How to generate parameters for RDF benchmarks?

Andrey Gubichev Renzo Angles Peter Boncz

Technische Universität München

CWI, Amsterdam

March 2014

DB Benchmarking Intro

A benchmark consists of:

- Dataset (Real or Synthetic)
- Queries: Templates with parameters

DB Benchmarking Intro

A benchmark consists of:

- Dataset (Real or Synthetic)
- Queries: Templates with parameters

Procedure:

- Sample N parameters from the domain, use template queries with these parameters
- Report average runtime (Query-per-Hour etc)

DB Benchmarking Intro

A benchmark consists of:

- Dataset (Real or Synthetic)
- Queries: Templates with parameters

Procedure:

- Sample N parameters from the domain, use template queries with these parameters
- Report average runtime (Query-per-Hour etc)

Two audiences:

- Users: what is the best system for my workload?
- Database architects: how does my system hadnle certain technical challenges?

Examples:

- Relational: TPC-family
- RDF: BSBM, SP2B, LUBM, LDBC, ...

Query Template: Example

Find all the people with the given name in the given country:

SELECT ?x WHERE {
?x hasName %*name*%.
?x livesIn %*country*% }

Query Template: Example

Find all the people with the given name in the given country:

SELECT ?x WHERE {
?x hasName %*name*%.
?x livesIn %*country*% }

Sample two pairs of parameters:

Name = John
Country = USA:

- very unselective query
- two unselective scans are joined
- tests the power of the join operator (and dictionary lookup)

Name = Mary
Country = China:

- very selective query
- sparse join
- forces the system to skip most of the input (Bloom filters)

Query Template: Example

Find all the people with the given name in the given country:

```
SELECT ?x WHERE {
?x hasName %name%.
?x livesIn %country% }
```

Pick 100 random parameters, got average query runtime: 120 ms

- Is our dictionary bad?
- Or do we need Bloom filters?

Plan

- What happens when we use uniformly sampled values as parameter bindings?
- How to generate better parameters?

Scope of the problem:

- Relevant for all benchmarks
- Relational benchmarks (TPC-H etc) use synthetic data with simple distributions and almost no correlations. Uniform sampling works fine
- RDF benchmarks: real data, synthetic data with generated correlations. Uniform sampling does not work

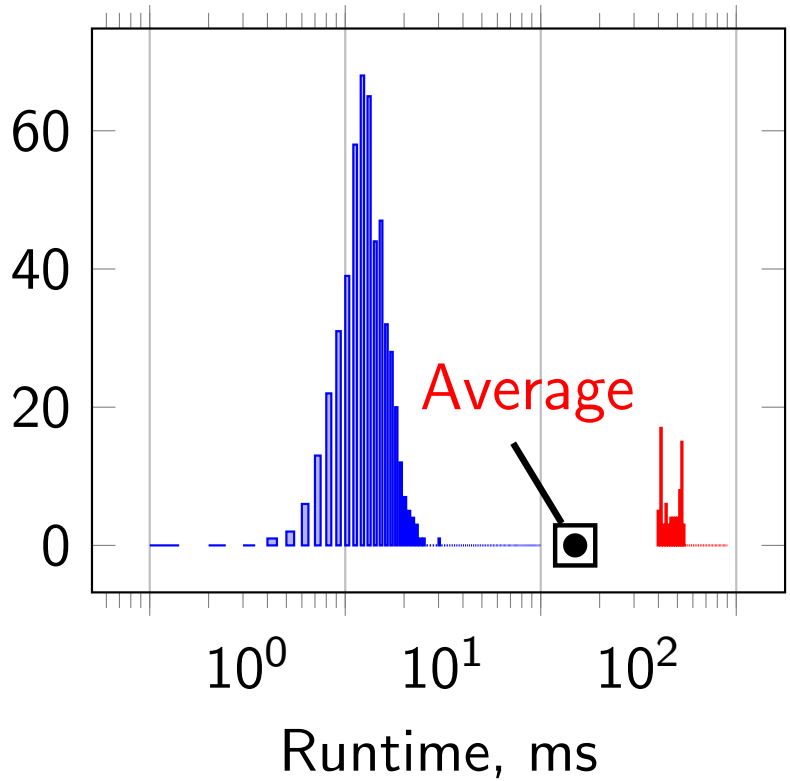
Runtime distribution has high variance

BSBM Query 4: *Among all the products of given type, find the feature with the highest ration between price with and without that feature*

- Type forms a hierarchy
- Amount of data touched depends on how generic the type is
- In a sample with 100 parameters the variance of runtime is $674 \cdot 10^6$

Average runtime is not representative

- BSBM Query 4: *Among all the products of given type, find the feature with the highest ration between price with and without that feature*
- Two groups of queries, depending on the parameter
- Average is outside of both groups



Different parameters result in different plans

LDBC Query 3: *Find friends of friends (FoF) of a given person P that have been to countries X and Y*

- X, Y = USA, Canada. Large intersection, first find FoF.
- X, Y = Zimbabwe, Finland. Small intersection, start with countries
- What if P has a lot of friends? or very few friends?

