Databaseless Queries: Using Calcite as Research Testbed for Hybrid Cloud Data Integration

Josef Spillner, Zurich University of Applied Sciences

Ninth International Workshop on Serverless Computing (WoSC9) 2023, December 11, Bologna, Italy

https://www.serverlesscomputing.org/wosc9/demos/d14 (abstract)

https://zenodo.org/records/8430233 (software)

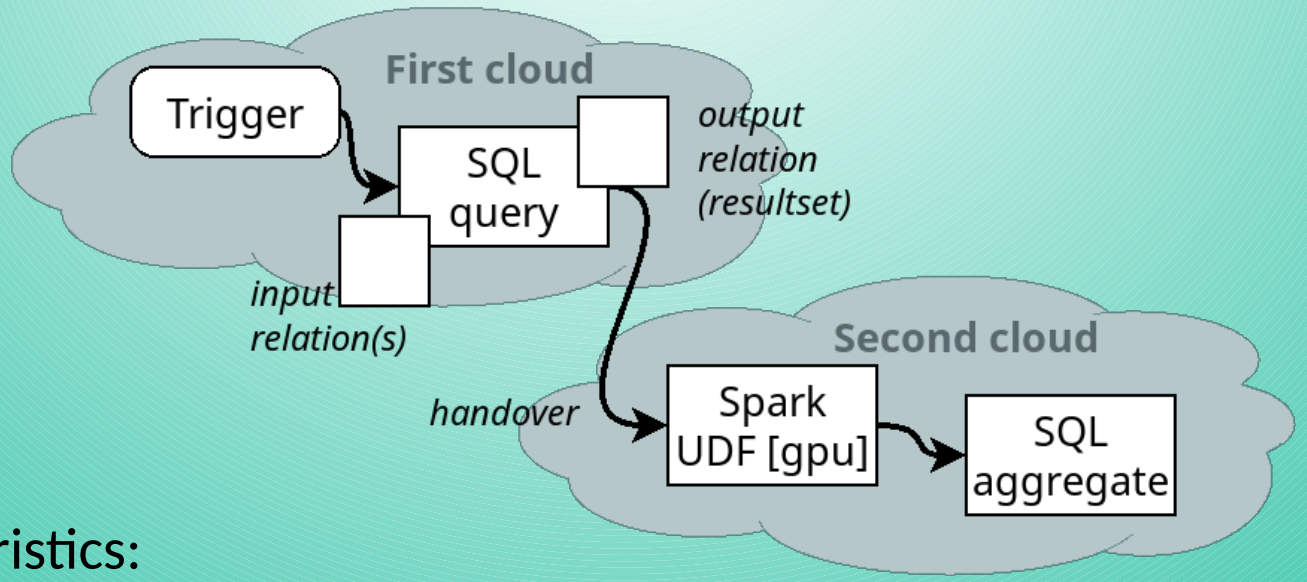
https://drive.switch.ch/index.php/s/AKDm4jBSegNTWur (demo video 3')

https://www.youtube.com/watch?v=YgRDskBv5V8 (related WOSCx2 talk)

