OMNeT++ Community Summit 2016, Brno University of Technology (FIT-BUT), Sept 15-16.

SQLite as a Result File Format in OMNeT++

Rudolf Hornig

OMNeT++ Result Files

- Scalar and Vector files
- Contents:
  - Run description
  - Scalar file: (module, scalar, value), histograms, result attributes
  - Vector file: vector data: (module, vectorname, vector data = (timestamp+value)*)
- Current format:
  - Line-oriented text file, human-readable, easy to parse
  - Tools:
    - Analysis Tool in the IDE: charting, export
    - Scavetool: export into CSV and other formats (CSV can be imported into spreadsheets and other tools)
    - R plugin (GNU R is a language and environment for statistical computing)

Pros and Cons of the Current Format

Pros:

- Human readable
- Easy to parse with command-line tools

Cons:

- Hard to use directly with third party tools
- Scalability issues when a lot of scalars are generated
- Hard to filter out the unnecessary scalars

Alternative Format: SQLite

- SQLite: embedded, low-resource database engine
  - Database is a local file
  - Engine is a single C file (easy to add into existing programs)
  - Capable SQL support
  - Robust and proven (used inside Android, Firefox, etc.)
  - Command-line SQL console (with CSV export support)
  - wealth of GUI tools
  - Great integration with third-party tools. Can be used from Python, R and other statistical tools.

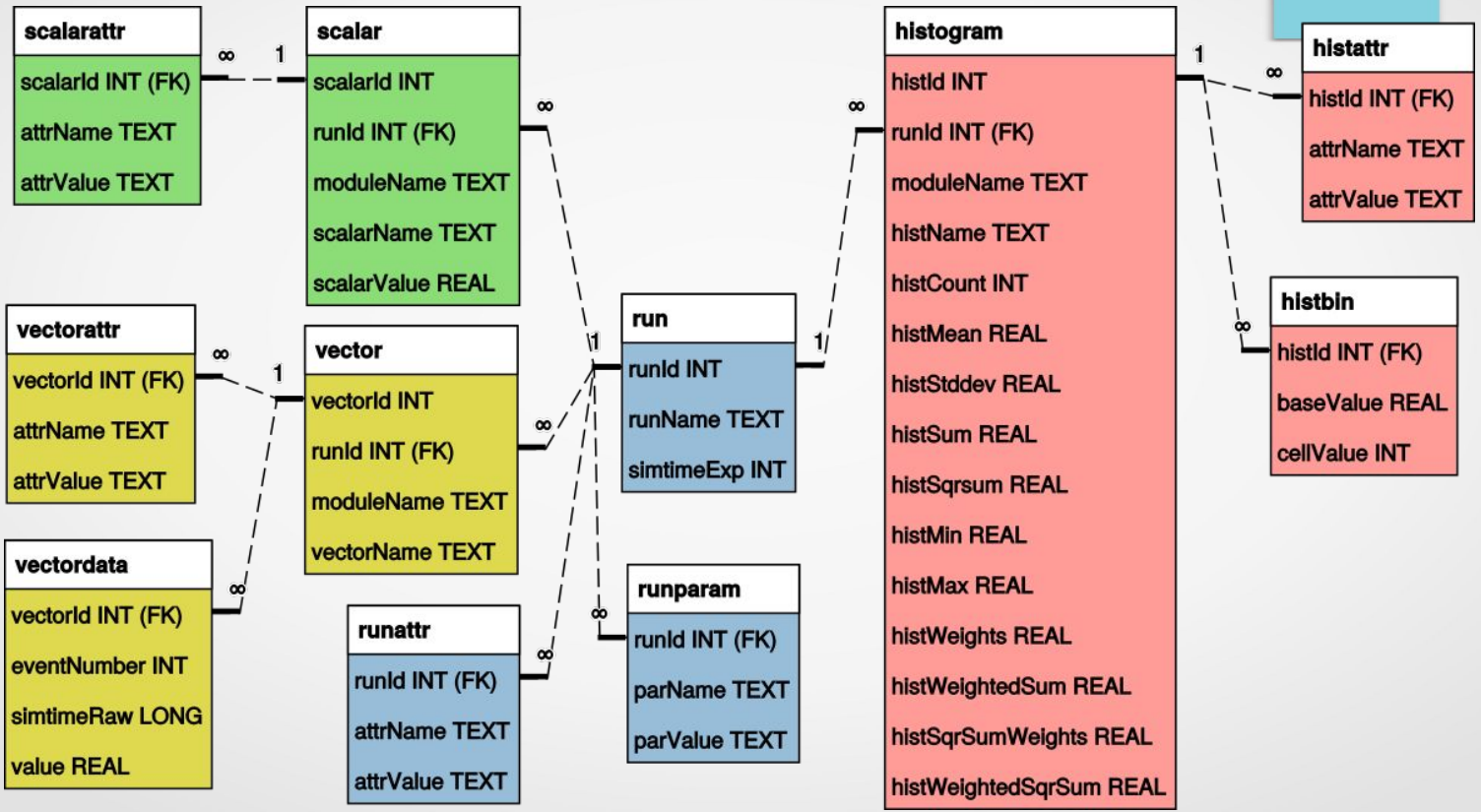
File Format Comparison (text based)

```
version 2
run PureAlohaExperiment-12-20160902-11:36:24-19332
attr configname PureAlohaExperiment
attr mean 9
attr numHosts 10
attr repetition 0
param Aloha.host[*].iaTime "exponential(${mean=1,2,3,4,5..9 step 2}s)"

scalar Aloha.server   duration     5400
scalar Aloha.server  collisionLength:mean 0.16657246074119
scalar Aloha.server  channelUtilization:last 0.18432244370657

statistic Aloha.server    collisionLength:histogram
field count 508
field mean 0.15209864334356
```