Some queries have up to 17 distinct plans!

Parameter Generation: Problem Definition

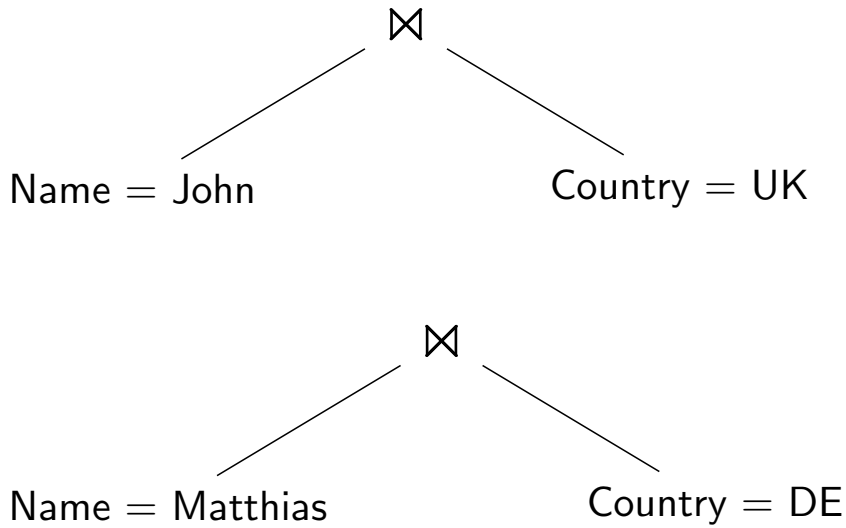
Select N distinct parameter bindings such that across all the parameters:

- the optimal plan is the same
- the intermediate results in the plan have the same size

Parameter Generation: Problem Definition

Select N distinct parameter bindings such that across all the parameters:

- the optimal plan is the same
- the intermediate results in the plan have the same size



Parameter Generation: Will it help?

- We look for parameters that result in the same plan and same cost function value
- Cost function strongly correlates with runtime
- If all our parameters result in the same cost function value, runtime will be normally distributed

Parameter Generation: How hard is it?

Not trivial!

- Finding the optimal plan is NP-hard
- Even if the have candidate parameter bindings, checking that they yield the same plan is NP-hard

Parameter Generation: How to solve it?

- What are the important factors that change the intermediate results of a query?

LDBC Query 4: New trends

```
SELECT ?x WHERE {
%Person% knows ?fr .
?post hasCreator ?fr.
?post hasTag ?tag .
?tag name ?tagname .
?post creationDate ?date . }
```

Parameter Generation: How to solve it?

- What are the important factors that change the intermediate results of a query?
- For every Person
  - Multiple Friends $\#F$
  - Friends have Multiple Posts $\#FP$
  - Posts have Multiple Tags $\#FPT$
- Order: Person – #F – #FP – #FPT
- Factors: amount of intermediate results
- Find Persons that have all these counts similar to each other

LDBC Query 4: New trends

```
SELECT ?x WHERE {
%Person% knows ?fr .
?post hasCreator ?fr.
?post hasTag ?tag .
?tag name ?tagname .
?post creationDate ?date . }
```

Parameter Generation: How to solve it?

- What are the important factors that change the intermediate results of a query?
- For every Person
  - Multiple Friends *#F*
  - Friends have Multiple Posts *#FP*
  - Posts have Multiple Tags *#FPT*
- Order: Person – #F – #FP – #FPT
- Factors: amount of intermediate results
- Find Persons that have all these counts similar to each other

Factors:

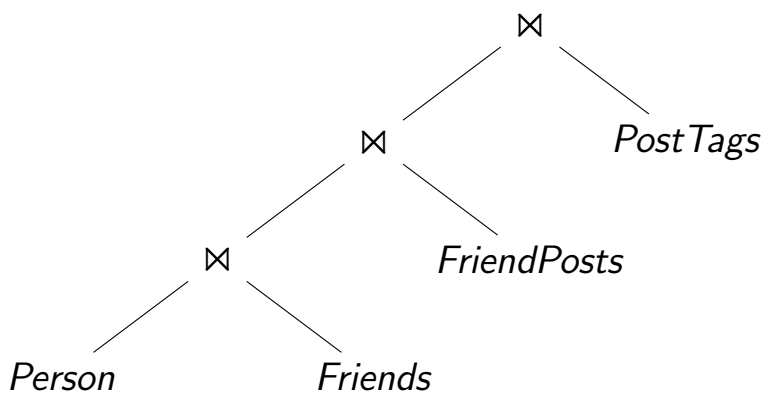
PersonID	#F	#FP	#FPT
12	80	213	441
34	81	223	510
...	...	...	...

Parameter Generation: How to solve it?

