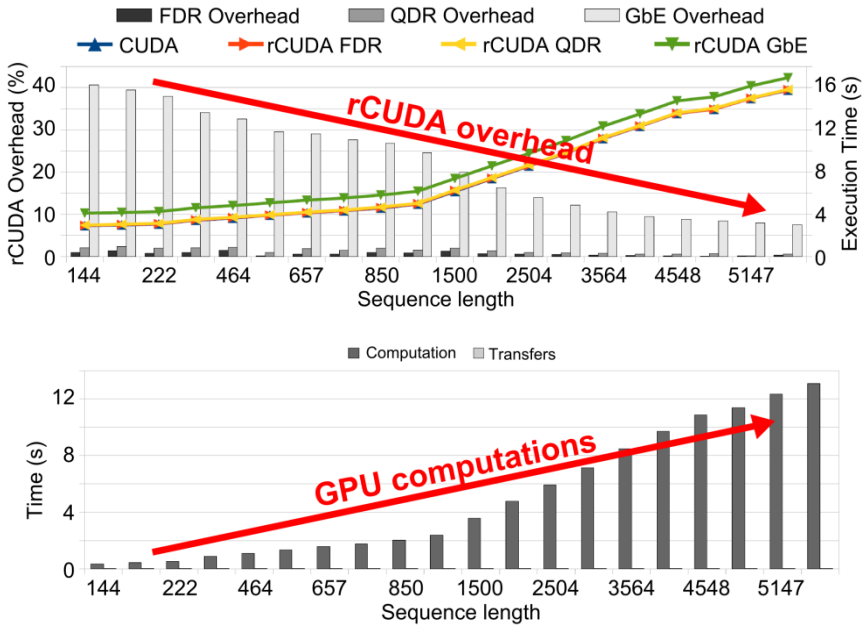
The rCUDA middleware and applications

Will my application work with rCUDA?

rCUDA currently provides binary compatibility with CUDA 8.0, virtualizing the CUDA API except for the graphics functions, which are rarely required in HPC scenarios. Regarding the network connecting the computer executing the application and the rCUDA server owning the real GPU, rCUDA provides specific RDMA support for the InfiniBand fabric and also supports the general TCP/IP stack on any interconnect. Therefore, **as long as your application is CUDA 8.0 compliant and your platform is equipped with an InfiniBand network or the network provides TCP/IP compatibility, your application will work.**

With rCUDA, which performance should I expect for my application?

rCUDA makes use of remote GPUs instead of local ones. Therefore, some performance loss is expected due to the longer distance to the GPU. The exact reduction in performance depends on the application being run. Applications such as LAMMPS, Gromacs, MAGMA, CUDASW++, HOOMD-Blue, mCUDA-MEME, and GPU-BLAST have been tested with rCUDA. Visit www.rcuda.net for an updated list of applications. Below we report performance examples for some of them.



CUDASW++

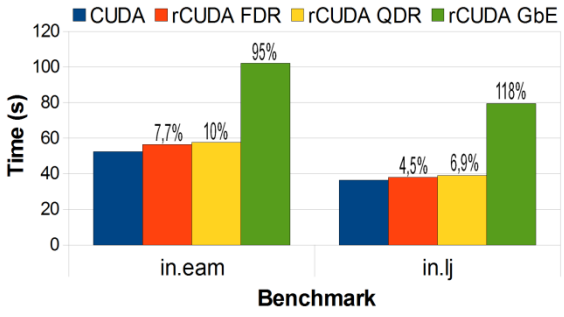
Execution time for queries of different sequence lengths, using CUDA and rCUDA over different networks. Primary Y-axis shows rCUDA's overhead. Secondary Y-axis depicts execution time.

The lower plot shows the time employed by computations (CUDA kernels) and by memory transfers (CUDA memcopy).

The conclusion is clear: **the more GPU computations, the less rCUDA overhead.**

Performance for other applications

LAMMPS executing in.eam and in.lj, scaled by a factor of 5 in all three dimensions. Numbers over the bars are the overhead with respect to CUDA.



rCUDA throughput with other interconnects

rCUDA can use the FDR and EDR versions of the InfiniBand fabric. By using this fabric, the bandwidth attained by rCUDA is almost the same as in the local case using regular CUDA.