File Format Comparison (SQLite)



Implementation

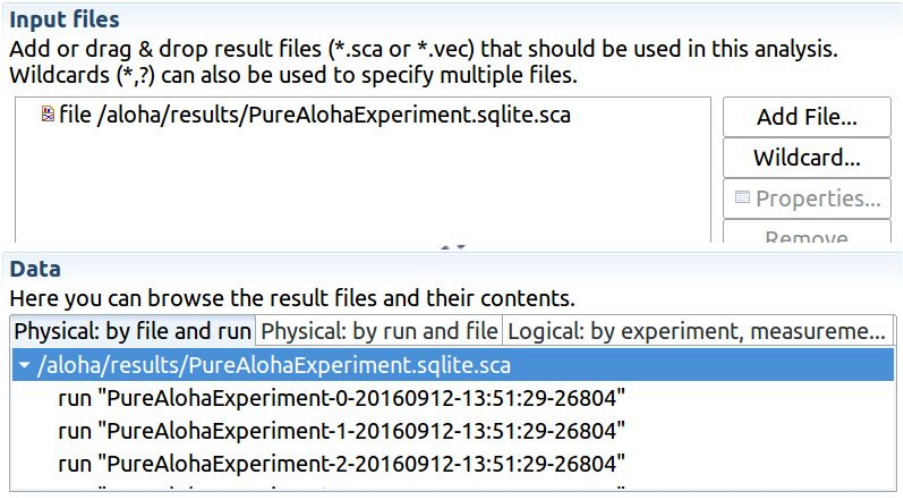
- Experimental implementation exists in the Technology Preview
- Extension classes that plug into envir
  - Can be selected from ini file, no other changes required
- Scavetool recognizes both text-based and SQLite files, all functionality is available for both formats
- IDE Analysis Tool relies on scave library, so it understands both formats

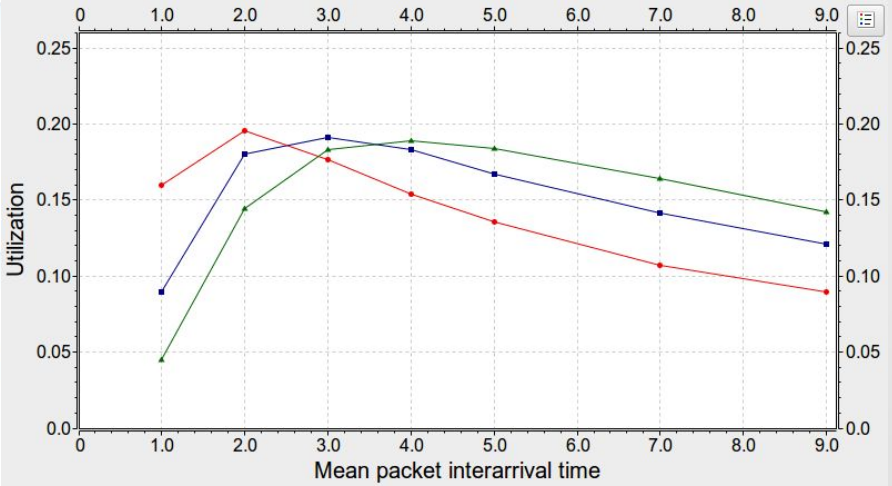
Example configuration:

```
outputscalarmanager-class = "omnetpp::envir::cSqliteOutputScalarManager"
outputvectormanager-class = "omnetpp::envir::cSqliteOutputVectorManager"
```

