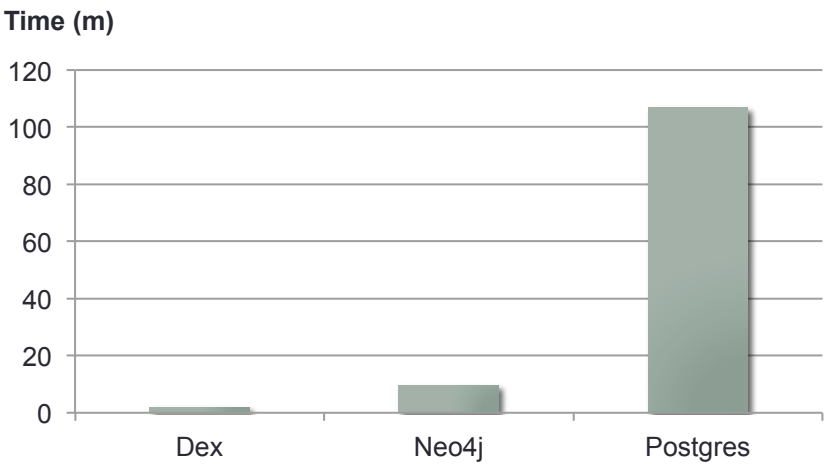


N=100.000 E=1.002.216





N=500.000 E=5.735.332



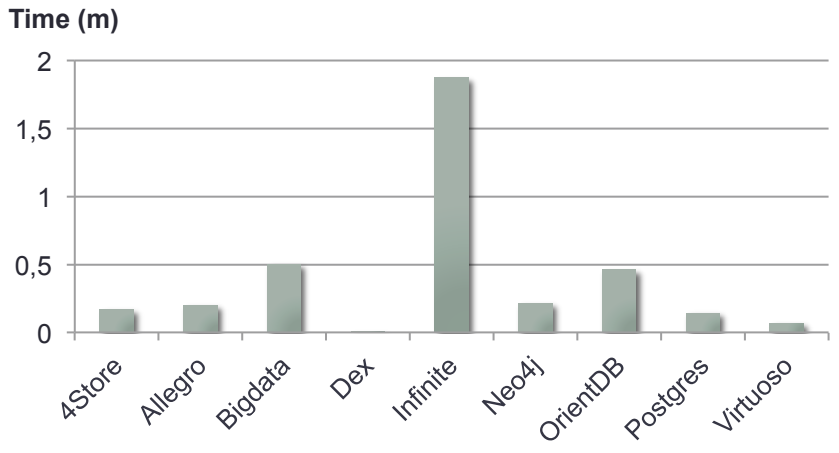
N=1.000.000 E=12.094.498

Expressing graph queries

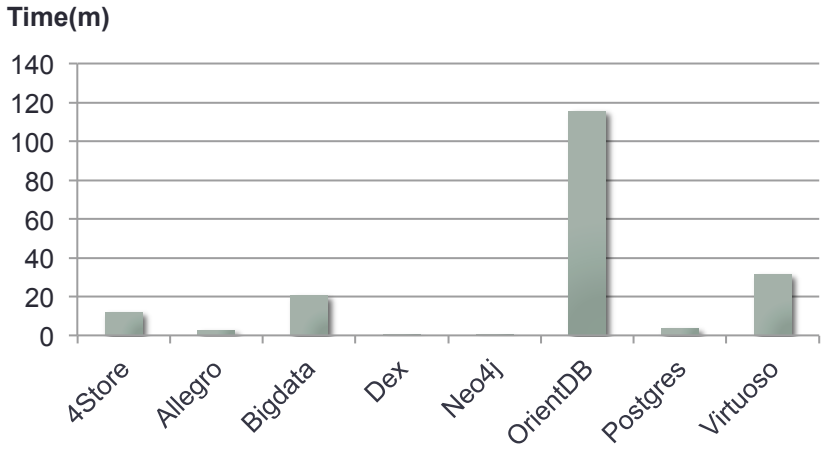
	Get the friends of a person identified by id 10
Dex	long person_id = dex_graph.findObject(pid, dexvalue.setLong(10)): dex_graph.neighbors(person_id, friend, EdgesDirection.Outgoing);
InfiniteGraph	Person person = this.findPersonById(10); Iterator<VertexHandle> it = person.getNeighbors().iterator(); while (it.hasNext()) { … }
Neo4j	START p=node:peopleIdIndex(id=10) MATCH p-[:friend]->f RETURN f
OrientDB	SELECT FROM ographvertex WHERE in[label='friend'].out in (select rid from index:personIdx where key = 10)
SPARQL	SELECT ?friend WHERE { ?person <http://sn.org/voc/person#id> 10 . ?person <http://sn.org/voc/friend> ?friend }

