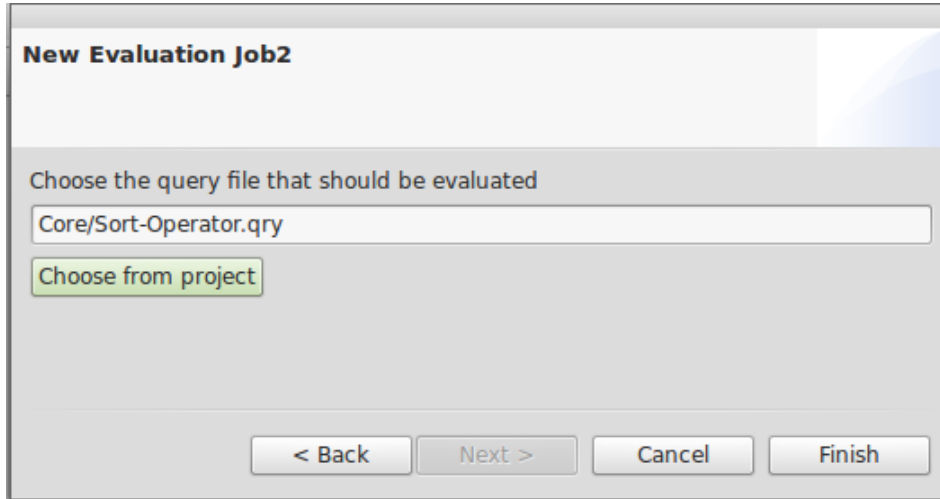


Evaluation Feature

The evaluation feature offers some graphical user interface and instrumentation to measure the latency and throughput of a given query and to create some plots.

Create a new evaluation



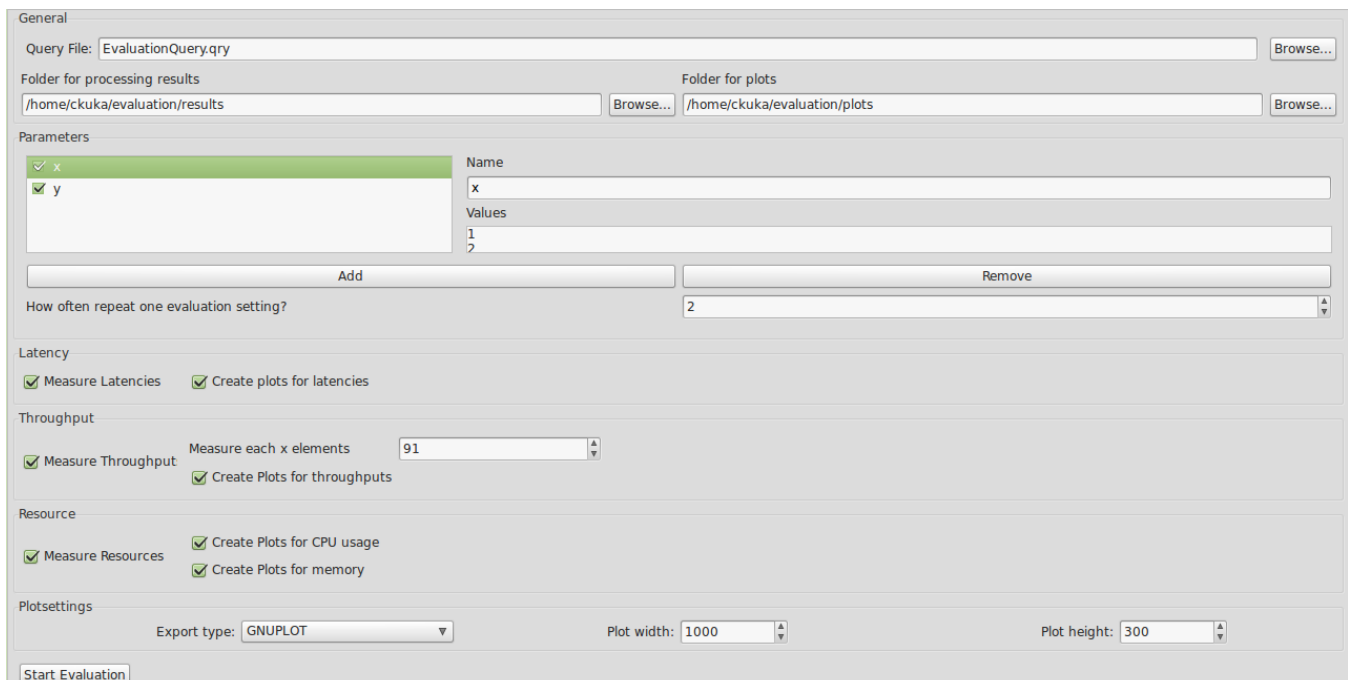
The dialog box titled "New Evaluation Job2" contains a text field with the value "Core/Sort-Operator.qry". Below the text field is a button labeled "Choose from project". At the bottom of the dialog are four buttons: "< Back", "Next >", "Cancel", and "Finish".