CLOUD STARS

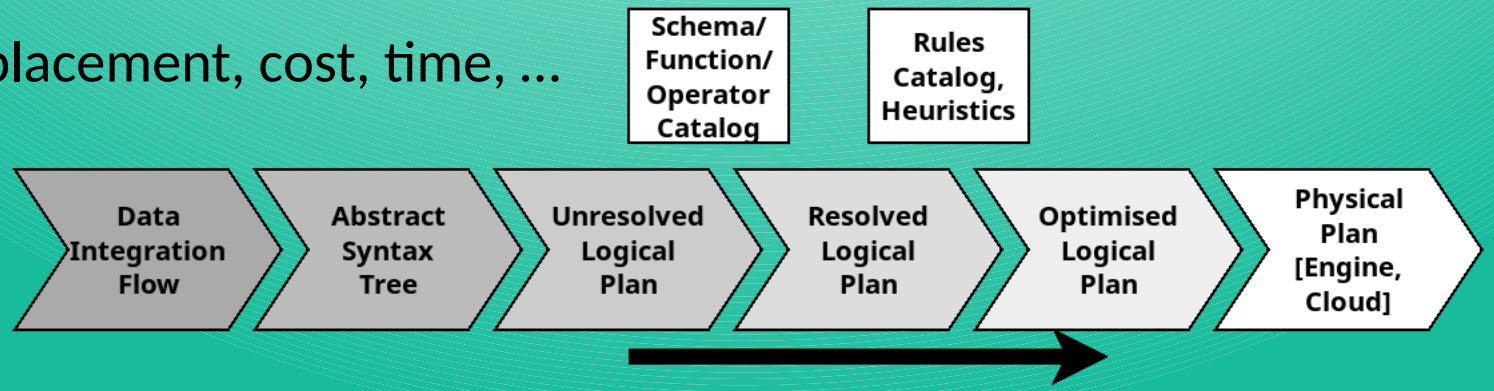
CLOUD OPEN SOURCE RESEARCH MOBILITY NETWORK

Hybrid Cloud Data Integration



Desired characteristics:
- Pipeline deadline (wall clock/SLAs)
- Cloud networking + input/output relations size
- Cloud compute + UDF complexity

→ Optimal placement, cost, time, ...



Apache Calcite

Calcite: dynamic data management framework

- SQL parser, query optimiser, DDL server support… → “RDBMS construction kit”
- Used by many projects (Flink, Storm, Beam…) + integrations

Dialect-specific Operators

The following operators are not in the SQL standard, and are not enabled in Calcite’s default operator table. They are only available for use in queries if your session has enabled an extra operator table.

To enable an operator table, set the fun connect string parameter.

The ‘C’ (compatibility) column contains value:

- ‘b’ for Google BigQuery (‘fun=bigquery’ in the connect string),
- ‘c’ for Apache Calcite (‘fun=calcite’ in the connect string),
- ‘h’ for Apache Hive (‘fun=hive’ in the connect string),
- ‘m’ for MySQL (‘fun=mysql’ in the connect string),
- ‘q’ for Microsoft SQL Server (‘fun=mssql’ in the connect string),
- ‘o’ for Oracle (‘fun=oracle’ in the connect string),
- ‘p’ for PostgreSQL (‘fun=postgresql’ in the connect string),
- ’s’ for Apache Spark (‘fun=spark’ in the connect string).

Non-scalar types

TYPE	DESCRIPTION	EXAMPLE LITERALS
ANY	The union of all types	
UNKNOWN	A value of an unknown type; used as a placeholder	
ROW	Row with 1 or more columns	Example: Row(f0 int null, f1 varchar)
MAP	Collection of keys mapped to values	
MULTISET	Unordered collection that may contain duplicates	Example: int multiset
ARRAY	Ordered, contiguous collection that may contain duplicates	Example: varchar(10) array
CURSOR	Cursor over the result of executing a query	

APACHE calcite™

HOME DOWNLOAD COMMUNITY DEVELOP NEWS DOCS

History

For a full list of releases, see github. Downloads are available on the downloads page.

1.36.0 / 2023-11-10

This release comes 3 months after 1.35.0, contains contributions from 30 contributors, and resolves 125 issues.