Data querying test (12 queries x 100 instances)

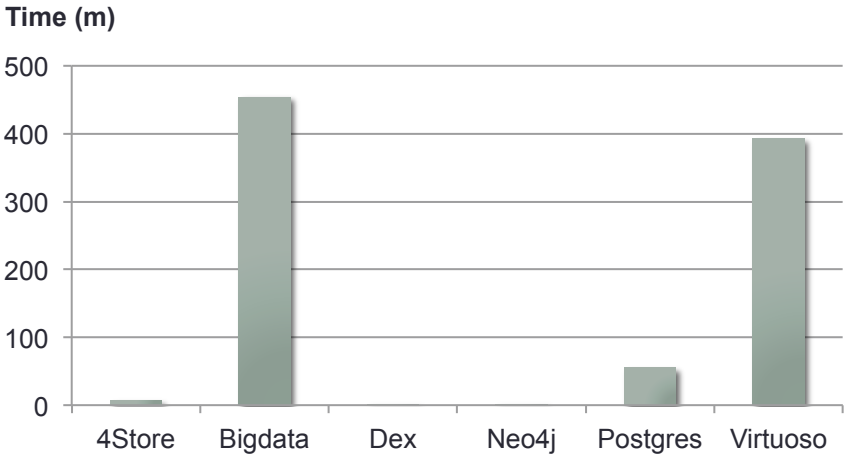
N=1.000 E=5.874



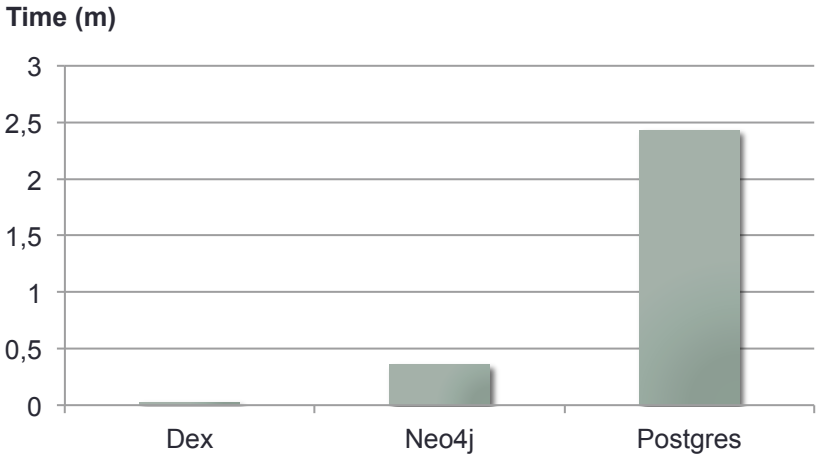
N=100.000 E=1.002.216



N=500.000 E=5.735.332



N=1.000.000 E=12.094.498



Conclusions

- We developed a benchmark for essential graph queries
- We present
  - our experience on using current graph databases
  - preliminary results of performance for loading and querying data
- The comparison of current graph databases is not an easy task
  - There are several approaches
  - Standards are required (e.g., a graph query language)
- Opportunity:
  - Research on foundations on graph data management
  - Development of benchmarks for graph databases
  - Use and testing of graph databases in real-life applications