- What are the important factors that change the intermediate results of a query?
- For every Person
  - Multiple Friends *#F*
  - Friends have Multiple Posts *#FP*
  - Posts have Multiple Tags *#FPT*
- Order: Person – #F – #FP – #FPT
- Factors: amount of intermediate results
- Find Persons that have all these counts similar to each other

Factors:

PersonID	#F	#FP	#FPT
12	80	213	441
34	81	223	510
...	...	...	...



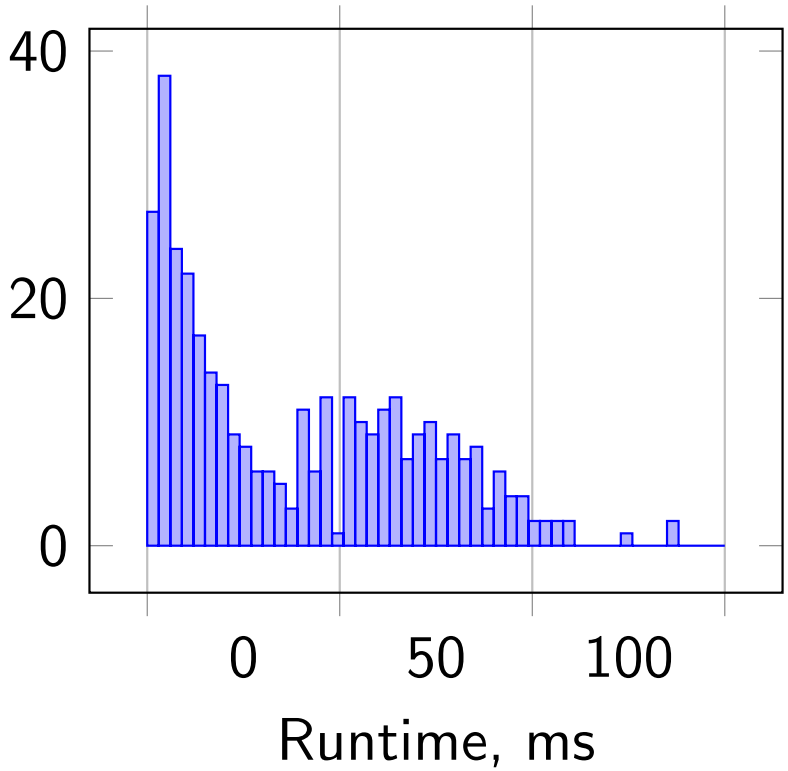
Parameter Generation: Data Mining Approach

- Step 1: Identify cardinality-changing factors from queries
  - in LDBC: Friend Count, Friends Posts Count, Tags Count, etc
- Step 2: Collect the counts of the factors for all possible parameter bindings
  - For every Person, find Friend Count, Friends Post Count etc
- Step 3: Greedily mine the most similar rows from the factors table
  - Find all Persons with similar number of Friends and Posts

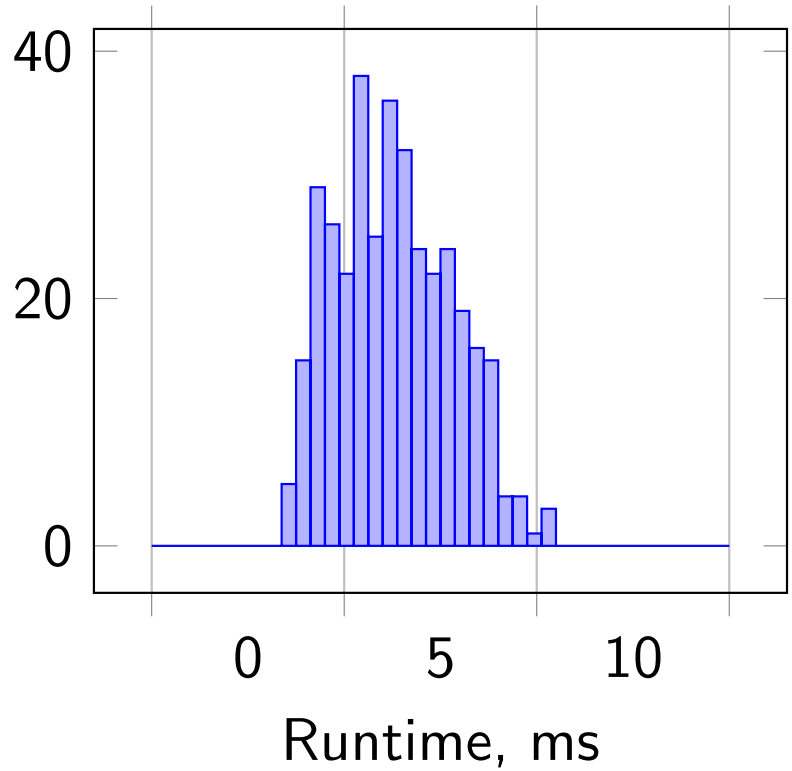
Parameter Generation: Results

LDBC Q4: *Find new trends in the network around user P in last month*

Uniform Sample:



Mined Parameters:



Conclusions

- Uniform sampling of parameter bindings makes fair comparison of benchmark results hard
- We introduce data mining-based parameter discovery