Before the evaluation feature can be used the features `rcp.evaluation.feature` and `latency` have to be added to the Odysseus product.

An evaluation is maintained by an evaluation job, which encapsulates all necessary settings. A evaluation job is project relative so that it must be assigned to an Odysseus project.

1. To create a new evaluation, go to an Odysseus project and use right click, browse to "new" and choose "Evaluation Job".
2. Choose a project wide unique name for your evaluation job and press next.
3. Choose the Odysseus Script file from the project you want to evaluate.
4. Now you should see the evaluation job in your editor view. You may change the settings, which are described in the following.

Settings for an evaluation



The "General" settings tab of the evaluation job configuration dialog. It includes fields for "Query File" (EvaluationQuery.qry), "Folder for processing results" (/home/ckuka/evaluation/results), and "Folder for plots" (/home/ckuka/evaluation/plots). A "Parameters" table lists parameters x and y with their respective values. Below the table are "Add" and "Remove" buttons. The "How often repeat one evaluation setting?" is set to 2. The "Latency" section has checkboxes for "Measure Latencies" and "Create plots for latencies". The "Throughput" section has checkboxes for "Measure Throughput" and "Create Plots for throughputs", with a "Measure each x elements" field set to 91. The "Resource" section has checkboxes for "Measure Resources", "Create Plots for CPU usage", and "Create Plots for memory". The "Plotsettings" section includes "Export type" (GNUPLOT), "Plot width" (1000), and "Plot height" (300). A "Start Evaluation" button is at the bottom.

Parameters	Name	Values
<input checked="" type="checkbox"/> x	x	1
<input checked="" type="checkbox"/> y	y	2

Query File: This is the query file you want to evaluate. It should deliver at least one executable, physical query plan. Remember that the query will be executed more than once, so you should design the script to be re-executable. This means for example, the query file may not try to install new sources which are already installed. Hint: you may use something like "#DROPALLSOURCES" or "#IFSRCDEF/#IFSRCNDEF" in [Odysseus Script](#).

New (2017.02.16): There is now the possibility to add a **Setup File and a Tear Down file**. Similar to JUnit these scripts are called directly before and after the run (e.g. to register sources). These scripts can ran before and after each run or onyl once before and after the whole runs (**Setup/Tear down every Run**).

Folder for processing results: This is a the folder where the evaluation dumps the results. For each new run of an evaluation, a new subfolder (named by date and time) is created.

Folder for plots: If you want to create plots from the results, the created plots are saved here.

Parameters: These parameters can be used to iterate over certain settings (or values) in your query file. If you created a new evaluation job, you have two predefined examples: "var_1" and "var_2". "var_1" is the name of the variable and the values are given at the right (each line is a value). So the predefined parameter "var_1" has the values "a", "b" and "c" and "var_2" the values "x", "y", and "z". If the parameters are active (the checkbox is checked), the evaluation will iterate over the values. It will produce each possible combination of all active parameters. For this example there will be 9 different runs:

1. var_1=a and var_2=x
2. var_1=a and var_2=y
3. var_1=a and var_2=z
4. var_1=b and var_2=x
5. var_1=b and var_2=y
6. var_1=b and var_2=z
7. var_1=c and var_2=x
8. var_1=c and var_2=y
9. var_1=c and var_2=z

At each run the variables are used to replaced defined variables in the query file. So, if we have a variable "\${var_1}" in the query file, it is replaced with "a", "b" and "c". So, the parameters allows you to change the query file and to automatically run different evaluations with different settings for the same query file.

For each setting, the results are stored within a certain folder which has the form "value1_value2_value3_...". For example, for the first setting this will be "a_x", the second "a_y" and so on.

You can add a new parameter using the "Add" button and remove a parameter by selecting the parameter and using the "Remove" button.