Using SQLite Result Files from OMNeT++ IDE

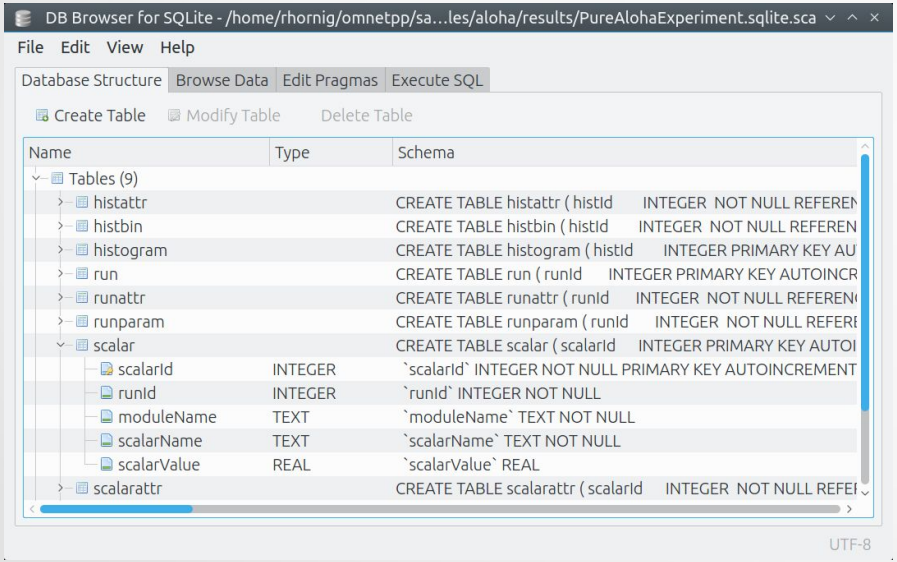
SQLite Result Files can be used from the IDE just like the text based format.

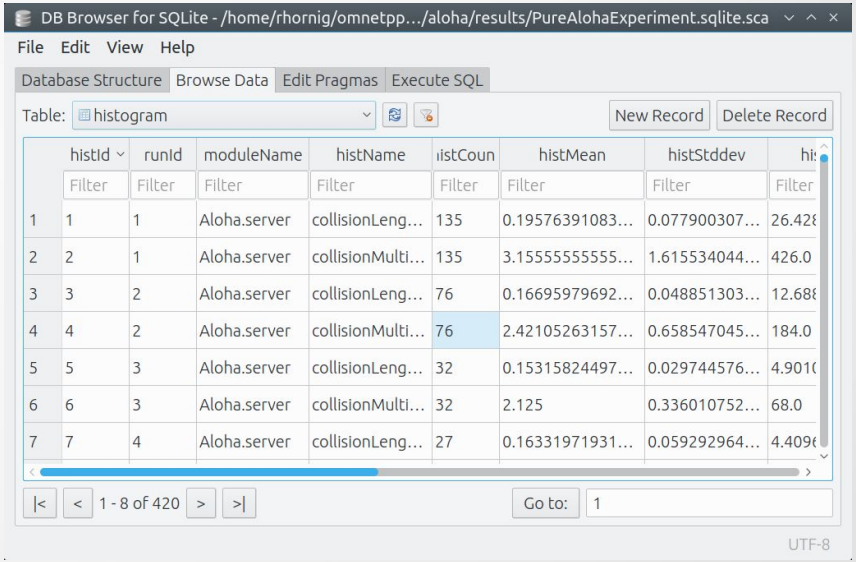




Using SQLite Result Files Directly

There are several GUI tools to browse and process SQLite files: SQLite Browser, SQLiteman, ...