1.35.0 / 2023-07-26

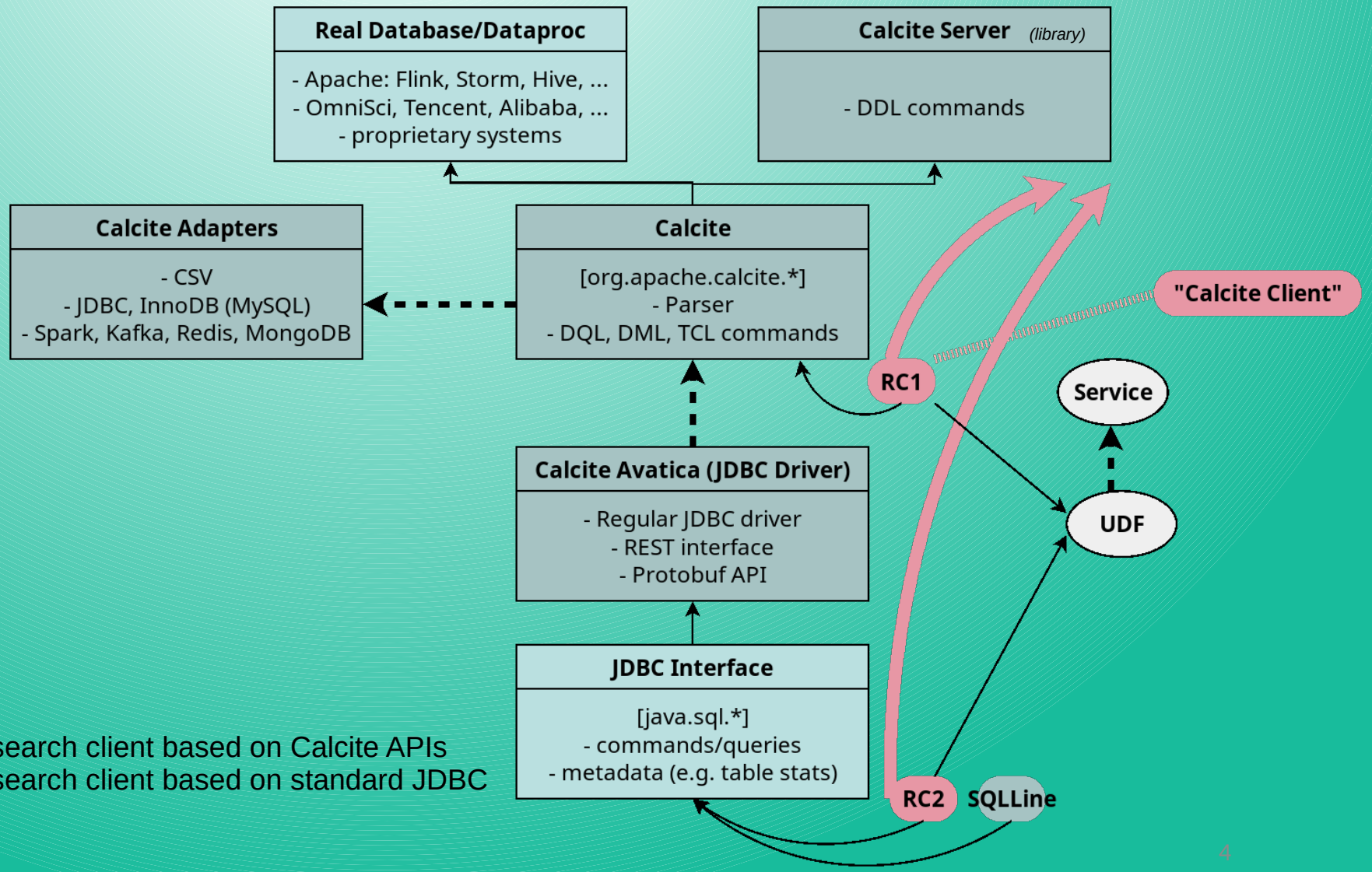
This release comes 4 months after 1.34.0, contains contributions from 36 contributors, and resolves 140 issues.

1.34.0 / 2023-03-14

This release comes 1 month after 1.33.0, contains contributions from 18 contributors, and resolves 34 issues. It’s worth highlighting the

Databaseless (in-memory) Calcite Client

	SQlite (`:memory:`)	Calcite
Working client	✓	
Programmability		✓



* RC1: research client based on Calcite APIs
* RC2: research client based on standard JDBC

Databaseless Calcite Client Paths

CalciteClient evolution and coverage

	Calcite API/Manual	Calcite API/DDL Node	JDBC API
CREATE TABLE (DDL)	works (custom parser!)	works (but requires ~1.34)	works (but only <= 1.23)
CREATE FUNCTION (DDL)	works (custom parser!)	works (but requires ~1.34)	"not supported" (as of 1.23)
INSERT (DML)	works	fails on mod.coll./ q-plan works/hack	works (but only <= 1.23)
SELECT (DQL)	works (even w/ cost metrics)		works

