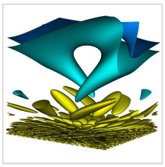

Mar 01, 2021

Prepare and run test on available dataset

Stepan Orlov[1], Alexey Zhuravlev[1], Егор Усик[1], Alexey Kuzin[1]

[1]Peter the Great St.Petersburg Polytechnic University

1 *Works for me* dx.doi.org/10.17504/protocols.io.bruzm6x6

Alexey Zhuravlev

SUBMIT TO PLOS ONE

ABSTRACT

This protocol shows how to perform tests whose results are presented in the manuscript dedicated to the ReVisE open source visualization system. The system implements interactive visualization of large datasets hosted on a remote server or a supercomputer. The protocol also refers to another protocol that describes how to build and install the ReVisE system; that step should be taken if ReVisE has not been built yet on the server machine. The entire process, as described in the protocols, has been tested on Ubuntu 18.04 and Ubuntu 20.04 Linux distributions.

EXTERNAL LINK

https://github.com/deadmorous/revise

DOI

dx.doi.org/10.17504/protocols.io.bruzm6x6

EXTERNAL LINK

https://github.com/deadmorous/revise

PROTOCOL CITATION

Stepan Orlov, Alexey Zhuravlev, Егор Усик, Alexey Kuzin 2021. Prepare and run test on available dataset. **protocols.io**
https://dx.doi.org/10.17504/protocols.io.bruzm6x6

KEYWORDS

ReVisE, visualization system

LICENSE

This is an open access protocol distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

CREATED

Jan 27, 2021

LAST MODIFIED

Mar 01, 2021

PROTOCOL INTEGER ID

46713

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

GUIDELINES

To run the protocol, one needs the access to a visualization server - a computer running the Linux operating system and equipped with a recent NVIDIA's GPU, and a client machine (which might in some cases be the same computer as the visualization server) with a display at FullHD resolution (1920 x 1080 pixels). See the Materials section.

The protocol contains two branches (see step 4). On the one hand, there is a detailed step-by-step instruction, requiring user to copy and paste many commands to Linux terminal; if a command fails, it is clear where specifically that happens, therefore we find it useful to have such a detailed description as a branch of the protocol. On the other hand, when one finds that all steps succeed, a more convenient way of test execution is available. The other branch explains how to use an interactive script that executes all necessary steps and instructs user about necessary actions.

In our paper, we consider three datasets, identified as **hump**, **cascade**, and **mwave**. Since performing the tests with any of those datasets takes considerable time required for downloading and/or preprocessing, there is another very small dataset, **sphere1M** (case matters!), that might be considered as a starting point due to its small size and fast execution. Once **sphere1M** appears to be working, other datasets can be tested.

MATERIALS TEXT

Visualization server

A computer (single node) equipped with one or more NVIDIA's GPU is required to run this protocol. The model of GPU may vary from GeForce 1060 to V100 or more recent.
The computer is expected to run the Linux operating system.
Minimum RAM requirement is 16 GB; one of the test requires 128 GB RAM.
Free disk storage requirement is 210 GB to store all source and preprocessed datasets at once.
The computer is expected to have an Internet connection for downloading ReVisE, any of necessary prerequisites to build it, and datasets.
If accessed remotely, the ssh connection is expected to work
Super-user privileges might be necessary in order to install ReVisE prerequisites.

Client machine

The client machine is expected to be a computer (desktop or laptop) having a display with FullHD resolution (1920 x 1080 pixels).
The client machine might be the same as the visualization server, or it can be a different computer connected to the visualization server remotely, via ssh.

BEFORE STARTING

The protocol assumes that the client machine is running Linux. however, this is not necessary if the client machine is used to access a remote visualization server. If a Windows machine is used, install PuTTY for establishing the ssh connection with the remote server.

Prepare to measure ReVisE visualization performance

1 Open Linux terminal.

If ReVisE is to be run on a remote machine, connect to the machine using ssh. In that case, make sure to use local forwarding since the ports 1234 and 3000 are needed for interacting with the ReVisE web server.

```
ssh -L 1234:localhost:1234 -L 3000:localhost:3000 username@hostname
```

2 Download and build the ReVisE system if it is not built.

Build ReVisE on Ubuntu
by Alexey Zhuravlev
PREVIEW RUN

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

Skip this step if ReVisE has already been built on the target machine.