How often to repeat: This is the number of times you want to repeat one certain setting. This allows to rerun the same setting multiple times to remove some inaccuracies in the measurements. At the end of the evaluation, all runs of the same settings are merged into one result. So, for the example with the parameters "var_1" and "var_2" above with a repeat of "10", you will have 90 evaluations (10 for each setting). For each setting (e.g. var_1=a and var_2=x) there will be an own folder within the setting folder (here: "a_x/1", "a_x/2", ..., "a_x/10". at the end, the 10 results are merged into one single result and will be put in a folder called "merged" (which leads into "a_x/merged").

Measure Latency: If it is activated, the evaluation job adds the his pre-transformation handler to measure the latency of the query. The pre-transformation handler changes the query files (e.g. it activates the metadata latency by adding "#METADATA Latency" to the query. Furthermore, it adds the LatencyCalculationPipe and a FileSink to each root operator of the query. Therefore, if your query has different sinks, you will also have different latency results. **TODO: This has changed. Currently your sources must contain valid metadata for processing!!**

Create plots for latencies: If activated, the evaluation will use the merged files to create some plots, which are stored within the plot-folder (look above)

Measure Throughput: If this is active, the evaluation job adds his pre-transformation handler to measure the throughput of a query. The pre-transformation handler changes the query by adding a throughput operator after each source.

Measure each x elements: This is the number of elements between a throughput measurement. This means: the time for a certain amount x of elements will be measured. So, if x is 100, the system will check how long it will take to process 100 element, then the next 100 elements, the next 100 elements and so on.

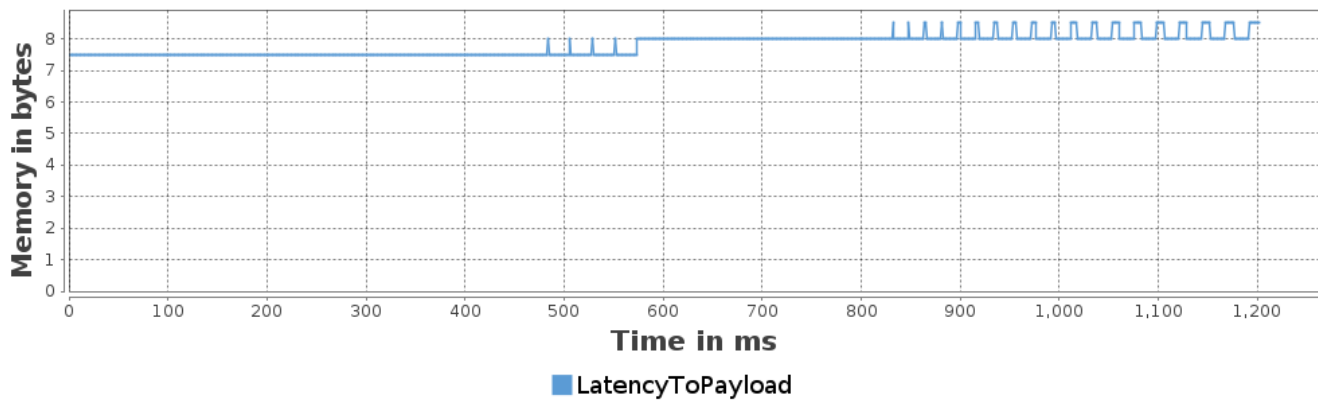
Create plots for throughput: If activated, the evaluation will use the merged files to create some plots, which are stored within the plot-folder (look above).

Measure Resources: If this is active, the evaluation job adds his pre-transformation handler to measure the CPU and memory consumption. The pre-transformation handler changes the query by adding the "#METADATA Systemload" directive to the query. To measure the systemload, the "Systemload Metadata Feature" is required.

Create plots for CPU: If activated, the evaluation will use the merged files to create a plot of the CPU consumption.

Create plots for memory: If activated, the evaluation will use the merged files to create a plot of the memory consumption.

Plotsettings: You can choose the export type: PDF, PNG, JPEG, or GNUPLOT. Furthermore you may define the dimension of the plot, its height and width.



If the export type is PDF, PNG, or JPEG the resulting plot will look like the figure above. If the export type is chosen to be GNUPLOT, the evaluation feature will generate a .dat file with the merged measurements and a .gnu file containing a [Gnuplot](#) script to generate a LaTeX file.

Run and stop the evaluation

You can simply start the evaluation by hitting "Start Evaluation".

To stop an evaluation you may click "Cancel" on the progressmonitor.

If you stop a query by hand (e.g. using the Queries view) - one evaluation run is canceled and the next run will be installed and started automatically.