Databaseless Calcite Client Prototype

```
DB-less Calcite Client v0.0
Type .help to see available commands

::: .help

Internal commands:
.help    Show help text
.quit    Exit this program
.trace   Toggle command tracing: .trace on|off (default: on)
.metrics Toggle writing out metrics: .metrics on|off (default: on)
.rowlim  Toggle automatic limit on result rows: .rowlim on|off (default: off)
.iface   Choose parser interface: .iface jdbc|calcite; or show active with 'show' (default: calcite)
.cost    Choose cost estimator: .cost builtin|custom; or show active with 'show' (default: builtin)
.udf     Load UDF: .udf <classname> [<methodname> <sqlfuncname>]; or list with 'list'; or scan with 'scan'
.table   List tables with 'list'
.view    List views with 'list'
.ops     List available operators with 'list'

Batch syntax: calcite-client <query> [<iface>]
              calcite-client -h | --help

DB-less (in-memory) database commands: CREATE {TABLE,VIEW,FUNCTION}, DROP {TABLE,VIEW,FUNCTION}, INSERT, SELECT
::: .trace off

::: CREATE TABLE wosc9 (paperid int, speakername varchar(256));

SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
::: INSERT INTO wosc9 (paperid, speakername) VALUES (1, 'Mika Hautz');

::: INSERT INTO wosc9 (paperid, speakername) VALUES (2, 'Aitor Arjona');

::: SELECT COUNT(*) FROM wosc9;

2
::: 
```

Databaseless Calcite Client with UDFs

```
DB-less Calcite Client v0.0
Type .help to see available commands

::: .udf scan
plugins.RandomTable
plugins.RandomTable$1
plugins.RandomTable$2
plugins.LithopsFunction
plugins.ApproximateSum
plugins.SquareFunction
::: .udf plugins.SquareFunction eval SQUARE
Loaded: class plugins.SquareFunction
::: .udf plugins.LithopsFunction invoke FAAS
Loaded: class plugins.LithopsFunction
::: .udf plugins.ApproximateSum result ASUM
Loaded: class plugins.ApproximateSum
::: .udf plugins.RandomTable produce RANDTABLE
Loaded: class plugins.RandomTable
::: .udf list
Code: class plugins.SquareFunction.eval()      -> SQL:              SQUARE()
Code: class plugins.LithopsFunction.invoke()   -> SQL:                FAAS()
Code: class plugins.ApproximateSum.result()    -> SQL:                ASUM()
Code: class plugins.RandomTable.produce()      -> SQL:           RANDTABLE()
::: .table list
No tables created yet.
Table-producing function: RANDTABLE
```

```
::: .trace off
::: SELECT * FROM TABLE(RANDTABLE(3));
83
99
77
::: SELECT SQUARE(99);
9801
::: SELECT FAAS(99, 'x**3');
[lithops] generate code
[lithops] invoke executor
[lithops] return result
970299
::: .udf plugins.SquareFunction evaltwo SQUARETWO
::: SELECT SQUARETWO(SQUARE(99), FAAS(99, 'x+1'));
[lithops] generate code
[lithops] invoke executor
[lithops] return result
96069601
```

Databaseless Calcite Client Tracing

** programmatic equivalent to EXPLAIN PLAN FOR SELECT … - with rule-based heuristic Hep planner or cost-based Volcano planner*

```
::: SELECT * FROM wosc9;
Interpreting as SQL command...
---- operator table
313 operators
---- framework parser
SELECT *
FROM `WOSC9`
---- pre-validation
---- validation
SELECT `WOSC9`.`PAPERID`, `WOSC9`.`SPEAKERNAME`
FROM `WOSC9` AS `WOSC9`
---- roundtripping
SELECT `WOSC9`.`PAPERID`, `WOSC9`.`SPEAKERNAME`
FROM `WOSC9` AS `WOSC9`
SELECT "WOSC9"."PAPERID", "WOSC9"."SPEAKERNAME"
FROM "WOSC9" AS "WOSC9"
---- logical plan creation (rel stage)
LogicalProject(PAPERID=[$0], SPEAKERNAME=[$1]): rowcount = 100.0, cumulative cost = {200.0 rows, 301.0 cpu, 0.0 io}, id = 5
  LogicalTableScan(table=[[WOSC9]]): rowcount = 100.0, cumulative cost = {100.0 rows, 101.0 cpu, 0.0 io}, id = 4
JSON:{
  "rels": [
    {
      "id": "0",
      "relOp": "LogicalTableScan",
      "table": [
        "WO.....
Cost CPU: 301.0
Cost Total: {200.0 rows, 301.0 cpu, 0.0 io}
```