2.1 Open Linux terminal. To install ReVisE on a remote machine, connect to it with ssh. To do so type the following command:

```
ssh username@hostname
```

Skip this step if Linux terminal is already opened and the ssh connection to the remote machine is established.

2.2 Make sure the following ReVisE prerequisites are installed

- cmake >= 3.18.1
- git
- clang >= 9.0 / gcc >= 7.4
- Qt >= 5.10 including QtSVG, QtWidgets
- Node.js
- npm
- D language
- CUDA Toolkit >= 11.1
- C++ Boost libraries

2.3 Go to the directory where the ReVisE repository will be downloaded and built.

```
cd <source directory of your choice>
```

2.4 Clone ReVisE. 1m

```
git clone https://github.com/deadmorous/revise.git
```

2.5 Go to repository directory and set *REVISE_ROOT_DIR* environment variable to the path where ReVisE was cloned.

```
cd revise
REVISE_ROOT_DIR=$PWD
```

2.6 Optionally set up Qt, and probably some library paths, e.g.

```
export QT_SELECT=qt5-11-2
export LD_LIBRARY_PATH=$(qmake -query
QT_INSTALL_LIBS):$HOME/oss/lib${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
```

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

This step is typically not necessary. One can need it in the case Qt libraries are installed locally, in a way bypassing package manager.

2.7 Run *bootstrap.sh* 5m

Make sure to also build tsv-utils, necessary to process visualization performance log files

```
./bootstrap.sh --with-tsv-utils
```

2.8 Build ReVisE with performance measurement enabled by running the build script like this 5m

```
./build.sh -DS3DMM_ENABLE_WORKER_TIME_ESTIMATION=ON
```

3 Source ReVisE environment script.

```
source <ReVisE directory>/scripts/env.sh
```

where <ReVisE directory> is the path to the directory ReVisE repository was cloned into.

4 Go to the directory where datasets are to be installed. Make sure to choose a location where sufficient free disk space is available (see table below).

```
cd <test datasets directory>
```

The table below provides details about datasets, such as download size, preprocessed dataset size, estimated preprocessing time, and memory (RAM) required for the preprocessing and visualization. Notice that the required RAM size is rounded to the nearest power of two.

Also notice that the downloading time is highly dependent on Internet connection speed, and maximum download speed available on our data hosting is 100 Mbit/s.

Dataset	RAM Required, GB	Preprocessing time, sec	Source dataset size, GB	ReVisE dataset size, GB
cascade	16	6000	27.9	30.4
hump	16	570	18.3	70
mwave	128	2300	0	5.8
sphere1M	8	13	0	0.015

Further steps can be performed either manually, using the detailed step-by-step instruction, or automatically with an interactive script. In the former case, it is easy to locate the problem if a command fails. In the latter case, it is convenient to have most of the work done by a single script.

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

To measure the performance, it is required to run a Web browser on the client machine. We run Web browser on a display with FullHD resolution (1920 x 1080 pixels). For other resolutions, test results might differ, as the resolution has major influence on rendering times.

Step 4 includes a Step case.

Step-by-step process

Using interactive script

Download and preprocess dataset

step case

Step-by-step process

In this case the process of ReVisE performance measurement is split into elementary commands, in order to provide an insight into the details of setting up the experiments and to provide full control over the measurement process.

5 Set REVISE_MACHINE environment variable, e.g, to the value returned by the hostname command.

```
REVISE_MACHINE=$(hostname)
```

Notice that the machine name appears further as the name of a subdirectory for performance log files.

6 Set the *REVISE_DATASET* environment variable.

```
REVISE_DATASET=<dataset name>
```

At the moment, <dataset name> may be one of the following:

- *cascade*
- *hump*
- *mwave*
- *sphere1M*

7 Download and preprocess the dataset.

```
revise_prepare_dataset.sh $REVISE_DATASET
```

If the download of the source dataset is interrupted due to an unstable internet connection, just repeat this step once again. The download will continue.