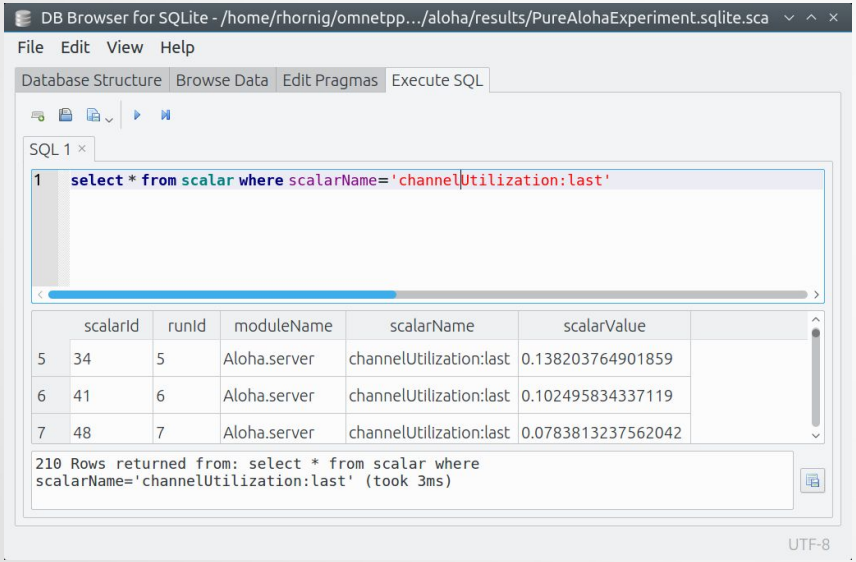


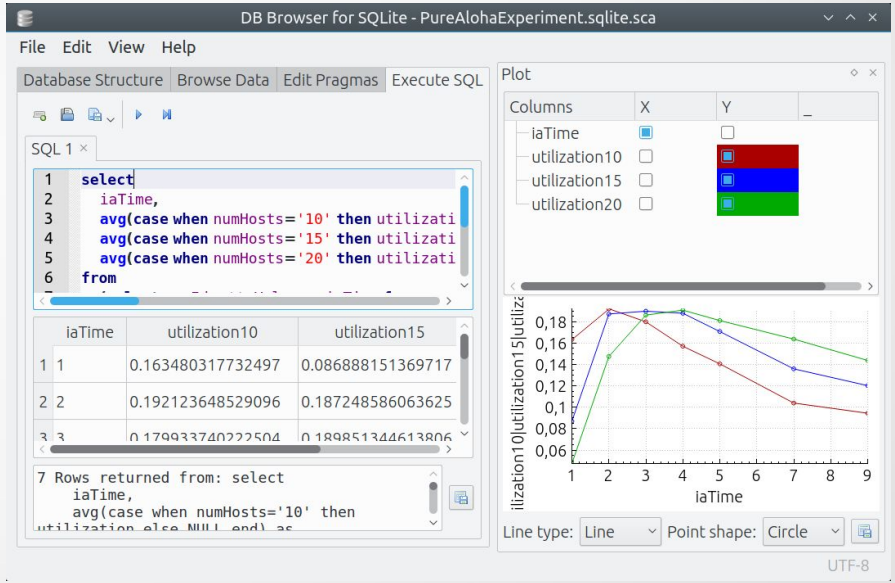


You can use SQL to select and organize the data you need. Everyone knows SQL, right?

Using SQLite Result Files Directly 2

Filtering and some basic statistical functions are directly available in SQL...





… and some tools even support basic charting.

Using SQLite Result File with CSV export

Scave Tool can export in CSV which can be further processed with other 3rd party tools like Libre Office Calc / Google Sheet Pivot Table or other statistical tools.

```
scavetool scalar -g name -F csv -O result.csv PureAlohaExperiment.sqlite.sca
```

This can be imported into a table which is further used as the source data for a pivot table.

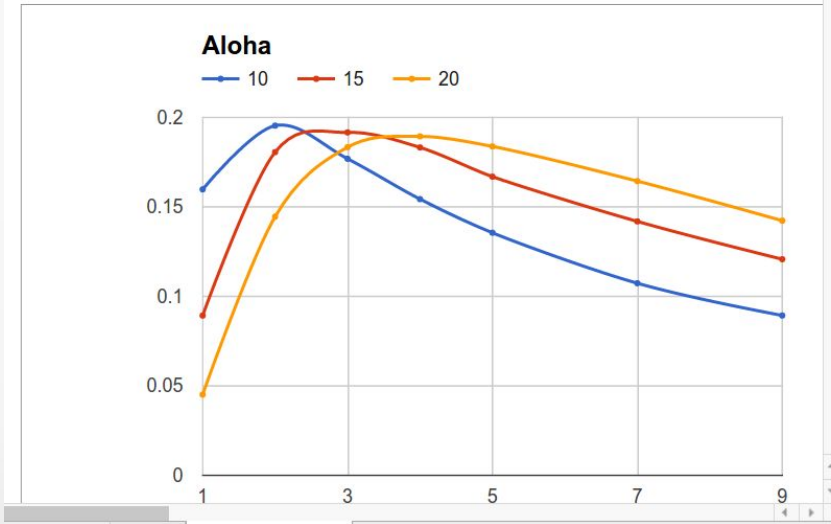
NOTE: You can export the SQLite database directly by using the `sqlite3` command.

```
sqlite3 -csv PureAlohaExperiment.sqlite.sca \
    'select * from scalar' >result2.csv
```

Using SQLite Result File with CSV export 2

CSV files can be loaded into spreadsheets to create Pivot Charts

	1	2	3	4	5	7
10	0.1597615	0.195359	0.1768435	0.154311	0.1355175	0.1073435
15	0.08929915	0.1805725	0.191573	0.1831895	0.166857	0.1418655
20	0.0451497	0.144524	0.183319	0.1893645	0.183734	0.164381



Rows - Add field

Group by: runattr:numHosts

Order: Ascending

Sort by: runattr:numHosts

Show totals

Columns - Add field

Group by: runattr:mean

Order: Ascending

Sort by: runattr:mean

Show totals

Values - Add field

Display: channelUtilization:last

out | Pivot Table 1

Using SQLite Result File from Python

Required packages

- Sqlite3
  - Access SQLite databases
- NumPy/SciPy
  - Numerical scientific computing
- MatPlotLib
  - Comprehensive 2D plotting

Using SQLite Result File from Python 2

How to access data from Python.

```
import sqlite3


conn = sqlite3.connect(fileName)
conn.row_factory = sqlite3.Row
cur = conn.cursor()
sql = "select numHosts, iaTime, avg(utilization) as utilization from ..."
cur.execute(sql)
rows = cur.fetchall()
numHosts = [row["numHosts"] for row in rows]
```

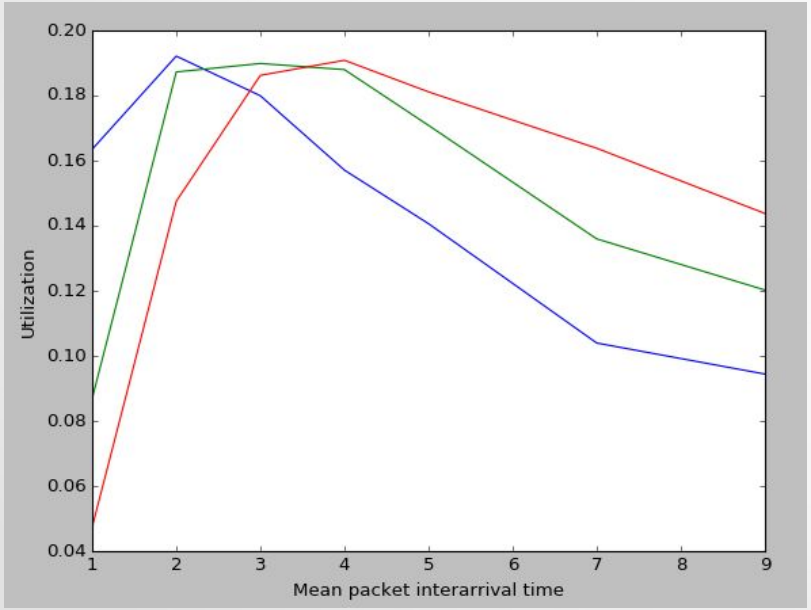
Using SQLite Result File from Python 3

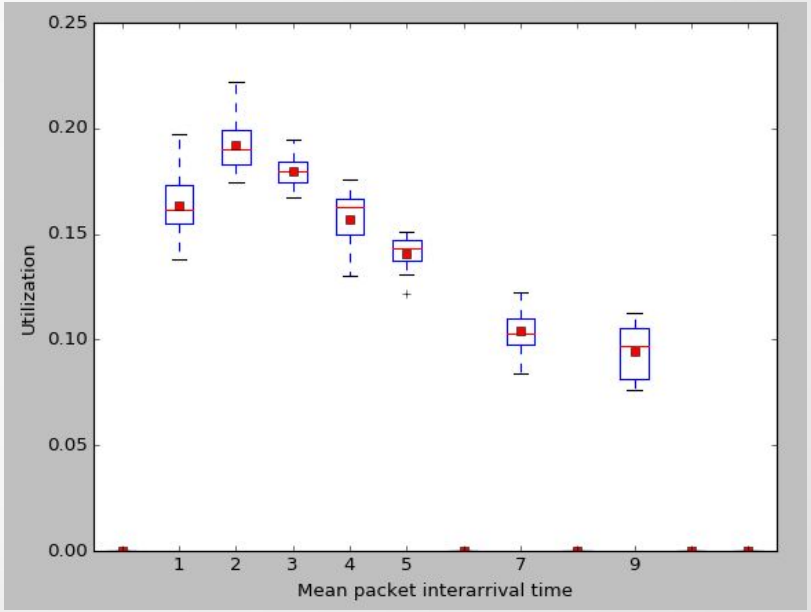
Plot the same chart in Python:

```
fig1 = plt.figure()
ax1 = fig1.add_subplot(111)
nh = sorted(list(set([row["numHosts"] for row in rows])))
for n in nh:
    x = [row["iaTime"] for row in rows if row['numhosts']==n]
    y = [row["utilization"] for row in rows if row['numhosts']==n]
    ax1.plot(x, y, '-')
ax1.set_xlabel('Mean packet interarrival time')
ax1.set_ylabel('Utilization')
```

Using SQLite Result File from Python 3

Charts rendered from the PureAlohaData in Python:





Other Useful Python Libraries

- Pandas is a really nice library for working with statistical data -- tabular data, time series, panel data. Includes many builtin functions for data summaries, grouping/aggregation, pivoting. Also has a statistics/econometrics library.
- Larry provides labeled arrays that play nice with NumPy. Provides statistical functions not present in NumPy and good for data manipulation.
- Python-statlib is a fairly recent effort which combined a number of scattered statistics libraries. Useful for basic and descriptive statistics if you're not using NumPy or pandas.
- Statsmodels helps with tatistical modeling: Linear models, GLMs, among others.
- Scikits is a statistical and scientific computing package -- notably smoothing, optimization and machine learning.
- PyMC is for your Bayesian/MCMC/hierarchical modeling needs.
- PyMix for mixture models
- If speed becomes a problem, consider Theano. Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.

Python vs R?

- **Case for R**: wealth of statistical packages available in R/CRAN
  - Drawback: special purpose programming language, unsuitable outside statistics
- **Case for Python**: General-purpose, hugely popular programming language with an extensive set of libraries; emerging as integration platform and preferred programming environment for many scientists
  - Drawback: statistical functionality is limited compared to R, but satisfactory for our purposes

Comparison: SQLite vs Text

- File size: about the same or a bit smaller than the text based format
- Vector Recording performance: 2-2.5x slower (net writing speed)
- Read performance: Depends on the complexity of the query, but it can be optimized by adding indexes after recording the database
- Current optimizations employed
  - Vectors are written in batches
  - Bathces are committed in separate transaction
  - Pragma synchronize = off

SQLite or not SQLite?

Perceived advantages

- More accessible: Browse and query with standard tools, using a standard language (SQL)
  - e.g. text-based result file format required special “omnetpp” R-plugin to get the data inside R. (R already has SQLite access library)

Possible drawbacks:

- Speed
  - vector recording performance is about 2-2.5x slower than text-based file format
  - in actual simulations, our experience with INET simulations has shown about 25% slowdown if ALL possible vectors are recorded (which is not common)

Try it

Available in Technical Preview (see Aloha example)

Feedback is needed...