- ↯ Poor consideration of cardinality (rowcount)!
- ↯ No consideration of row data volume!
- ↯ No consideration of network traffic!
- ↯ No consideration of cloud egress cost!
- ↯ No consideration of cloud compute power!


```
"rels": [
  {
    "id": "0",
    "relOp": "LogicalTableScan",
    "table": [
      "WOSC9"
    ],
    "inputs": []
  },
  {
    "id": "1",
    "relOp": "LogicalProject",
    "fields": [
      "PAPERID",
      "SPEAKERNAME"
    ],
    "exprs": [
      {
        "input": 0,
        "name": "$0"
      },
      {
        "input": 1,
        "name": "$1"
      }
    ]
  }
]
```

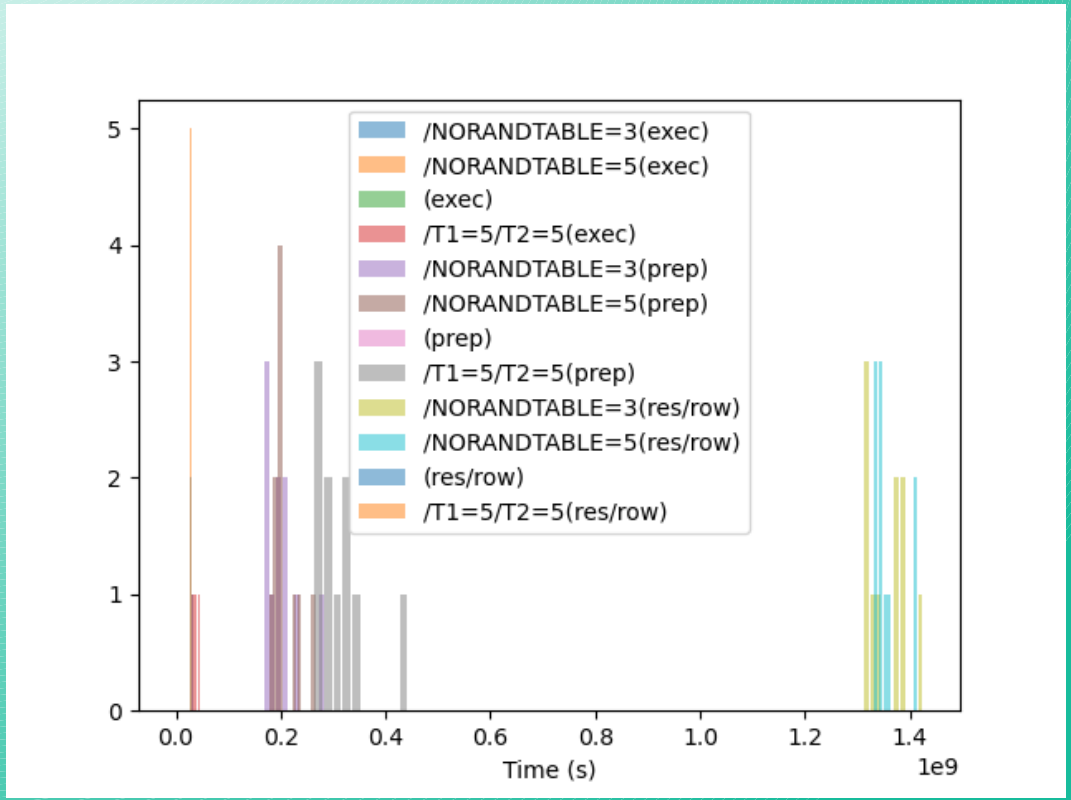
Query time prediction (architecture)