On rare occasion, the preprocessing may fail. In this case, the file $REVISE_DATASET/prepare.log will contain the word ERROR. If that happens, execute the following commands:

find $REVISE_DATASET -name "*.s3dmm*" -exec rm {} \;

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

rm $REVISE_DATASET/revise_ready

then repeat this step once again (no downloading will be repeated).

8 Prepare custom ReVisE Web server problem list files for testing the specified dataset.

```
revise_webdata.js install test_datasets.json
```

9 Add the ReVisE dataset, prepared at the previous step, to the list of problems available to the ReVisE Web server.

```
revise_install_dataset.sh $REVISE_DATASET
```

Measure visualization performance

10 Measure the performance of visualization using ReVisE.

10.1 Set the *REVISE_GPUS* environment variable to the number of GPUs to use in the test.

```
REVISE_GPUS=1
```

10.2 Set the *REVISE_WORKERS* environment variable to the number of rendering worker threads to use in the test.

```
REVISE_WORKERS=1
```

10.3 Set the REVISE_ASM_THREADS environment variable to the number of threads used to compose frame of several rendered parts, and write its value into the ReVisE hardware configuration file.

```
REVISE_ASM_THREADS=2
revise_hw_config.js set assemble_threads_per_node $REVISE_ASM_THREADS
```

For the value of this variable, we typically specify twice the total number of GPUs installed on the system (not *REVISE_GPUS*, which is the number of GPUs used in one measurement!), and never change the value during the test.

10.4 Write the number of GPUs and workers per GPU to use in the measurement to the ReVisE hardware configuration file.

```
revise_hw_config.js set gpus_per_node $REVISE_GPUS
revise_hw_config.js set render_threads_per_node $REVISE_WORKERS
```

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

10.5 Create directory for performance log files.

```
LOGDIR=log/$REVISE_MACHINE/$REVISE_DATASET
mkdir -p $LOGDIR
```

10.6 Start ReVisE Web server.

```
revise_web.sh
```

when the Web server is started for the first time, the script downloads and installs all necessary dependencies and builds two ReVisE modules that interoperate with the web server. This process takes a few seconds. When the server is started and ready to serve request, it displays the message
"Starting web server"

10.7 Prepare Web browser for measuring ReVisE performance.
1. Open Web browser (we use Chrome and Firefox, other browser should work too)
2. Type localhost:3000 in the address bar and press Enter.
3. Maximize browser window
4. Select problem: choose dataset name
5. Select field. For each dataset, a specific field is used in the performance test - see the table below.

cascade	mwave	hump	sphere1M
rho_nu	f	Nut	F

6. Open the *Settings* dialog. To do so press the button with the following icon:
settings.png

7. Press *Reset data*; wait a few seconds
8. Open the *Settings* dialog again. Set the value of *fovY* (camera field of view angle [deg] in the Y direction) from the table below; Fields are shown as relative values; Time interval, ms = 1; Press *Ok.*

cascade	mwave	hump	sphere1M
11	23	11	19

9. Select Visualization mode *MIP* in a dropdown list on the left side of the window.
10. Move to the left the slider below the visualization mode selector.
11.Stop web server: go to terminal where the web server is running in and press Ctrl+C.
12.Go to the Web browser and refresh the window; return to the terminal again.
13. Start ReVisE Web server again

```
revise_web.sh
```

13. Go to the Web browser. Open *Colormap settings* dialog pressing the button with the following icon:

14. Press *Ok.*
15. Stop web server; refresh browser window; return to terminal.
16. Remove timestamp log file.

```
rm $REVISE_ROOT_DIR/src/webserver/VsRendererDrawTimestamps.log
```

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

10.8 Measure ReVisE performance for specific hardware configuration.
Start ReVisE Web server again.

```
revise_web.sh
```

1. Refresh browser window. Press the following button (if it is present) to restore settings:

2. Wait till the last level is reported. The table below shows the last level number for each dataset, as well as the typical warming-up time - an estimation of time necessary to render the very first frame at the last level, for the minimal hardware configuration. Notice that the warming-up time will be eliminated in a future release of ReVisE.