Client with additional research code

Measurement file (JSON)

Raw query
Query plan (non-canonical)
Prep/exec/fetch metrics
Result rows
Input relations* (all for now)

PREDICT SELECT*
script/service



** Proper PREDICT SQL extension + input relations checking would require more Calcite code…*

** x-fold reproduction of query times on identical hardware, software, input rels*

PREDICT parameterisation + ML approach

```
spio@tougener2:CalciteClient$ ./calcite-client
DB-less Calcite Client v0.0
Type .help to see available commands

::: .trace off
::: SELECT 2
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
2
::: PREDICT SELECT 2
*** Prediction based on 5 instances: times=0.085s prep/0.013s exec/0.0s results top10
::: PREDICT AND VERIFY SELECT 2
*** Prediction based on 5 instances: times=0.085s prep/0.013s exec/0.0s results top10
2
::: 
```

```
spio@tougener2:perf-udf$ python inoutcorrelation.py
For which of the following queries do you want to predict the runtime?
 (0) [splan:1726ec5d0330d6c1cea9ae47207d93fa07c006cbbb889fcc59a2e861] SELECT LSQUARE(rand, 'x+1') FROM norandtable
 (1) [splan:c08989fe5e57d31764b03ae609f452e2f31d70206783d2e558305fd8] SELECT 2
 (2) [splan:98612b20d57d26295fcd0f5d532bc103beaad19afed56a9a9983bdb5] SELECT t1.x + t2.y FROM t1, t2
Query number: 0
Random or manual input relation cardinalities (r = random, m = manual)? r
Verbose prediction (y/n))? n
Predicting results relation for {'NORANDTABLE': 504} ...
Results rows [504.]
Predicting processing time for results rows 504 ...
Time will be ~ 684 s
```

Training set generator

```
[
    {
        "t1": {"x": "int"},
        "t2": {"y": "int", "z": "il:plate"},
        "query": "SELECT t1.x + t2.y FROM t1, t2"
    }
]
```

```
spio@tougener2:generator$ python gen.py
CREATE TABLE t1 (x int);
INSERT INTO t1 (x) VALUES (18);
INSERT INTO t1 (x) VALUES (8);
INSERT INTO t1 (x) VALUES (98);
INSERT INTO t1 (x) VALUES (93);
CREATE TABLE t2 (y int, z varchar(128));
INSERT INTO t2 (y, z) VALUES (24, '258-62-707');
INSERT INTO t2 (y, z) VALUES (73, '322-83-776');
INSERT INTO t2 (y, z) VALUES (62, '963-86-327');
INSERT INTO t2 (y, z) VALUES (43, '897-59-737');
INSERT INTO t2 (y, z) VALUES (63, '528-98-314');
INSERT INTO t2 (y, z) VALUES (16, '981-33-740');
INSERT INTO t2 (y, z) VALUES (51, '007-01-271');
SELECT t1.x + t2.y FROM t1, t2;
```

```
spio@tougener2:generator$ python gen.py | sqlite3
129
58
50
147
76
68
```

```
spio@tougener2:CalciteClient$ (echo ".trace off"; python ../../perf-udf/generator/gen.py) | ./calcite-client
DB-less Calcite Client v0.0
Type .help to see available commands

::: ::: CREATE TABLE t1 (x int)SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
::: INSERT INTO t1 (x) VALUES (60)|||||||| VALUES ROW(60)
|||||||| = 60 @row 0
=>MADD [Ljava.lang.Object;@5d8445d7
::: CREATE TABLE t2 (y int, z varchar(128)):::: INSERT INTO t2 (y, z) VALUES (47, '777-44-668')|||||||| VALUES
|||||||| = 47, '777-44-668' @row 0
=>MADD [Ljava.lang.Object;@615f972
```

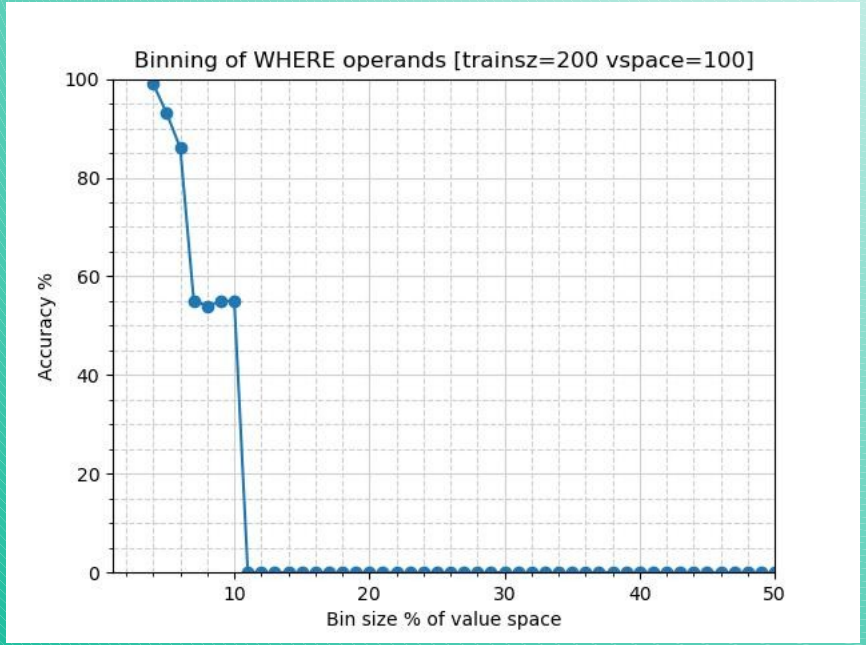
Training set generator

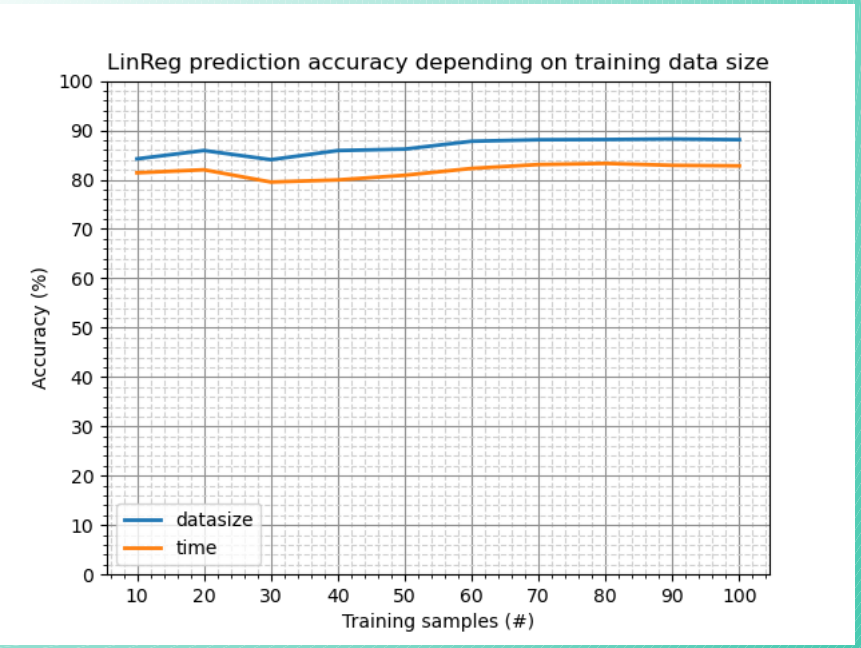
```
cd ../calcite-client/CalciteClient/
rm -rf measurements

for i in `seq 1 10`
do
    (echo ".trace off"; python ../../perf-udf/generator/gen.py --maxrows 100) | ./calcite-client
done
```

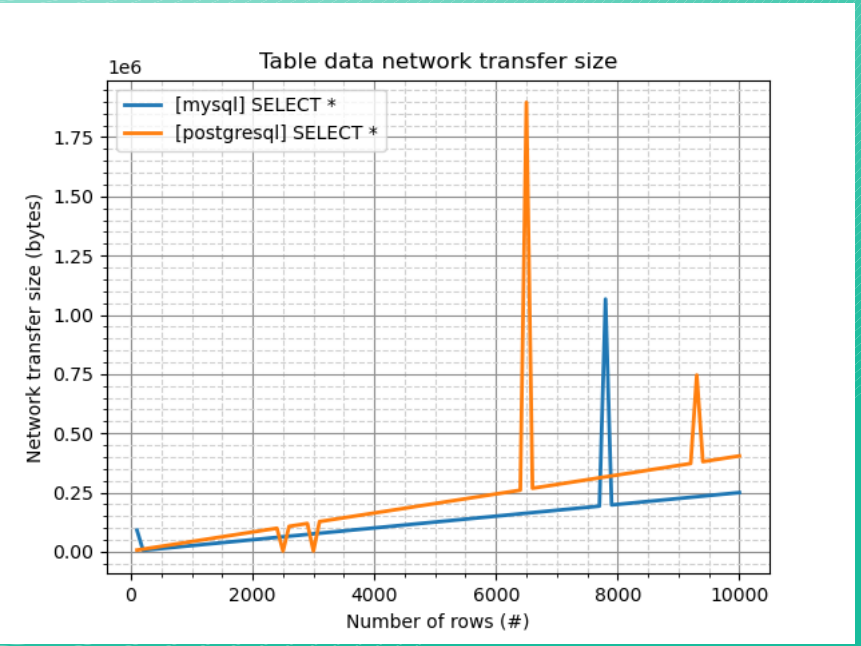
```
[
    {
        "t": {"s": "str"},
        "query": "SELECT CHAR_LENGTH(s) FROM t"
    },
    {
        "t1": {"x": "int"},
        "t2": {"y": "int", "z": "il:plate"},
        "query": "SELECT t1.x + t2.y FROM t1, t2"
    },
    {
        "nrtable": {"rand": "int"},
        "query": "CREATE FUNCTION LSQUARE AS 'plugins.LithopsFunction.invoke'; SELECT LSQUARE(rand, 'x+1') FROM nrtable"
    }
]
```

```
spio@tougener2:perf-udf$ python clusteranalysis.py; python inoutcorrelation.py
* 98612b20d57d26295fcd0f5d532bc103beaad19afed56a9a9983bdb5 x 10 =~ avg.   0.1s
* 2c7fc64cd408bcdb88e64c571c0836aca43b7f46f8d7d350d8a56931 x 10 =~ avg.  67.8s
* 3520474f2c3056cb6eda0abd0f02ce81887a6dedbdc091982de6a898 x 10 =~ avg.   0.3s
For which of the following queries do you want to predict the runtime?
 (0) [splan:98612b20d57d26295fcd0f5d532bc103beaad19afed56a9a9983bdb5] SELECT t1.x + t2.y FROM t1, t2
 (1) [splan:2c7fc64cd408bcdb88e64c571c0836aca43b7f46f8d7d350d8a56931] SELECT LSQUARE(rand, 'x+1') FROM nrtable
 (2) [splan:3520474f2c3056cb6eda0abd0f02ce81887a6dedbdc091982de6a898] SELECT CHAR_LENGTH(s) FROM t
Query number:
```

Performance + Traffic







** SELECT … WHERE a < X AND b < Y …*
0 <= X/Y <= 100

CLOUD STARS

Thank you

UNIVERSITAT ROVIRA i VIRGILI
BSC
UNIVERSIDAD DE MURCIA
Technische Universität München TUM
Julius-Maximilians-UNIVERSITÄT WÜRZBURG
VU
AGH
UNIVERSITÀ DI TRENTO
TU WIEN
IBM
NEARBY COMPUTING
inesc id lisboa
zh aw
Università della Svizzera italiana
Imperial College London

BDV BIG DATA VALUE ASSOCIATION

Funded by the European Union

Partially funded by Switzerland