Dataset name	Warmingup time, sec	Max level
cascade	20	3
mwave	500	2
hump	3	3
sphere1M	0	2

3. Press the button (its the icon is presented below) 10 times, each time waiting till the last level is reported.

z-pos.png

4. Stop web server; refresh browser window; return to terminal.
5. Move the timestamp log file to the log directory.

```
mv $REVISE_ROOT_DIR/src/webserver/VsRendererDrawTimestamps.log
$LOGDIR/VsRendererDrawTimestamps-$REVISE_GPUS-$REVISE_WORKERS-
$REVISE_ASM_THREADS.log
```

10.9 Measure ReVisE performance for more hardware configurations. At this point, repeat sequence of steps **go to step #10.1** , **go to step #10.2** , **go to step #10.4** , **go to step #10.8** multiple times, each time setting different values of REVISE_GPUS and REVISE_WORKERS. For example, on a DGX-1 machine that has 8 GPUs, we would take values from the table below:

Experiment No	REVISE_GPUS	REVISE_WORKERS
1	1	1
2	1	2
3	1	4
4	2	2
5	2	4
6	2	8
7	4	4
8	4	8
9	4	16
10	8	8
11	8	16
12	8	32

10.10 Process measurement results with the special tool.

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

```
revise_process_render_perf_logs.sh $LOGDIR
```

10.11 Print results.

```
cat $LOGDIR/total_avg.tsv
```

Compare test results with the expected results

11 Compare results of rendering performance measurements obtained in your tests against our results.

The expected result depends on the dataset and the target machine. The results obtained can be compared against our results. To do so, first download our log files, together with the results of processing:

```
curl https://ftp.mpksoft.ru/revise_datasets/orig_log_processed.tar.gz |tar -zx
```

This will result in a directory, named **orig_log_processed**. The content of the directory is structured in the same way as the **log** directory: it contains subdirectories <MACHINE>/<DATASET>, each of which contains corresponding log-files and results of processing. In particular, there are files orig_log_processed/<MACHINE>/<DATASET>/total_avg.tsv that can be compared against the results obtained by user. The orig_log_processed directory contains results for the following combination of MACHINE and DATASET:

MACHINE	hump	cascade	mwave
GeForce	x	x	-
Tesla	x	x	x
DGX-1	x	x	x

Parameters of tested machines are as follows.

parameter	GeForce	Tesla	DGX-1
CPU info	Intel i7-8700 CPU @ 3.20GHz	Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz	Intel(R) Xeon(R) CPU E5-2698 v4
CPU count	1	2	2
cores per CPU	6	12	20
RAM, Gb	16	128	512
GPU info	GeForce 1060 GPU (6 GB memory, 1280 CUDA cores)	Tesla V100-PCIE GPUs (32 GB memory, 5120 CUDA cores per GPU)	V100-SXM2 GPUs (16 GB memory, 5120 CUDA cores per GPU)
GPU count	1	2	8
remarks	-	Intel Optane 960 storage system	-

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

To compare your results against our results, find a closest match between your machine and our machine (GPU count, multiplied by the number of cores per GPU, is the most important parameter). Then compare the data in files
log/<YOUR_MACHINE>/<DATASET>/total_avg.tsv
and
orig_log_processed/<OUR_MACHINE>/<DATASET>/total_avg.tsv

12 Compare results of preprocessing time in your tests against our results.

ReVisE visualizes a dataset in a special format, obtained by preprocessing of the corresponding source dataset. The preprocessing is currently not parallelized (although different time steps can be preprocessed in parallel, the most time consuming preprocessing operations are done in a single thread of execution). Since we present preprocessing times in our paper, those times can also be compared with your test results.

There is a preprocessing log file, named **prepare.log**, that is written to the subdirectory of the corresponding dataset. The file contains information about the time of various preprocessing stages, as well as the total preprocessing time. The total preprocessing time, in seconds, can be extracted from the prepare.log file using the following command:

```
grep "finished: \\[Run the entire job" <DATASET>/prepare.log |sed -r "s/^.*+\\]
([^s]+)s.*$/\\1/"
```

where <DATASET> is the dataset name.

The table below presents total preprocessing times, in seconds, rounded off, that we have obtained on the Tesla machine (see table in Step 10).

sphere1M	hump	cascade	mwave
14	583	6227	1791

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited